A Study of Syntactic Typed-Dependency Trees for English and Japanese and Graph-Centrality Measures

A doctoral dissertation
submitted in the partial satisfaction of the requirements
for the degree of Doctor of Education

Waseda University

Masanori Oya
2014

Acknowledgements

First of all, I am truly honored to express my deepest gratitude to my supervisor, Dr. Michiko Nakano, for her critical, profound and wide-reaching suggestions on my work, and also for her longstanding and enthusiastic help for my study as a graduate student at Waseda University, and as a graduate student at School of Computing, Dublin City University. Her suggestions substantially improved the quality of this dissertation, and she taught me a variety of significant ideas through her teaching and work at Waseda University. Moreover, her recommendation of me to Prof. Josef van Genabith at School of Computing, Dublin City University, gave me the opportunity to study there as a fully-scholarshipped graduate student. I would also like to thank Dr. Masakazu Kuno, Dr. Tomoko Okuma, Dr. Ryo Otoguro, and Dr. Yasuyo Sawaki for their careful reviews and critical suggestions.

I would like to thank all my colleagues at Graduate School of Education, Waseda University: Dr. Junko Negishi, Koichi Ano, Kazuharu Owada, Norifumi Ueda, Kanji Horiguchi, Eiichiro Tsutsui, Dr. Yusuke Kondo, Aya Kitagawa, Akiko Watanabe, Kota Wachi, Satoshi Yoshida, Yutaka Ishii, and all the other members of Dr. Nakano's laboratory. Thanks to them, I have been enjoying my academic and social life in Tokyo.

Special thanks go to my parents, Tetsuro Oya and Sanae Oya, who always wish me good health and a happy life. Thank you very much indeed, and I wish you good health and longevity.

Lastly, but not the least, I thank my wife Seiko for her wholehearted help and determined encouragement for me to complete this dissertation. Seiko, I love you, and your smile.

Table of Contents

List of Figures

List of Tables

1. Introduction

The aim of this thesis is to introduce the typed-dependency trees for English and Japanese sentences, and to introduce graph-centrality measures to capture the structural characteristics of these typed-dependency trees. Typed-dependency trees are syntactic structures for sentences that illustrate the dependency relationships among the words in a sentence as a network of words. The structural characteristics of the network can be captured by a number of measures that have been developed in the field of graph theory and network analysis. Introducing these measures into the typed-dependency trees for sentences allows us to capture the structural characteristics of these sentences as networks of words, and ultimately, this analysis sheds new light on Japanese and English speakers' syntactic intuitions.

In order to accomplish this aim, this thesis asks several questions. First, *what are the dependency relationships among the words in a sentence*? Chapters 2 and 3 answer this question. Chapter 2 introduces the concept of dependency grammar through a discussion of Tesnière's (1959) seminal assumption about dependency along with more recent theories of dependency grammar proposed by I. Mel'čuk and his colleagues (Iordanskaja & Mel'čuk 2000; Mel'čuk & Pertsov 1987; Mel'čuk 1988, 2003, 2004, 2009, and 2011). This chapter also addresses the difference between dependency grammar and phrase-structure grammar. Section 2.2 presents an overview of dependency grammar and Section 2.3 focuses specifically on Tesnière's (1959) seminal work on dependency grammar. Section 2.4 discusses Mel'čuk's work on Deep Syntactic Relations and Surface Syntactic Relations as a development of Tesnière's (1959) concept of dependency. Finally, the difference between dependency and phrase-structure grammar is briefly discussed in Section 2.5 with reference to Osborne, Putnam, & Gross (2011). Chapter 3 examines whether a typed-dependency tree for a sentence is equivalent to a functional-structure representation according to Lexical-Functional Grammar (LFG) (Bresnan

1978; Bresnan 1982; Kaplan & Bresnan 1982; Bresnan 2001). Both LFG and dependency grammar theory assume that the individual pieces of lexical information contained in a sentence are integrated into the whole, and both frameworks are concerned with making explicit the process through which these pieces of lexical information are integrated. Dependency grammar does so at only one level of representation (i.e., a dependency tree for a sentence), while LFG does so by connecting multiple levels of representation (i.e., constituent structure, functional structure, argument structure, and phonological structure). The idea of structural correspondence in LFG can be seen as an extension of typed-dependency tree representation of grammatical knowledge. In this sense, LFG represents one direction of development of the dependency grammar tradition started by Tesnière (1959). In Chapter 3, the revised version of Mel'čuk's Criteria for surface syntactic dependencies is proposed; the idea behind this revision is that two words in a sentence establish a dependency relationship iff they constitute one fragment functional structure.

Second, *what are the graph centrality measures, and how are they calculated?* Chapter 4 answers this question by introducing the representation of a typed-dependency syntactic tree as a directed acyclic graph (DAG) and examining the idea of quantifying the structural property of typed-dependency trees in terms of graph centrality. The advantage of dependency grammar representation is that a sentence's dependency can be interpreted as a DAG, allowing the formal syntactic properties to be defined and analyzed mathematically in terms of graph theory (Oya 2010b, 2011, 2013a, and 2013b). Dependency grammar makes explicit the connections among the words in a sentence, or the *network* of words (Tesnière 1959). Approaches in the field of graph theory and network analysis can help make salient the characteristics of these networks. In other words, the structural properties of networks of words in sentences can be made explicit in dependency grammar and then quantified by applying graph theory. Quantified structural properties are useful for linguistic analyses that have previously relied on the subjective judgment of researchers, such as investigations into stylistic differences across different genres or similarities in syntactic structures across different languages. Quantitative approaches to syntactic structure contribute to these types of linguistic analyses by incorporating more objectivity. The centrality measures used in this thesis are *degree centrality* and *closeness centrality*, based on Freeman 1979 and Wasserman & Faust 1994. Degree centrality of a given typed-dependency tree indicates how flat the tree is, while Closeness centrality of a given typed-dependency tree indicates how embedded the tree is (Oya 2010b).

Third, *how can we obtain the typed-dependency trees for given sentences, and what are their characteristics*? Chapter 5 answers this question for English sentences, and Chapter 6 for Japanese sentences. Chapter 5 introduces the Stanford Parser (Klein & Manning 2003; de Marneffe & Manning 2012), along with the definition of each dependency type according to the revised version of Mel'čuk's Criteria introduced in Chapter 3. Stanford Parser is a state-of-the-art parser used in this study for acquiring typed-dependency tree representations for English sentences. In traditional analyses, it is time-consuming for the researcher to construct typed-dependency trees for each sentence in a corpus and manually calculate their centrality measures. This chapter proposes this syntactic parser as a more efficient method to obtain typed-dependency trees for individual sentences in large corpora. Each dependency type is defined according to the revised version of Mel'čuk's criteria, which is proposed in Chapter 3, so that it is based on a tradition of dependency grammar which was started by Tesnière and developed by Mel'čuk. The functional structures for example sentences are also provided in this chapter, so as to examine the equivalence between the typed-dependency tree for a sentence and its functional-structure representation, which is proposed in Chapter 3. Chapter 6 introduces another parser for Japanese called KNP (Kurohashi & Nagao 1992, 1994, 1998; Kawahara & Kurohashi 2007), including the dependency-type annotation for KNP output and the definition of each dependency type and functional structures for sentences containing each dependency type. KNP is a rule-based dependency parser used for generating automatic dependency tree representations for Japanese sentences. The accuracy of this parser has been improved since its use in the development of Kyoto University Text Corpus ver. 4, a parsed corpus of Japanese (Kurohashi & Nagao 1998). Since the parsed output of KNP does not contain the type of each dependency, it is necessary to annotate the parsed output. Doing so allows us to use the KNP output to obtain cross-linguistic typed-dependency tree representations of Japanese. The annotated dependency types must be based on a tradition of dependency grammar which was started by Tesnière and developed by Mel'čuk. Similarly to dependency types of English, dependency types of Japanese are defined according to the revised version of Mel'čuk's criteria, which is proposed in Chapter 3. The functional structures for example Japanese sentences are also provided in this chapter, so as to examine the equivalence between the typed-dependency tree for a sentence and its functional-structure representation, which is proposed in Chapter 3.

Fourth, *from which source are the graph centrality measures obtained, and what is the result?* Chapter 7 answers this question. The accuracies of the Stanford Parser and the KNP are examined by comparing the typed-dependency trees obtained from the parsed output of the English sentences and their Japanese counterparts in a small-scale parallel corpus (Iida 2010) to their manually corrected typed-dependency trees. Results show that the distributions of both degree centralities and closeness centralities before and after manual corrections are almost identical. Thus, the Stanford Parser and KNP are accurate enough to obtain degree centralities and closeness centralities. Next, the distributions of degree and closeness centralities for English typed-dependency trees are compared to those for their Japanese counterparts, and results show that their distributions are different. Thus, the structural properties of the typed-dependency trees for sentences in these two languages are different in terms of their degree centralities (flatness) and closeness centralities (embeddedness). Lastly, the distributions of degree centralities and of closeness centralities obtained from the parsed output of sentences from different genres of texts in Manually Annotated Sub-corpus of American National Corpus (MASC 500k) (Ide, Baker, Fellbaum, Fillmore, & Passonnau 2008) are compared to each other. It is shown that sentences from different genres have different distributions of these measures; sentences in the subsections Fiction, Ficlets and Jokes are flatter and more embedded than sentences in other subsections. However, it is pointed out that these different distributions are dependent on the word counts of the sentences. It is also pointed out that controlling the word count of the sentences taken from different genres could make explicit that difference in genre is reflected on the number of sentences of the same degree centrality and of the same closeness centrality.

2. Dependency Grammar

2.1 Introduction

This chapter introduces the concept of dependency grammar through a discussion of Tesnière's (1959) seminal assumption about dependency along with more recent theories of dependency grammar proposed by I. Mel'čuk and his colleagues (Iordanskaja & Mel'čuk 2000: Mel'čuk & Pertsov 1987; Mel'čuk 1988, 2003, 2004, 2009, and 2011). This chapter also addresses the difference between dependency grammar and phrase-structure grammar. Section 2.2 presents an overview of dependency grammar and Section 2.3 focuses specifically on Tesnière's (1959) seminal work on dependency grammar. Section 2.4 discusses Mel'čuk's work on Deep Syntactic Relations and Surface Syntactic Relations as a development of Tesnière's (1959) concept of dependency. Finally, the difference between dependency and phrase-structure grammar is briefly discussed in Section 2.5 with reference to Osborne et al. (2011).

2.2 Dependency Grammar: an Overview

The role of dependency in syntactic representations of a sentence has a long history in linguistic inquiry. Gerdes, Hajičová, & Wanner (2011) point out that Ibn Mada, a 12th-century Cordobian, first used the term *dependency* in the grammatical sense. More recent approaches to dependency grammar can be traced back to Tesnière (1959). Some argue that his work has been somewhat obscure in the field of syntax because of the advance of Chomskyan generative syntax. However, dependency grammar has been used in computational linguistic research, such as work on ontology construction (Snow, Jurafsky, & Ng 2005), machine translation (Ding & Palmer 2004a, 2004b, and 2005), and parsing (Buchholz & Marsi 2006; Nivre, Hall, Kübler, McDonald, Nilsson, Riedel, & Yuret 2007).

Dependency representations have advantages over phrase-structure grammar representations because of their "conciseness, intuitive appeal, and closeness to semantic representations such as predicate-argument structures" (Debusmann & Kuhlmann 2007, p.1). In fact, McDonald & Nivre (2011, p.198) argue that the advantage of dependency representations is their "natural mechanism for representing discontinuous constructions, which arise due to long-distance dependencies or in languages where grammatical relations are often signaled by morphology instead of word order."

2.3 Tesnière's (1959) Dependency Grammar

Our current understanding of dependency grammar has its origins in the following assumption about dependency by Tesnière (1959, p.12-14) (translation from French by the author):

> Les connections structurales établissent entre les mots des rapports de **dépendance**. Chaque connexion unit en principe un terme **supérieur** à un terme **inférieur**.
> (The structural connections among words establish dependency relations. In principle, each connection unites a superior term and an inferior term.)
>
> Le terme supérieur reçoit le nom de **régissant**. Le terme inférieur reçoit le nom de **subordonné**. Ainsi dans la phrase *Alfred parle* …, *parle* est le régissant et *Alfred* le subordonné.
> (The superior term is called "régissant" (*governor*). The inferior terms are called "subordonné" (*dependent*). For example, in the phrase "Alfred parle (Alfred speaks)," 'parle' is the governor and 'Alfred' the dependent.)
>
> On exprime la connexion supérieure en disant que le subordonné **dépend** du régissant, et la connexion inférieure en disant que le régissant **commande** ou **régit** le subordonné. Ainsi dans la phrase *Alfred parle* …, *Alfred* dépend de *parle*, tandis que *parle* commande *Alfred*.
> (The superior connection can be expressed by saying that the dependent depends on the governor, and the inferior connection can be expressed by saying that the governor commands or governs the dependent. In the example above, 'Alfred' depends on 'parle,' and 'parle' commands on 'Alfred.')

> Un mot peut être à la fois subordonné à un mot supérieur et régissant d'un mot inférieur. Ainsi dans la phrase *mon ami parle …*, *ami* est à la fois le subordonné de *parle* et le régissant de *mon*.
> (A word can be a dependent to a superior word and a governor to another, inferior word at the same time. For example, in the phrase "mon ami parle" (My wife speaks), 'ami' is the dependent of 'parle' and the governor of 'mon' at the same time.)
>
> L'ensemble des mots d'une phrase constitue donc une véritable **hiérarchie**. …
> (The ensemble of words in a phrase constitutes an actual hierarchy. …)
>
> L'étude de la phrase, qui est l'objet propre de la syntaxe structurale …, est essentiellement l'étude de sa structure, qui n'est autre que la **hiérarchie de ses connexions**.
> (The study of a phrase, which is the object of structural syntax, is essentially the study of its structure, which is nothing other than the hierarchy of its connections.)
>
> Le trait de connexion sera en principe **vertical** ..., puisq'il symbolise le lien entre un terme supérieur et un terme inférieur.
> (The character of the connection will be vertical in principle, because it symbolizes the line between a superior term and an inferior term.)

This account describes the core tenets of Tesnière's dependency grammar theory that have been extended by later researchers. These tenets propose that each word in a sentence is dependent on another word, no word in a sentence is independent, and the dependency relationship between words is characterized by a governor and a dependent.

However, Tesnière's (1959) concept of dependency does not provide a full description of the structure of a sentence and raises several questions. For example, what principle determines which words function as governors and which words function as dependents? Are all the dependent relationships among words in a sentence the same? Furthermore, what is the difference between dependency grammar and phrase-structure grammar? Moreover, is the concept of dependency a universal feature of language? Following Tesnière (1959), a number of researchers have tried to answer these questions, and their research is summarized in the following section.

2.4 Mel'čuk's Dependency Grammar

Igor Mel'čuk is one of the most prominent linguists who has worked within the framework of dependency grammar, focusing on the formalisms of this theory (Mel'čuk 1988, 2003, 2004, 2009, and 2011). In his work, he takes a *Meaning-Text* approach and examines linguistic forms "from meaning to text" (Mel'čuk 2011, p.2). He also argues that linguists need different formalisms for different levels of linguistic representation and a set of rules that govern the relationships between these formalisms. Specifically, he poses three types of relationships: semantic dependency ("Sem-D" in his terminology, a predicate-argument relations), syntactic dependency ("Synt-D"), and morphological dependency ("Morph-D;" agreement relations between words[1]). Syntactic dependency "determines the distribution of the phrase within sentences" (Mel'čuk 2011, p.3). In other words, the position of a word is determined by its governor. Mel'čuk considers this general relationship to be a universal feature of dependency in language; whether the dependent or the governor comes first is a language-specific feature. For example, in the English phrase 'red books,' there is a dependency relation from 'book' to 'red.' The governor in this dependency relation is the noun 'book' and the dependent is the

[1] The distinction between syntactic dependency and morphological dependency is relevant to the constructions in which two words that agree with each other morphologically are not in syntactic dependency relationship. One of such cases is the subject-verb agreement, such as 'David has written this book.' Morphologically, the subject agrees with the auxiliary because the former depends on the latter. Syntactically, on the other hand, the subject and the auxiliary have no dependency relationship; hence, the subject-verb agreement is not necessarily represented in, or defined in terms of, the typed-dependency tree. See Section 3.3 for the treatment of morphological agreement in functional structures.

adjective 'red.' In English, if the dependent is an adjective and the governor is a noun, the dependent is positioned before its governor. Different languages have different positions for a noun governor and an adjective dependent (e.g., French *livres rouges*) as well as for governors and dependents in other lexical categories.

2.4.1 Deep-Syntactic representation and Surface-Syntactic representation

Mel'čuk distinguishes two levels of linguistic representation in syntax: *Deep-Syntactic representation* (DSyntR) and *Surface-Syntactic representation* (SSyntR). DSyntR is language-independent and SSyntR is language-specific. In order to explain the relationship between DSyntR and SSyntR in different languages, Mel'čuk (2011, p.5) examines the different grammatical functions of the English verb 'help' and its Russian equivalent 'pomogat' by showing that the former takes a direct object and the latter takes an indirect object. The difference of grammatical functions in these examples is represented at SSyntR.

(2.1)

a. Sarah helps David.

b. Сара помогат Давиду.
   Sarah pomogat David-u
   Sarah help-sg.3rd David-IO

At the DSyntR level, these two constructions are "homogenized." The different grammatical functions of these two constructions are integrated into one deep syntactic relation called "II."

(2.2)

a. Help =II=> David

b. Pomogat =II=> David

The number “II” is one of the types of deep syntactic relations (DSyntRels in Mel’čuk’s terms). This type of syntactic relation is discussed in more detail in the next section.

2.4.2 DSyntRels: language-independent dependency relations

Mel’čuk (2011, p.6) describes twelve types of DSyntRels, as shown in Table 2.1.

Table 2.1. Inventory of DSynt-Rels in Mel’čuk (2011, p.6)

coordinate DSyntRels		subordinate DSyntRels									
		weak subordi-nate DSyntRel	strong subordinate DSyntRels								
			modification		complementation						
COORD	QUASI-COORD	APPEND	ATTR	ATTRdescr	I	II	III	IV	V	VI	IIdir-sp

These DSyntRels are characterized by five binary oppositions, discussed in turn below.

1. Coordinations vs. Subordinations

Coordinate constructions connect words or phrases. In Mel’čuk’s categorization of DSyntRels, coordinate constructions are divided into two types: COORD and Quasi-COORD. COORD refers to the dependency relations among coordinates of words. For example, in ‘Sarah, David and Abraham have read this book,’ there are two DSyntRels categorized as COORD: Sarah=COORD=>David=COORD=>Abraham. QUASI-COORD refers to the dependency relations among coordinates of prepositional phrases. For example, in ‘David is now at his dormitory on Santry in Dublin,’ there are two QUASI-COORDs: at his dormitory =QUASI-COORD=> on Santry =QUASI-COORD=> in Dublin.

Subordination constructions include all constructions that are not characterized by coordination. These constructions can be categorized as either weak or strong, and this binary opposition is discussed in more detail below.

2. Weak subordinations vs. Strong subordinations

Weak subordinations do not have strong structural links. In Mel'čuk's categorization of DSyntRels, weak subordinations are called APPEND. For example, in 'David is, *surprisingly*, present at the meeting,' the dependency between 'is' and 'surprisingly' is a DSyntRel categorized as APPEND. This dependency is represented as follows: is =APPEND=> surprisingly.

On the other hand, strong subordinations are described in terms of the binary opposition between modification and complementation, discussed in the next section.

3. Modification vs. Complementation

Modification involves a DSyntRels in which the dependent modifies its governor (e.g., David works *hard*). There are two types of modification: restrictive modification and descriptive modification, discussed in the next section.

Complementation, on the other hand, involves a DSyntRels in which the dependent is a complement of its governor. Complementation is further characterized according to seven actantial roles, discussed in Section 5 below.

4. Restrictive Modification vs. Descriptive modification

Restrictive modification involves a DSyntRels (called ATTR) in which the dependent identifies the governor. For example, in ‘Sarah reads only interesting books,’ there is one ATTR: books=ATTR=>interesting.

Descriptive modification involves a DSyntRels (called ATTRdescr) in which the dependent describes the governor. For example, in ‘Sarah, who has read this book, says it is interesting,’ there is one ATTRdescr: Sarah=ATTRdescr=>read.

5. Different Actantial Roles

Actantial roles are related to Tesnière’s (1959) idea about the syntactic roles of nouns in relation to verbs, i.e., subject, direct object, and indirect object. Tesnière (1959, p.102) coined the word *actant* and *circumstant* as follows:

> The verbal node [in a clause] … expresses a whole little drama. As a drama, it implies a process and, most often, actors and circumstances. The verb expresses the process …. Actants are beings or things that … participate in the process … Circumstants express the circumstances of time, place, manner, etc. [translation in Mel’čuk (2004)]

The idea of actants and circumstants is further developed in Mel’čuk (2004) whereby three types of actants are proposed: semantic actants, deep-syntactic actants, and surface-syntactic actants. Semantic actants correspond to *argument structure* (Grimshaw 1990), and syntactic actants correspond to *grammatical relations*. Deep-syntactic actants (DSynt-As) are divided into seven categories numbered with Roman numerals:

DSyntA I: the subject in the surface syntactic relation

DSyntA II: the direct object in the surface syntactic relation, or the complement of a preposition or conjunction in the surface syntactic relation

DSyntA III: the indirect object in the surface syntactic relation

DSyntA IV - VI: more oblique or prepositional object

DSyntA IIdir-sp: direct speech, such as ‘I have read this book,’ said Sarah.

2.4.3 SSyntRels: language-specific dependency relations

Given the abstract nature of DSyntRel, it is necessary to establish the criteria for surface syntactic relationships (SSyntRel in Mel’čuk’s terms), or language-specific dependency relations, for a particular language. Mel’čuk (2009, 2011) proposes three types of criteria for SSyntRels for particular languages. First, Criterion A accounts for the SSyntRel between two words. This criterion explains why two given words have a dependency relation in a sentence. Second, Criterion B accounts for the orientation of a SSyntRel between two words. This criterion explains which word functions as the governor. Third, Criterion C accounts for the type of a SSyntRel. This criterion explains which type of dependency connects the two words. Each criterion is discussed in turn below.

2.4.3.1 Criterion A: the presence of a SSyntRel between two words

Mel’čuk (2011, p.6) states that Criterion A determines the presence of a SSyntRel in the following way:

> Criterion A:
>
> In a sentence, the lexemes L1 and L2 have a direct Synt-D link, only if L1 and L2 can form in language L an utterance – i.e., a *prosodic unit*, or a *prosodic phrase* of L – such as *the*

> *window*, *of John*, *spouts water* or *stained glass*, out of any context; the linear position of one of these lexemes in the sentence must be specified with respect to the other. (Mel'čuk 2011, p.6)

For example, the phrase 'the window' has the following dependency relation:

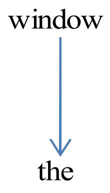


Figure 2.1. The dependency relation between 'the' and 'window' in the phrase 'the window'

The former part of the definition of Criterion A is ambiguous, because Mel'čuk (2011) does not make explicit what he means by the phrase *prosodic units* or *prosodic phrases.*[2] In addition, there are instances of dependency relationships in which the head and its dependent do not constitute what can be called a prosodic unit (e.g., 'Sarah' and 'read' in the sentence 'Sarah has read this book.'). For example, a subject-taking element and its dependent usually constitute a prosodic unit (e.g., 'Sarah reads'), and their linear order is such that the noun precedes the subject-taking element. However, the subject-taking element and its dependent do not constitute a prosodic unit if a subject-taking element is accompanied by an auxiliary, which is the case for the present perfect aspect (e.g., 'has gone'), the present progressive aspect or the passive

[2] It is not necessarily irrelevant to define the possibility for words to be in a dependency relationship in terms of prosody. For example, Ohsuga, Horiuchi and Ichikawa (2003) introduced a method for estimating the syntactic structure of Japanese speech using two prosodic features (F0 contour and pause duration). They defined prosodic units as being "divided by the local minimal point of an F0 contour or pause" (Ohsuga et al. 2003, p.558).

voice (e.g., ‘is going’ or ‘is taken’), or modal auxiliaries (e.g., ‘can’ or ‘ought to’). For example, ‘Sarah’ and ‘gone’ do not constitute a prosodic unit in ‘Sarah has gone there.’ Thus, we need to revise Mel’čuk’s (2011) Criterion A in order to account for cases with a subject-taking element and its dependent in which the two words do not constitute a prosodic unit. Specifically, the subject-taking element and its dependent must be defined in terms of the type of subject-taking element, such as its argument structure (Grimshaw 1990) or its lexical form according to Lexical-Functional Grammar (Kaplan & Bresnan 1982). This logic applies to dependency types other than subjects (see Section 5.3 for other dependency types in English, and Section 6.4 for those in Japanese).

Instead of defining the possibility for two words to be in a dependency relationship in terms of prosody, it is preferable to define this possibility in terms of semantics. In other words, two words can have a dependency relationship if they function as a *semantic unit*. In this way, the word ‘Sarah’ and the word ‘gone’ in the example above can have a dependency relationship because they form a semantic unit in the sentence; the word ‘gone’ is a subject-taking element and the word ‘Sarah’ can function as the subject of the word ‘gone.’ With respect to this, Oya (2013c, p. 29) proposed a tentative revision of Mel’čuk’s (2011) Criterion A as follows:

Criterion A (revised):

In a sentence, the lexemes L1 and L2 have a direct Synt-D link, only if L1 and L2 can form *a semantic unit* in language L.

This revision needs more clarification in terms of the definition of semantic units. In the example ‘Sarah has gone there,’ the semantic unit is considered to be a subject-taking element and its subject. However, this is not the only way a semantic unit can be constructed. We return to this issue in Section 3.3 by comparing the functional-structure representation and the typed-dependency tree representation for a sentence, and in Section 5.3 during a discussion of the definition of dependency types in Stanford Dependencies (de Marneffe & Manning 2008, 2012, 2013).

2.4.3.2 Criterion B: the orientation of a SSyntRel between two words

Mel'čuk's (2009, 2011) Criterion B involves identifying the governor of a given dependency relation. He divides Criterion B into three subcriteria: syntactic (Criterion B1), morphological (Criterion B2), and semantic (Criterion B3) SSyntRels. He argues that these subcriteria are hierarchically ordered; B1≻B2≻B3. In other words, Criterion B2 is applied if, and only if, Criterion B1 cannot determine the governor of a given dependency relation, and Criterion B3 is applied if, and only if, Criterion B2 cannot determine the governor. These subcriteria are further explained below.

Criterion B1

Criterion B1 has to do with the ability of a word to be dependent on another word. Mel'čuk (2011, p.7) states Criterion B1 as follows:

> Criterion B1 (syntactic):
>
> In the syntactic phrase L1=synt=L2, the lexeme L1 is the Synt-governor, if the *passive SSynt-valence* of the whole phrase is determined to a greater extent by the passive SSynt-valence of L1 rather than by that of L2. (Mel'čuk 2011, p.7)

The passive (SSynt-) valence of a lexeme refers to "its ability to be subordinated, in a specified role, to lexemes of a certain class." (Mel'čuk & Pertsov 1987, p.80). For example, the passive valence of the phrase 'for Sarah' is determined by the preposition 'for'; in 'David has read this book for Sarah,' it is the preposition 'for,' not 'Sarah,' that depends on the verb 'read' and also acts as its modifier. Therefore, the orientation of the SSyntRel between 'for' and 'Sarah' is for=synt=>Sarah.

Criterion B2

Criterion B2 has to do with the inflectional marking on the lexemes. Mel'čuk (2011, p.7) states Criterion B2 as follows:

> Criterion B2 (morphological):
> In the syntactic phrase L1=synt=L2, the lexeme L1 is the Synt-governor, if L1 controls the inflection of lexemes external to the phrase or its own inflection is controlled by such lexemes. (Mel'čuk 2011, p.7)

For example, in the English phrase 'operations manager' in (2.3), the Synt-governor is 'manager,' not 'operations,' because the verb agrees with 'manager' and not 'operations.' The noun 'manager' controls the inflection of the verb 'rejects,' which is external to the phrase 'operations manager.'

(2.3)

a. The operations manager rejects Sarah's proposal.

b. *The operations manager reject Sarah's proposal.

There are cases whereby Criterion B2 is not satisfied, yet a dependency relation between particular lexical categories in a dependency tree is still considered valid in a dependency grammar framework. For example, in Stanford Dependencies (de Marneffe & Manning 2008, 2012, 2013), the root of the dependency tree for a sentence is its main verb, regardless of the presence of an auxiliary (or auxiliaries). Consider the sentence (2.4) below along with the dependency tree in the style of Stanford Dependencies (Figure 2.2), in which the auxiliaries depend on the main verb[3].

(2.4)

Sarah has read this book.

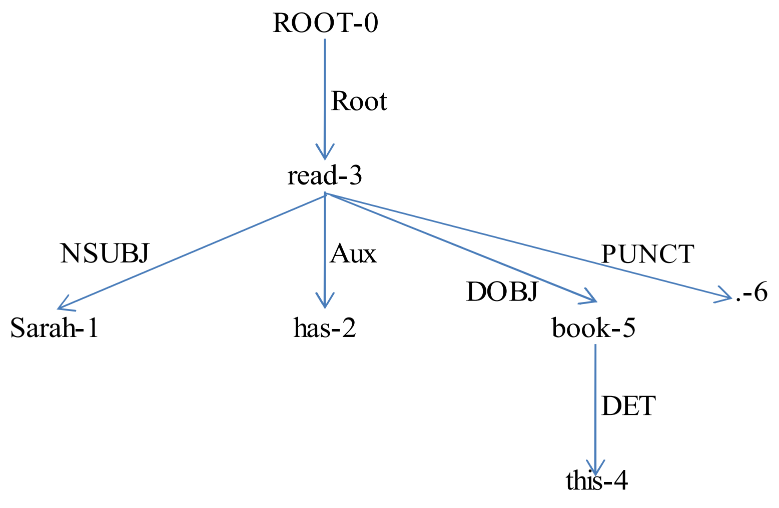


Figure 2.2. The dependency tree for ‘Sarah has read this book.’

[3] In this dissertation, the number next to each word specifies the place in which it appears in the sentence, just like the output style of the Stanford Parser.

For the phrase ‘has read,’ if the head is the word “has” and its dependent is the word “read”, contrary to the style of Stanford Dependencies, this phrase satisfies Criterion B2; the inflection of the word “has” is controlled by the word “Sarah”, which is external to the phrase.

Criterion B3

Criterion B3 involves the semantics of the words. Mel’čuk (2011, p.7) states Criterion B3 as follows:

> Criterion B3 (semantic):
> In the syntactic phrase L1=synt=L2, the lexeme L1 is the Synt-governor, if L1=synt=L2 denotes a kind/an instance of the denotation of L1 rather than a kind/an instance of the denotation of L2. (Mel’čuk 2011, p.7)

For example, the noun compound ‘a white house’ has a dependency relation ‘house=synt=>white’ where the governor is ‘house’ and the dependent is ‘white’ because this phrase denotes a kind of house, rather than a kind of white.

2.4.3.3 Criterion C: the type of a SSyntRel between two words

Mel’čuk’s (2009, 2011) Criterion C clarifies the different types of syntactic dependencies. He divides Criterion C into three subcriteria, namely *absence of semantic contrast*, *syntactic substitutability*, and *repeatability with the same Synt-governor*. For a given dependency L_1=**r**=>L_2, if at least one of these subcriteria of Criterion C is satisfied, the SSyntRel **r** must be categorized as a unique type. These subcriteria are discussed in turn below.

Criterion C1 (minimal pairs): Absence of semantic contrast

Criterion C1 has to do with the absence of semantic contrast for two SSyntRels. Mel’čuk (2011, p.8) states Criterion C1 as follows:

> An SSyntRel **r** cannot describe two phrases $W_1(L_1)$=**r**[?]=>$W_2(L_2)$ and $W_3(L_1)$=**r**[?]=>$W_4(L_2)$, which 1) contrast semantically and 2) differ formally by some syntactic means of expression – i.e., by word order, syntactic prosody or syntactic grammemes. (the notation “W(L) denotes “a wordform w of the lexeme L”) (Mel’čuk 2011, p.8)

An example of how Criterion C1 is applied to English word order is found in sentences (2.5a and b).

(2.5)

a. Sarah saw David.

b. David saw Sarah.

The dependency relations between ‘saw=r[?]=>Sarah’ in (2.5a) and ‘saw=r[?]=>Sarah’ in (2.5b) contrast semantically and also differ formally, in this case by word order. Therefore, the SSyntRels of ‘saw=r[?]=>Sarah’ in (2.5a) must be different from the SSyntRels of ‘saw=r[?]=>Sarah’ in (2.5b).

In principle, this criterion states that a particular dependency type can imply a certain kind of semantic relationship between two words, and this semantic relationship is different from what is implied by another dependency type.

Criterion C2 (substitutability in context): syntactic substitutability

Criterion C2 involves substitutability in context. Mel'čuk (2011, p.8) states Criterion C2 as follows:

> An SSyntRel **r** of L must possess the following (= "quasi-Kunze") property: L has a syntactic class X, different from substitute pronouns and such that, for any SSynt-phrase L=**r**=>$D_{(Y)}$, replacing $\Delta_{(Y)}$ by $\Delta_{(X)}$ (but not necessarily vice versa!) in any SSyntS of L does not affect its syntactic well-formedness. (Mel'čuk 2011, p.8)

Mel'čuk then paraphrases this definition by stating that "an SSyntRel must have a prototypical Dependent, which passes with any possible Governor" (Mel'čuk 2011, p.8).

Thus, this criterion states that the type of a given dependency determines the prototypical dependent. Moreover, the syntactic well-formedness of the dependency for a given type (e.g., *subj*) is not affected by replacing its dependent A with a new dependent B, as long as A and B are both prototypical dependents of the dependency type *subj*.

Criterion C3 (repeatability):

Criterion C3 has to do with repeatability with the same Synt-governor. Mel'čuk (2011, p.8) states Criterion C3 as follows:

> A SSyntRel **r** must be either non-repeatable (= no more than one branch labeled **r** can start from a Synt-governor) or unlimitedly repeatable (= any number of branches labeled **r** can start from a Synt-governor). (Mel'čuk 2011, p.8)

However, Criterion C3 does not function as a clear criterion for differentiating two SSyntRels; rather, it simply *describes* some of the properties that SSyntRels can have. For example, dependency types for the arguments of a verbal predicate are not repeatable, while those for the adjuncts to a verbal predicate are repeatable[4].

Taking into account these three subcriteria, Mel'čuk's Criterion C must be revised so that it can define a given dependency type in a more straightforward manner. First, the type of a given dependency implies a certain kind of semantic relationship between the governor and its dependent. Thus, I propose to revise Mel'čuk's Criterion C1 as follows:

Criterion C1 (revised):

In the syntactic phrase L1=synt=L2, the syntactic dependency must be categorized into a type so that the type implies a unique semantic relationship between the governor and its dependent.

Second, the type of a given dependency also determines the prototypical dependent. Thus, I propose to revise Mel'čuk's Criterion C2 as follows:

Criterion C2 (revised):

[4] The arguments of a verbal predicate seem to be repeatable when they are in conjunction. For example, in 'Sarah has read this book and that book,' the verb 'read' seems to have two objects; hence, the dependency type 'obj' seems to be repeated. We will return to this issue with respect to the dependency type 'conj' in Section 5.3.

In the syntactic phrase L1=synt=L2, the syntactic dependency type must be clarified so that the type determines the prototypical dependent.

2.4.4 Significance of Mel'čuk's Criteria

These criteria can be used to create a list of SSyntRels for a particular language with reference to how each type is related to DSyntRels (e.g., SSyntRels for English in Mel'čuk & Pertsov 1987, p.85-156, Mel'čuk 2009, p.52-58, and Mel'čuk 2011, p.13-15).

In addition, these criteria can be used to provide other dependency-oriented syntactic frameworks with a theoretical basis created by Tesnière (1959) and later developed by Mel'čuk (1988, 2003, 2004, 2009 and 2011). For example, Stanford Dependencies (de Marneffe & Manning 2008, 2012, 2013), the framework I will use in this dissertation, make only a brief reference to the tradition of dependency grammar. However, it is possible to support de Marneffe & Manning's (2012) dependency framework with a theoretical backbone established in the works of Tesnière and Mel'čuk. Chapter 5 specifically addresses this issue through a detailed discussion of the dependency types presented in Stanford Dependencies.

As a means to define language-specific dependency types, these criteria can be employed to account for the dependency types of natural languages other than English, for example, Japanese language. Chapter 6 deals with the typed-dependency trees for Japanese language, and in Section 6.5, each dependency type in Japanese is defined with reference to these criteria for SSynt-Rel.

2.5 Dependency Grammar and Phrase-structure Grammar

The previous section discussed the basic idea of dependency grammar and its development.

Another important topic to address is the relationship between dependency grammar and other formal approaches to grammar, such as phrase-structure grammar. Debusmann and Kuhlmann (2007, p.1) argue that researchers who work with dependency-oriented grammar formalisms have not yet made clear how dependency approaches relate to phrase-structure frameworks. This section deals with this specific issue.

2.5.1 Comparison of two formalisms

In order to understand the difference between these two approaches, the syntactic structures of the same sentence are shown under phrase-structure grammar (Figure 2.3) and dependency grammar (Figure 2.4) below.

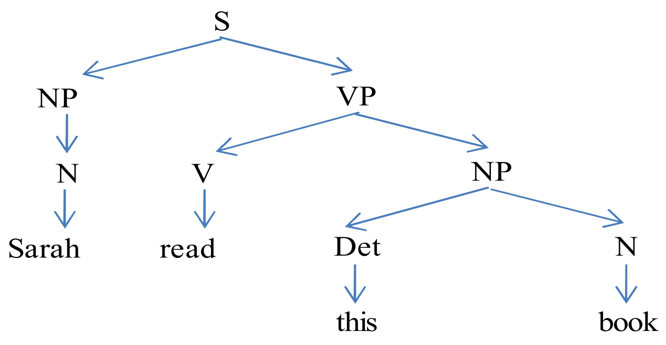


Figure 3.3. The phrase-structure tree for ‘Sarah read this book.’

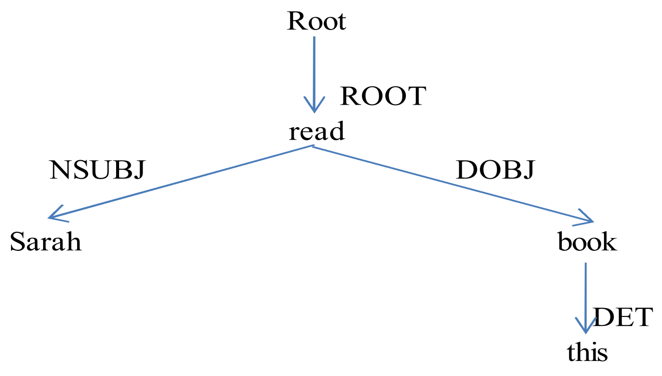


Figure 3.4. The dependency tree for 'Sarah read this book.'

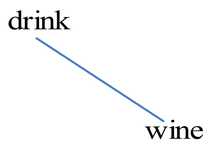
The chief difference between phrase-structure grammar and dependency grammar is that the former contains non-terminal nodes (NP, VP, etc.) while the latter does not. In this way, dependency grammar is simpler than phrase-structure grammar, and therefore easier to handle (Horáček, Zámečníková, & Burgetová 2011).

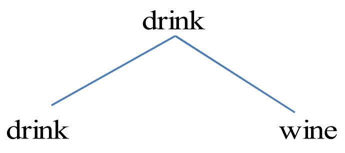
2.5.2 Translation of phrase structure into dependency structure

This section addresses the issue of translating phrase structure into dependency structure. It is possible to argue that the difference between phrase structure and dependency structure is not an essential one, and that the two frameworks are simply variations of the same representation. If this claim is true, then dependency structure can serve as a viable alternative to phrase-structure representations. This section presents a discussion that supports this claim by showing that the phrase structure of a sentence can be translated into a dependency structure, without losing any information contained in the original phrase structure. Other researchers who have attempted this translation have hit various roadblocks in their analyses. For example, Osborne et al. (2011) explore the possibility of translation across phrase structure and dependency structure, concluding that they are "not merely notational variants" (Osborne et al. 2011, p.325) because exocentric structures can be properly represented in phrase structure, but not in dependency structure[5]. Another well-known difference between phrase-structure representation and dependency-structure representation is that the former cannot properly account for *non-projective* structures while the latter can (McDonald, Pereira, Ribarov, & Hajič 2005). This section reviews the issues related to translating phrase structure into dependency structure (and vice versa), paying careful attention to the account of endo- and exocentricity and flatness of structure provided in Osborne et al. (2011), as well as to the production of dependency trees from phrase structure parses provided in de Marneffe, MacCartney & Manning (2006).

Osborne et al. (2011, p.322) summarizes the difference between dependency and constituency by presenting the following structures as examples. Both (2.6a) and (2.6b) represent the structure for the phrase 'drink wine.'

(2.6)

a. 

b. 

In (2.6b), the category label for each projection is replaced by the word. This structure is an example of *Bare Phrase Structure* (BPS) discussed in Chomsky (1995). The authors claim that the difference between these two structures is not substantial; however, they do not go so far as

[5] Exocentricity or endocentricity of a syntactic structure depends on the theoretical framework, and it is not possible to attempt a comprehensive comparison of all of them in this section. In this dissertation, I focus on Osborne et al.'s (2011) framework and de Marneffe et al.'s (2006) dependency-tree framework. For purely exocentric syntactic structure, see Bresnan (2001) and Dalrymple (2001) on Warlpiri and Walsh syntactic structures.

to discard BPS as redundant.

Osborne et al. (2011, p.325) also argue that an exocentric phrase structure cannot be translated to a corresponding dependency structure. They provide the following phrase structure, shown below in Figure 2.5, as an example (Osborne et al. 2011, p.325).

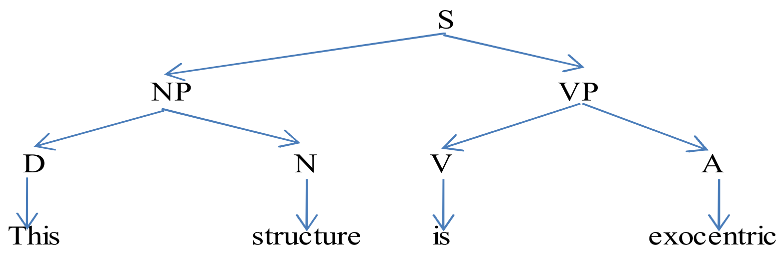


Figure 2.5. The phrase structure for ‘This structure is exocentric.’

They argue that this structure is exocentric because the category S is distinct from the categories N(P) and V(P), and that this exocentric structure cannot be represented in dependency structure (they do not provide us with a BPS structure of the same sentence). However, the same sentence can be represented in terms of the dependencies among words, as shown in Figure 2.6 below (the style of this dependency tree follows de Marneffe & Manning (2012)).

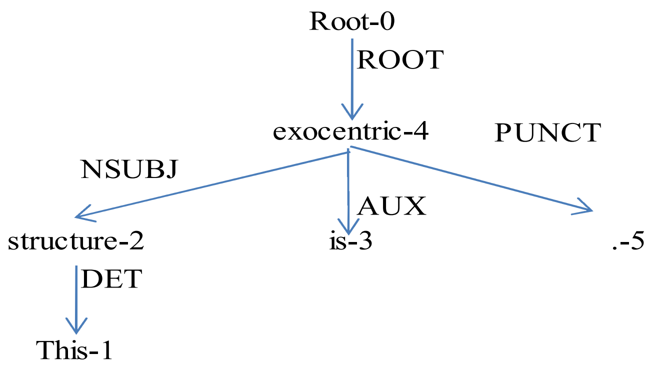


Figure 2.6. The dependency structure for ‘This structure is exocentric.’

In the dependency structure above, the root word (or, the presumed node Root in Stanford Dependencies), not the category S, is at the top of the structure. This makes the copula ‘is’ a dependent on the adjective rather than the root of the sentence. Note that this analysis follows Mel’čuk’s Criterion B1, which states that the head determines how it can be subordinated to another element. Adjectives can appear on their own in an utterance, especially in exclamations (e.g., ‘Beautiful!’), while copulas cannot. The adjective ‘exocentric’ also can appear on its own within a certain context; for example, ‘Sarah, is this structure exocentric, or endocentric?’ ‘Exocentric.’ Therefore, in the example sentence above, the adjective ‘exocentric,’ not the copula ‘is,’ functions as the root of the sentence.

In terms of endocentric structure, Osborne et al. (2011, p.325) argue that endocentric phrase structure can be translated into a corresponding dependency structure. They provide an example of the endocentric phrase structure in the style of BPS, in which the category label for each projection is replaced by the word, and a second example of the dependency structure. In this case, the translation across the two frameworks appears straightforward. Their examples are shown below in Figures 2.7 and 2.8.

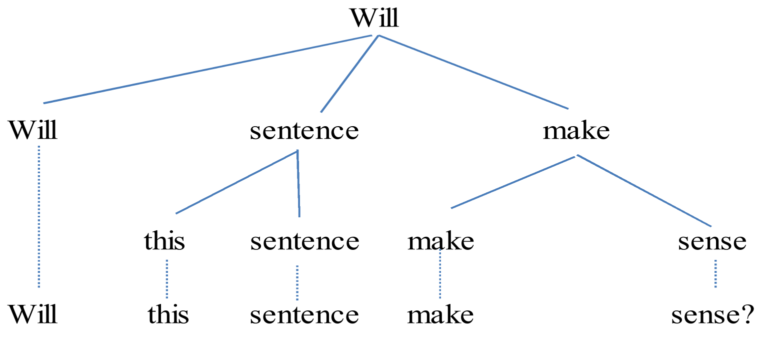


Figure 2.7. The phrase structure for ‘Will this sentence make sense?’

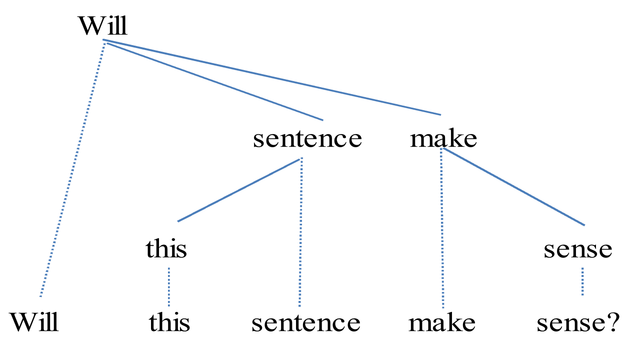


Figure 2.8. The dependency structure for ‘Will this sentence make sense?’

In terms of the flatness of structure, Osborne et al. (2011, p.326) discuss the issue of movement in the tradition of Government and Binding and the Minimalist Program (GB/MP). The phrase structure in that tradition for the same sentence is shown below in Figure 2.9, in which there are traces of the auxiliary.

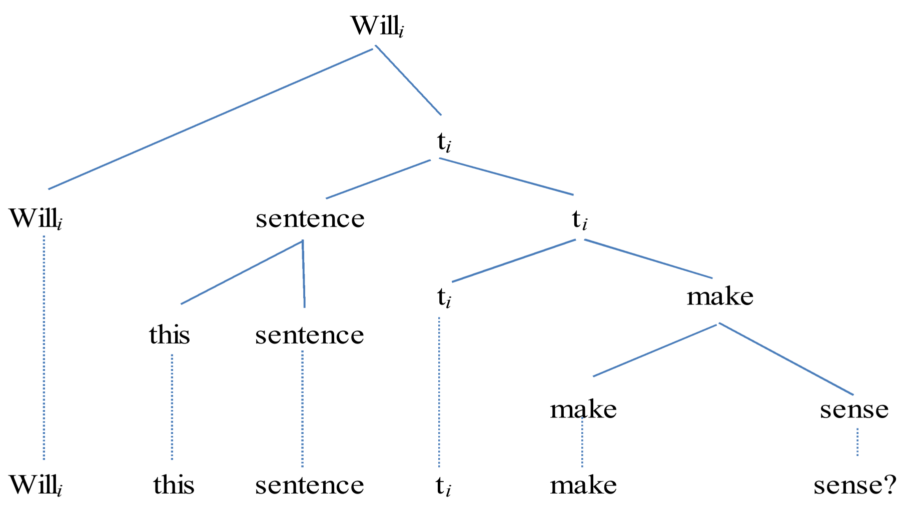


Figure 2.9. The phrase structure for ‘Will this sentence make sense?’ with traces of the moved

auxiliary.

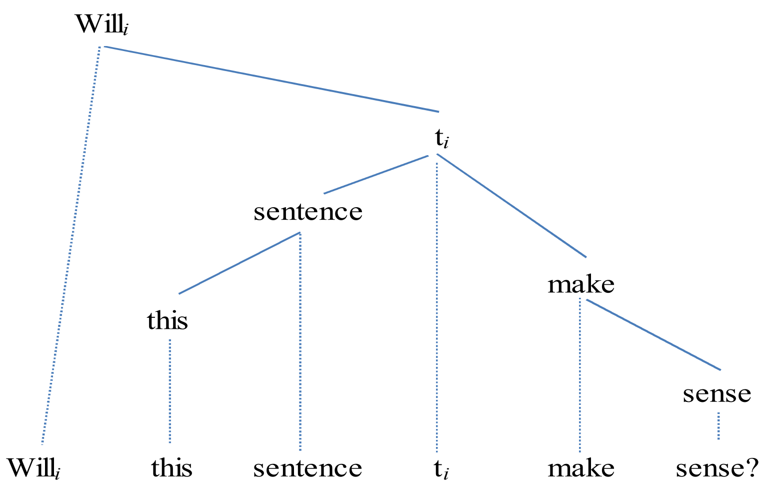


Figure 2.10. The dependency structure for ‘Will this sentence make sense?’ with traces of the moved auxiliary.

Osborne et al. (2011, p.327) admit that the dependency structure above is problematic because it contains the traces of the moved auxiliary, which is uncommon in dependency formalisms. A better dependency structure would show that the root of the sentence is the verb and the auxiliary is its dependent. The Stanford-Dependencies model would generate the structure in this way, as shown in Figure 2.11.

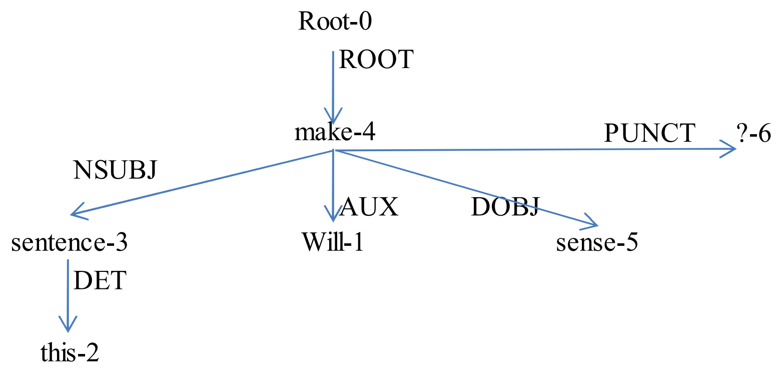


Figure 2.11. The dependency structure for ‘Will this sentence make sense?’ in Stanford-Dependencies format.

Note that this analysis follows Mel’čuk’s Criterion B1, which states that the head determines how it can be subordinated to another element. Verbs can appear on their own in an utterance, especially in the imperative mood (e.g., ‘Run!’), while auxiliaries cannot. Therefore, in the sentence shown in Figure 2.11 above, the verb ‘make,’ not the auxiliary ‘will,’ should function as the root of the sentence.

2.6 Summary

This chapter introduced the concept of dependency grammar through a discussion of Tesnière’s (1959) seminal assumption about dependency along with more recent theories of dependency grammar proposed by I. Mel’čuk and his colleagues (Iordanskaja & Mel’čuk 2000: Mel’čuk & Pertsov 1987; Mel’čuk 1988, 2003, 2009, 2011). This chapter also addressed the difference between dependency grammar and phrase-structure grammar. Section 2.2 presented an overview of dependency grammar and Section 2.3 focused specifically on Tesnière’s (1959) seminal work on dependency grammar. Section 2.4 discussed Mel’čuk’s work on Deep Syntactic Relations and Surface Syntactic Relations as a development of Tesnière’s (1959) concept of dependency. Finally, the difference between dependency and phrase-structure grammar was briefly discussed in Section 2.5 with reference to Osborne et al. (2011).

3. Typed-Dependency Trees and Lexical-Functional Grammar[6]

3.1 Introduction

The previous chapter reviewed the basic ideas of dependency grammar, and examined the similarities and differences between dependency-based and phrase-structure-based representations of syntactic structure. This chapter continues to explore the equivalence between a typed-dependency tree for a sentence and its functional-structure representation according to Lexical-Functional Grammar (LFG) (Bresnan 1978; Bresnan 1982; Kaplan & Bresnan 1982). Both LFG and dependency grammar theory assume that the individual pieces of lexical information of a sentence are integrated into the representation for the whole sentence, and both frameworks make explicit the process through which these pieces of lexical information are integrated. Dependency grammar employs one level of representation (i.e., a dependency tree for a sentence), while LFG employs multiple levels of representation (i.e., constituent structure, functional structure, argument structure, etc.). The idea of structural correspondence in LFG can be considered as an extension of typed-dependency tree representation of grammatical knowledge. In other words, LFG represents one direction of development of the dependency grammar tradition started by Tesnière (1959).

This chapter is organized as follows. First, the basic architecture of the LFG framework is briefly summarized in Section 3.2. Next, Section 3.3 shows the equivalence between a functional-structure representation for a sentence and its typed-dependency tree. This equivalence supports the idea that LFG is one direction of development of the dependency grammar. This section also introduces the revision of Mel'čuk's Criterion A for the existence of dependency relationship between two words, in terms of their possibility to constitute a fragment

[6] This chapter is based on Oya (2013c).

functional structure.

3.2 Lexical-Functional Grammar (LFG)

This section is a brief introduction of the basic framework of LFG. This system of grammatical representation was first proposed by Bresnan (1978), and has since been developed by a number of linguists and incorporated in various fields of research (e.g., Bresnan 1982, 2001; Butt, King, Niño & Segond 1999; Dalrymple 2001; Kaplan & Bresnan 1982). The LFG framework proposes different levels of representation for grammatical knowledge about a sentence, and the pieces of information represented at each of these different levels correspond to each other through functional descriptions of the phrase-structure tree for the sentence. The three levels of representation in the LFG framework are constituent structure (c-structure), functional structure (f-structure), and argument structure (a-structure).

3.2.1 Structural correspondence[7]

Describing the syntactic properties of natural languages only at the language-specific tree level can lead us to postulate operations that are linguistically unmotivated, such as movement in the tradition of Government and Binding (Chomsky 1981). In contrast, LFG can capture invariant properties of grammatical knowledge across languages at the functional level of representation, which is called *functional structure* or *f-structure*. The functional structure for a grammatical sentence is constructed in a step-by-step manner by integrating the lexical information of each word in the sentence. The functional structure for a grammatical sentence observes

[7] This subsection is based on Kaplan and Bresnan (1982), and Oya (2013c).

well-formedness constraints (see Section 3.2.2). Various linguistic phenomena, such as long-distance dependency, control, and anaphora, can be represented at f-structure.

The f-structure for a sentence corresponds to its c-structure, and vice versa. C-structure is a phrase-structure tree that is specified by the phrase structure rules (PS rules) of a context-free grammar. Each node of a phrase-structure tree is annotated with *functional equations* and corresponds to the f-structure. Functional equations specify the correspondence between f-structures and the nodes, and the top node S corresponds to the f-structure for the sentence as a whole.

The PS rule in (3.1) states that the syntactic category S (=sentence) is expanded into NP (=noun phrase) and VP (verbal phrase). The up arrows in the functional equations refer to the functional structures which correspond to the nodes that immediately dominate the annotated nodes. The functional equation annotated below NP states that the value of the SUBJ (=subject) feature of the f-structure corresponding to S is the f-structure corresponding to NP. The equation annotated below VP states that the f-structure corresponding to S equals the f-structure corresponding to VP.

(3.1)

$$\begin{array}{lll} S \rightarrow & NP & VP \\ & (\uparrow \text{SUBJ})=\downarrow & \uparrow=\downarrow \end{array}$$

The up arrows and the down arrows are instantiated by variables which stand for an underspecified functional structure. In the PS rule below, the up arrows are instantiated by *f1*, which is the functional structure corresponding to the node S. The down arrow in the functional equation (↑SUBJ)=↓ annotated to the NP is instantiated by *f2*, which is the functional structure corresponding to the node NP. The down arrow in the functional equation ↑=↓ is instantiated by *f3*, which is the functional structure corresponding to the node VP.

(3.2)

S → NP VP

(*f1*SUBJ)=*f2* *f1*= *f3*

The instantiated functional equation (*f1*SUBJ)=*f2* states that the functional structure *f1* has an attribute SUBJ whose value is a functional structure *f2*. The instantiated functional equation *f1*= *f3* states that the functional structure *f1* equals the functional structure *f3*, thus they *merge*[8] with each other. The PS rule (3.2) represents the correspondence between the phrase structure and the functional structure, as illustrated below

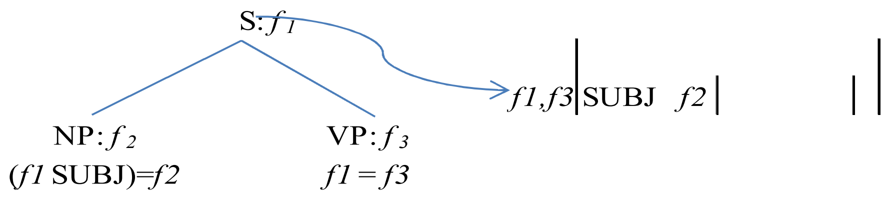


Figure 3.1 The correspondence between the phrase structure and the functional structure represented by the PS rule (3.2)

The PS rule in (3.3) states that the syntactic category VP is divided into V and NP. The

[8] Bresnan and Kaplan (1982, p.191) use the term *Merge* which "checks to see whether those values are the same and hence already satisfy the equality relation", prior to Chomsky (1999).

equation below V states that the f-structure corresponding to V is equal to the one corresponding to VP. The equation below NP states that the value of the OBJ feature of the f-structure corresponding to VP is the f-structure corresponding to NP.

(3.3)

$$\begin{array}{lll} VP \rightarrow & V & NP \\ & \uparrow=\downarrow & (\uparrow \mathrm{OBJ})=\downarrow \end{array}$$

In the instantiated PS rule below, the up arrows are instantiated by *f3* which is the functional structure corresponding to the node VP. The down arrow in the functional equation ↑=↓ annotated to V is instantiated by *f4* which is the functional structure corresponding to the node V. The down arrow in the functional equation (↑OBJ)=↓ is instantiated by *f5* which is the functional structure corresponding to the node NP.

(3.4)

$$\begin{array}{lll} VP \rightarrow & V & NP \\ & f3=f4 & (f3\mathrm{OBJ})=f5 \end{array}$$

The instantiated functional equation *f3*=*f4* states that the functional structure *f3* equals the functional structure *f4*, thus they merge with each other. The instantiated functional equation (*f3*OBJ)= *f5* states that the functional structure *f3* has an attribute OBJ whose value is a functional structure *f5*. The PS rule (3.4) thus represents the correspondence between the phrase structure and the functional structure, as illustrated below

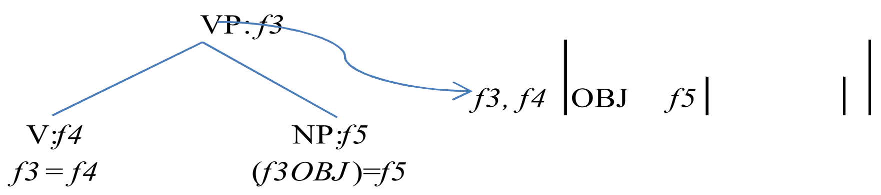


Figure 3.2. The correspondence between the phrase structure and the functional structure represented by the PS rule (3.4)

The PS rule in (3.5) states that the syntactic category NP is divided into DET (=determiner) and N (=noun). The DET and the functional equation are parenthesized because they are optional. The functional equation below DET states that the value of the DET feature of the f-structure corresponding to NP is the f-structure corresponding to DET. The functional equation below N states that the f-structure corresponding to N is equal to the one corresponding to NP.

(3.5)

$$\mathrm{NP} \rightarrow \left(\begin{array}{c} \mathrm{DET} \\ (\uparrow\mathrm{DET})=\downarrow \end{array}\right) \begin{array}{c} \mathrm{N} \\ \uparrow=\downarrow \end{array}$$

In the instantiated PS rule below, the up arrows are instantiated by *f2* which is the functional structure corresponding to the node NP. The down arrow in the functional equation (↑DET)=↓ is instantiated by *f6* which is the functional structure corresponding to the node DET. The down arrow in the functional equation ↑=↓ annotated to N is instantiated by *f7*, which is the functional structure corresponding to the node N.

(3.6)

$$NP \rightarrow \left(\begin{array}{c} DET \\ (f2DET)=f6 \end{array} \right) \begin{array}{c} N \\ f2=f7 \end{array}$$

The instantiated functional equation (*f2*DET)= *f6* states that the functional structure *f2* can have an attribute DET whose value is a functional structure *f6*. The instantiated functional equation *f2*=*f7* states that the functional structure *f2* equals the functional structure *f7*, thus they merge with each other. The PS rule (3.6) thus represents the correspondence between the phrase structure and the functional structure, as illustrated below

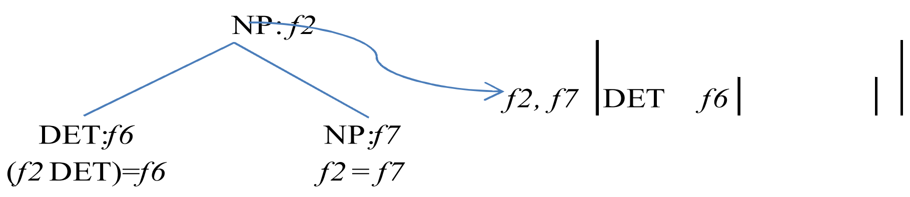


Figure 3.3. The correspondence between the phrase structure and the functional structure represented by the PS rule (3.6)

Thus, all the functional structures above are integrated into one functional structure corresponding to the root node S, as illustrated below.

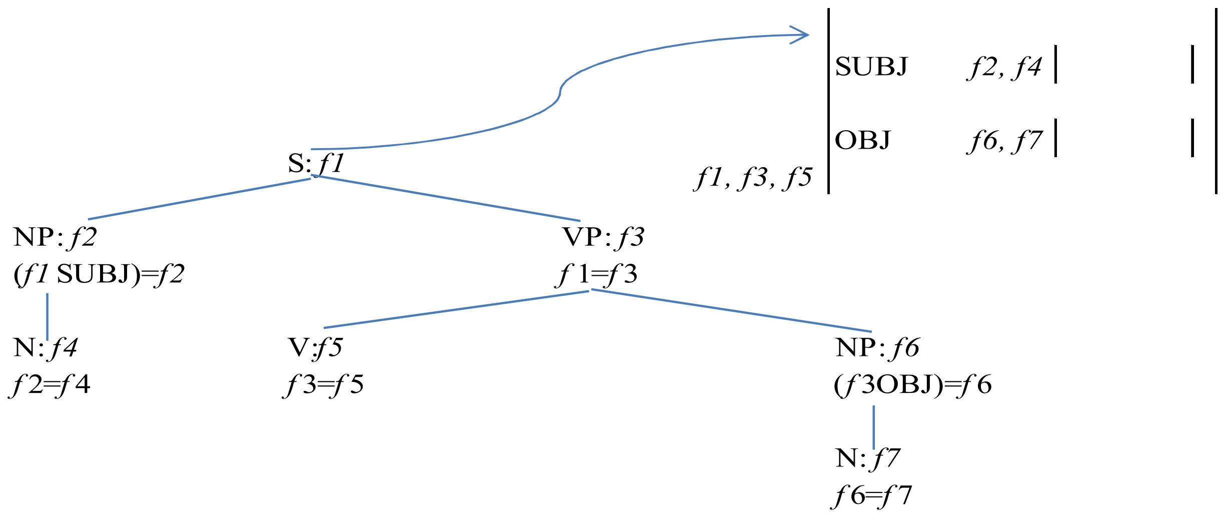


Figure 3.4. The correspondence between the phrase structure and the functional structure represented by the PS rules (3.2), (3.4) and (3.6)

The lexical information for each word in a sentence is integrated into a single f-structure that corresponds to the S node of the phrase-structure tree. For example, the pieces of lexical information in the sentence 'John studies languages,' which are shown in (3.7), (3.8), and (3.9) below, are integrated into one f-structure at the S node of the phrase-structure tree. These pieces of information are called *functional equations*, and they are *attribute-value pairs*. For example, the first functional equation of the word 'John' is (↑PRED)='John,' which shows that this word has an attribute PRED whose value is 'John.' The second functional equation (↑NUMBER) = Singular shows that the word 'John' has an attribute NUMBER whose value is 'Singular.' The lexical information for the words 'studies' and 'languages' are shown in (3.8) and (3.9), respectively. The equations (↑SUBJ NUMBER) $=_C$ SINGULAR and (↑SUBJ PERSON) $=_C$ 3RD are *constraining equations*; unlike ordinary equations, they do not define the value of the designated attribute. Rather, they give a constraint on the value of the designated attribute in the functional structure. If the attribute does not have the value as indicated in a constraining equation, then the functional structure is not well-formed. For example, the equation (↑SUBJ NUMBER) $=_C$ SINGULAR states that the NUMBER attribute in the SUBJ must have the value SINGULAR.

(3.7)

John, N:

(↑PRED)= 'John'

(↑NUMBER) = Singular

(↑PERSON) = 3^{RD}

(3.8)

studies, V:

(↑PRED) = 'study <SUBJ, OBJ>'

(↑SUBJ NUMBER) $=_C$ SINGULAR

(↑SUBJ PERSON) $=_C$ 3^{RD}

(↑TENSE) = PRESENT

(3.9)

languages, N:

(↑PRED) = 'language'

(↑PERSON) = 3RD

(↑NUMBER) = PLURAL

Figures 3.5 and 3.6 illustrate the constituent and function structures for the sentence 'John studies languages,' respectively. The subscript on each node of the tree in Figure 3.5 represents a functional variable that refers to the f-structure corresponding to the node. The up and down arrows on the functional equations are replaced by these functional variables, so that the f-structures are integrated through equations, and we obtain the f-structure for the whole sentence corresponding to the root S, as shown in Figure 3.6.

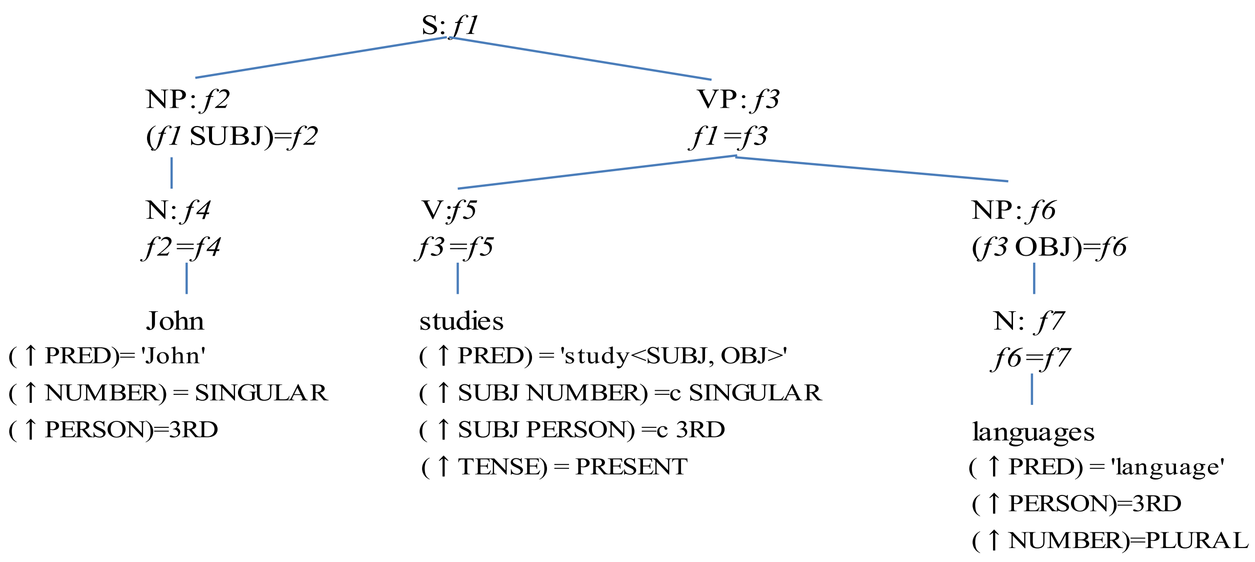


Figure 3.5. The constituent structure for 'John studies languages' with the lexical information for each word

$$
{}_{f1,\,f3,\,f5}\left|\begin{array}{ll}
\text{SUBJ} & {}_{f2,\,f4}\left|\begin{array}{ll}\text{PRED} & \text{'John'}\\ \text{NUMBER} & \text{SINGULAR}\\ \text{PERSON} & \text{3rd}\end{array}\right| \\
\text{OBJ} & {}_{f6,\,f7}\left|\begin{array}{ll}\text{PRED} & \text{'languages'}\\ \text{NUMBER} & \text{PLURAL}\\ \text{PERSON} & \text{3rd}\end{array}\right| \\
\text{PRED} & \text{'study<SUBJ, OBJ>'} \\
\text{TENSE} & \text{PRESENT}
\end{array}\right|
$$

Figure 3.6. The functional structure for ‘John studies languages.’

3.2.2 Well-formedness constraints

According to LFG, an f-structure for a sentence must observe the following well-formedness constraints: completeness, coherence, and consistency. Each constraint is discussed in turn below.

The completeness constraint states that a predicate and all its arguments must be present in an f-structure (Kaplan & Bresnan 1982, p.211-212). The sentence presented in (3.10) is ungrammatical because the f-structure lacks the object required for the predicate ‘read.’ In other words, the f-structure, shown in Figure 3.7, does not observe completeness constraint.

(3.10) Sarah reads.

```
| PRED    'read<SUBJ, OBJ>'                  |
| SUBJ    | PRED      'Sarah'      |         |
|         | PERSON    3RD          |         |
|         | NUMBER    SINGULAR     |         |
|         | GENDER    FEMININE     |         |
| OBJ     ?                                  |
| TENSE   PRESENT                            |
| PUNCT   | FORM      '.'          |         |
|         | STMT-TYPI DECLARATIVE  |         |
```

Figure 3.7. The f-structure for ‘Sarah reads.’

The completeness constraint addresses some issues not covered by Mel’čuk’s criteria for syntactic relations (SSyntRel, discussed in Section 2.4.3). The completeness constraint does not have any equivalent in Mel’čuk’s criteria for surface syntactic relations. This is the case because Criterion A is concerned with whether two given words *actually present* in a sentence have a dependency relationship. Criterion A is not concerned with the lack of a dependent word (or phrase) that might be required by a head word, as exemplified above in sentence (3.10). In addition, Criterion B is concerned with the direction of existing dependency relations, and Criterion C simply determines the type of these dependency relations. Thus, Criteria B and C are not relevant measures to identify whether a head word lacks a dependent. In this way, the completeness constraint is an extension of Mel'čuk's criteria for syntactic relations (SSyntRel).

The coherence constraint states that all arguments in an f-structure must be required by the predicate (Kaplan & Bresnan 1982, p.212). The following sentence (3.11) is ungrammatical because the f-structure contains an argument that is not required by the predicate 'fell.' Put another way, the f-structure, as shown in the figure below, does not observe coherence.

(3.11) *Sarah fell the book.

```
|PRED    'fall<SUBJ>'                           |
|SUBJ    |PRED     'Sarah'       |              |
|        |PERSON   3RD           |              |
|        |NUMBER   SINGULAR      |              |
|        |GENDER   FEMININE      |              |
|                                               |
|OBJ     |PRED     'book'              |        |
|        |PERSON   3RD                 |        |
|        |NUMBER   SINGULAR            |        |
|        |DET      |FORM  THE  |       |        |
|        |         |TYPE  DET  |       |        |
|                                               |
|TENSE   PAST                                   |
|PUNCT   |FORM      '.'          |              |
|        |STMT-TYPI DECLARATIVE  |              |
```

Figure 3.8. The f-structure for 'Sarah fell the book.'

Finally, the consistency constraint states that every attribute in an f-structure must have a unique value (Kaplan & Bresnan 1982, p.181). In other words, there must be no inconsistency between an attribute and its value. The following sentence (3.12) is ungrammatical because the number of the determiner does not agree with the noun. The lexical information for the determiner 'these' is presented in (3.13). This determiner requires that the noun it modifies be plural. However, the noun after 'these' in (3.12) does not satisfy this requirement; hence, this sentence violates the consistency constraint. The f-structure for this sentence is shown in the figure below.

(3.12) *Sarah has read these book.

(3.13)

these, determiner:

(↑NUMBER) $=_C$ PLURAL

(↑FORM) = 'these'

(↑TYPE)=DEMONSTRATIVE

```
|PRED    'read<SUBJ, OBJ>'                              |
|SUBJ   |PRED      'Sarah'    |                         |
|       |PERSON    3RD        |                         |
|       |NUMBER    SINGULAR   |                         |
|       |GENDER    FEMININE   |                         |
|                                                       |
|OBJ    |PRED      'book'                        |      |
|       |PERSON    3RD                           |      |
|       |NUMBER    SINGULAR                      |      |
|       |NUMBER    PLURAL?                       |      |
|       |DET      |FORM  THESE                |  |      |
|       |         |TYPE  DEMONSTRATIVE        |  |      |
|                                                       |
|AUX    |TENSE     PRESENT    |                         |
|       |ASPECT    PERFECT    |                         |
|                                                       |
|PUNCT  |FORM      '.'        |                         |
|       |STMT-TYPE DECLARATIVE|                         |
```

Figure 3.9. The f-structure for '*Sarah has read these book.'

3.3 Equivalence of the Functional-Structure Representation and the Typed-Dependency Tree Representation for a Sentence[9]

This section proposes the idea of equivalence between the functional-structure representation for a sentence and the typed-dependency tree representation for the same sentence. This idea is related to the basic assumption of unification grammar (Sag, Kaplan, Karttunen, Kay, Pollard, Shieber & Zaenen 1986) that the lexical information contained in each word in a sentence is integrated into one complete and coherent level of syntactic representation for the sentence. This integration is processed according to the dependency relationship between the words in a sentence (See Section 3.2.1 for structural correspondence between the constituent structure and the functional structure for a sentence). Therefore, the lexical information of each word in a

[9] This section is based on Oya (2013c).

sentence can be integrated into one single representation only if it is clearly defined which word is related to, or dependent on, which word in the sentence. There must be well-defined criteria for rejecting incorrect dependencies between words in a sentence; without such criteria, incorrect connections between words in a sentence would yield an incorrect representation.

Mel'čuk's Criterion A (see Section 2.4.3) can work for the purpose mentioned above; however, his version of Criterion A employs the notion of *prosodic unit*. This notion, however, is too vague to be a criterion for identifying the dependency relationship between two words in a sentence. To solve the problem of dependency-relationship identification, Gerdes & Kahane (2011) introduced the notion of "acceptable fragments of an utterance." They argue that acceptable fragments are the building blocks of their dependency grammar. In this section, we attempt to define "acceptable fragments of an utterance" in terms of functional structure in the LFG framework which is the building block of our dependency grammar.

3.3.1 Overview

Consider the sentence presented in (3.13). The functional structure for the sentence is shown in Figure 3.10, and the corresponding typed-dependency tree is shown in Figure 3.11[10].

(3.13) David has written this article.

[10] In this study, the form of the verbal predicate in a functional structure is in its dictionary form, following the convention of standard Lexical-Functional Grammar, while its form in the corresponding typed-dependency tree is as appeared in the sentence.

```
|          |PRED     'write<SUBJ, OBJ>'                          |  |
|          |SUBJ    |PRED      'David'             |             |  |
|          |        |PERSON    3RD                 |             |  |
|          |        |NUMBER    SINGULAR            |             |  |
|          |        |GENDER    MASCULINE           |             |  |
|          |                                                     |  |
|          |OBJ     |PRED      'article'               |         |  |
|          |        |PERSON    3RD                     |         |  |
|          |        |NUMBER    SINGULAR                |         |  |
|          |        |DET      |FORM 'this'          ||         |  |
|          |        |         |TYPE DEMONSTRATIVE   ||         |  |
|          |                                                     |  |
|          |AUX     |FORM      'has'               |             |  |
|          |        |TENSE     PRESENT             |             |  |
|          |        |ASPECT    PERFECT             |             |  |
|          |                                                     |  |
|          |PUNCT   |FORM      '.'                 |             |  |
|ROOT      |        |STMT-TYPE DECLARATIVE         |             |  |
```

Figure 3.10. The functional structure for the sentence (3.13)

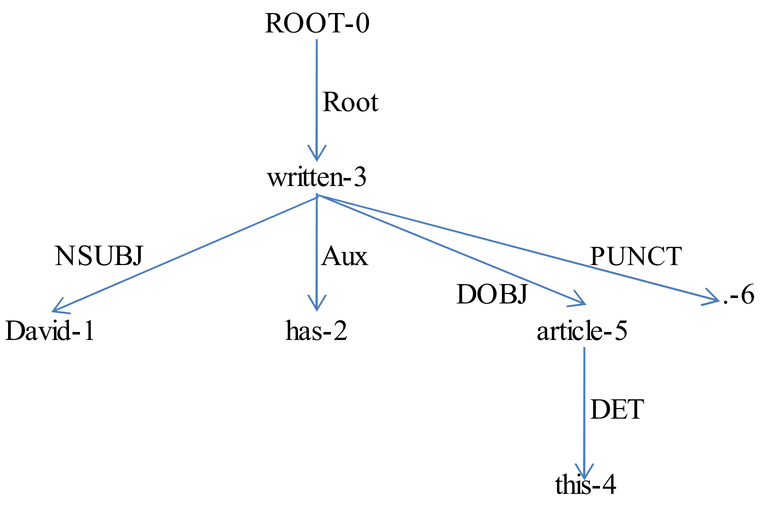


Figure 3.11. The typed-dependency tree for the sentence (3.13)

For two words in a dependency relationship, the lexical information for the tail word is unified with the information for its head word. This process goes backward along the typed-dependency arc of the tree. As a result of this unification process, the information for the whole sentence is at the ROOT level. In the example above, the information for the word ‘this’ is unified with the information for the word ‘article,’ whose information is unified with the word ‘written.’ The information for ‘David’ and ‘has’ is also unified with the information for the word ‘written.’ At the ROOT level, we have the information for the whole sentence ‘David has written this article.’

3.3.2 Fragment functional structure and dependency

The unification process of lexical information along the typed-dependency tree that was described briefly in the previous section needs further explanation. To do this, let us look at the minimum dependency relationship between two words. The dependency relationship between the head and tail of a typed-dependency tree is interpreted as a *fragment functional structure* (Oya 2013c, p.24), and can be schematized as shown in Figure 3.12.

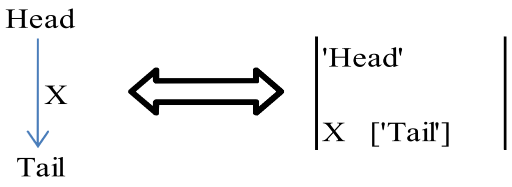


Figure 3.12. Schema for the equivalence between a typed-dependency tree and a functional structure (Oya 2013c, p.24)

Fragment functional structures are constructed from two words, and one of these words functions as an argument or an adjunct (a modifier) for the other word[11]. In Figure 3.12 above, the “X” next to the dependency arc represents the type of the dependency. The schema shows that the head of the typed-dependency tree has an attribute whose name is “X,” and the value of the attribute “X” equals the tail of the same typed-dependency tree. For example, the relation between ‘David’ and ‘written’ in sentence (3.13) is such that the word ‘written’ has a value ‘David’ in terms of the attribute NSUBJ. This relation can be schematized in a fragment functional structure below. In this fragment functional structure, the word ‘David’ functions as an argument for the verb ‘written.’

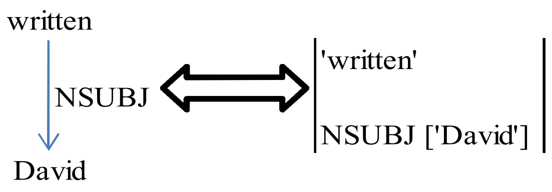


Figure 3.13. Schema for the equivalence between the typed-dependency tree and the fragment functional structure for ‘written’ and ‘David’ in (3.13)

The relation between ‘written’ and ‘has’ is such that the word ‘written’ has a value ‘has’ in terms of the attribute ‘AUX.’ In the fragment functional structure below, the auxiliary ‘has’

[11] The definition of surface syntactic fragments of an utterance by Gerdes & Kahane (2011, p.23) is not the same as that of fragment functional structures. In their definition, if fragments “can stand alone,” they are autonomizable, and if a single word can replace them, then they belong to a distributional class. On the other hand, fragment functional structures are not always autonomizable, because two words which constitute a fragment functional structure do not always stand alone. In addition to this, fragment functional structures do not belong to a distributional class, because they are not always replaced by a single word. The idea of fragment functional structure, though, is based on their assumption that fragments are the building block of dependency grammar.

functions as a modifier for the verb 'written.[12]' Notice here that the dependency relationship between them is not the other way round. That is, 'has' provides 'written' with lexical information about the tense and the aspect of the clause, and the number and person of the subject (3rd person and singular).

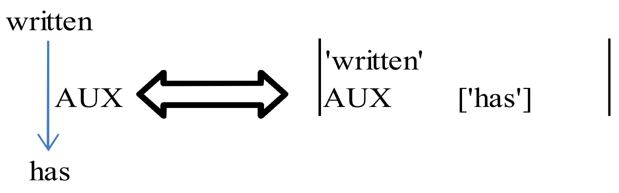


Figure 3.14. Schema for the equivalence between the typed-dependency tree and the fragment functional structure for 'written' and 'has' in (3.13)

The relation between 'written' and 'article' is such that the word 'written' has a value 'article' in terms of the attribute OBJ. In the fragment functional structure below, the noun 'article' functions as one of the arguments of the verb 'written.'

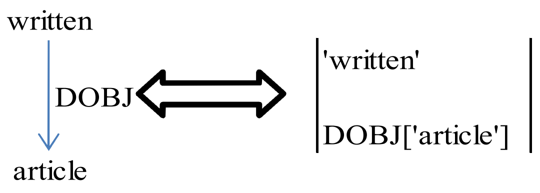


Figure 3.15. Schema for the equivalence between the typed-dependency tree and the fragment functional structure for 'written' and 'article' in (3.13)

[12] This analysis does not follow the standard LFG analysis on auxiliaries with reference to the notion of *f-structure co-head* (Falk 1984, Grimshaw 1991). This is due to the definition of the dependency type AUX in Stanford Dependency that, for a dependency between a verb and an auxiliary, the verb functions as the head and the auxiliary functions as the tail. See Section 5.3.2 for the definition of the dependency type AUX.

The relation between ‘article’ and ‘this’ is such that the word ‘article’ has a value ‘this’ in terms of the attribute DET. In the fragment functional structure below, ‘this’ functions as a modifier for ‘article.’

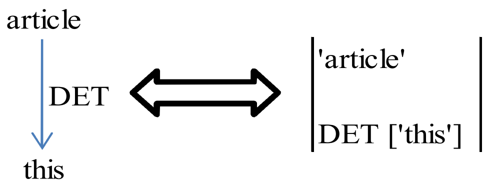


Figure 3.16. Schema for the equivalence between the typed-dependency tree and the fragment functional structure for ‘article’ and ‘this’ in (3.13)

The relation between ‘written’ and ‘.’ is such that the word ‘written’ has a value ‘.’ in terms of the attribute PUNCT. In the fragment functional structure below, ‘.’ functions as a modifier for ‘written.’

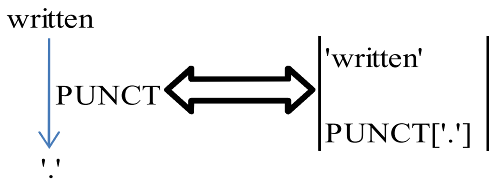


Figure 3.17. Schema for the equivalence between the typed-dependency tree and the fragment functional structure for ‘written’ and ‘.’ in (3.13)

3.3.3 Integrating fragment functional structures

The lexical information for each word is integrated into the functional-structure representation. For example, the lexical entries for the words in the sentence (3.13) are shown below in (3.14) to (3.19).

(3.14)

write, V

(↑PRED) = ‘write<(↑SUBJ), (↑OBJ)>’

(3.15)

David, N

(↑PRED) = ‘David’

(↑PERSON)=3^{RD}

(↑NUMBER)=SINGULAR

(↑GENDER)=MASCULINE

(3.16)

has, AUX

(↑FORM)= ‘has’

(↑TENSE)=PRESENT

(↑ASPECT)=PERFECT

(↑SUBJECT PERSON)=c 3rd

(↑SUBJECT NUMBER) =c SINGULAR

(3.17)

article, N

(↑PRED) = ‘article’

(↑PERSON)=3^{RD}

(↑NUMBER)=SINGULAR

(3.18)

this, DET

(↑TYPE) = DEMONSTRATIVE

(↑FORM) = ‘this’

(3.19)

. PERIOD

(↑FORM)= ‘.’

(↑STMT-TYPE) = DECLARATIVE

The equivalence schema in Figure 3.12 in Section 3.3.2 can be lexicalized, as shown in Figure 3.18 below. The verb ‘written’ is a transitive verb, which requires that the attributes “nsubj” and “dobj” be specified. The noun ‘David’ provides the verb ‘written’ with the “nsubj” value. The attribute “nsubj” is one of the subcategories of “subj” in Stanford Dependencies (de Marneffe & Manning 2012).

The “obj” attribute in Figure 3.18 is not specified because no element in the dependency specifies this attribute. Therefore, this fragment functional structure does not satisfy the completeness constraint (see Section 3.2.2).

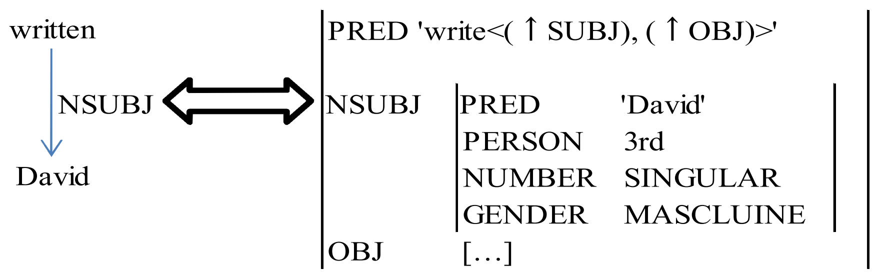


Figure 3.18. Lexicalized schema for the equivalence between the typed-dependency tree and the

fragment functional structure for ‘written’ and ‘David’ in the sentence (3.13)

Notice that a fragment functional structure does not necessarily follow the well-formedness constraints (Section 3.2.2). For example, the functional structure in Figure 3.18 is not well-formed; it is not complete because the “obj” value is not specified. Oya (2013c, p.27) points out that the well-formedness constraints should be applied to the functional structure for a sentence as a whole, not to the fragment functional structures for the words in dependency relationship in the sentence.

The equivalence schema in Figure 3.15 can be lexicalized, as shown in Figure 3.19. The noun ‘article’ provides the verb ‘written’ with the “obj” value, while the “subj” value remains underspecified.

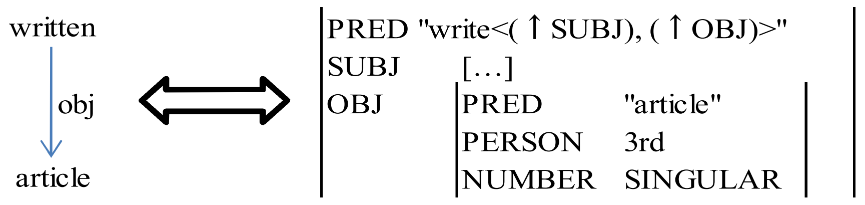


Figure 3.19. Lexicalized schema for the equivalence between the typed-dependency tree and the fragment functional structure for ‘write’ and ‘article’ in the sentence (3.13)

Lexicalized schema in Figure 3.18 for ‘David’ and ‘written,’ and that in Figure 3.19 for ‘written’ and ‘article,’ share the same dependency head; hence, they are integrated to form one lexicalized schema, as shown in the figure below.

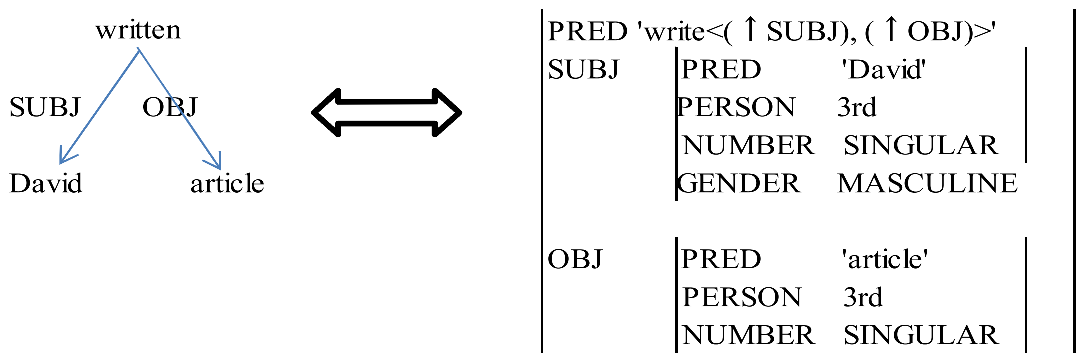


Figure 3.20. Lexicalized schema for the equivalence between the typed-dependency tree and the fragment functional structure for 'David,' 'written' and 'article' in the sentence (3.13)

The equivalence schema in Figure 3.14 for 'written' and 'has' can also be lexicalized, as shown in Figure 3.21. The auxiliary 'has' provides the verb 'written' with the tense and aspect values, and also specifies the person and number values of the subject of the verb.[13]

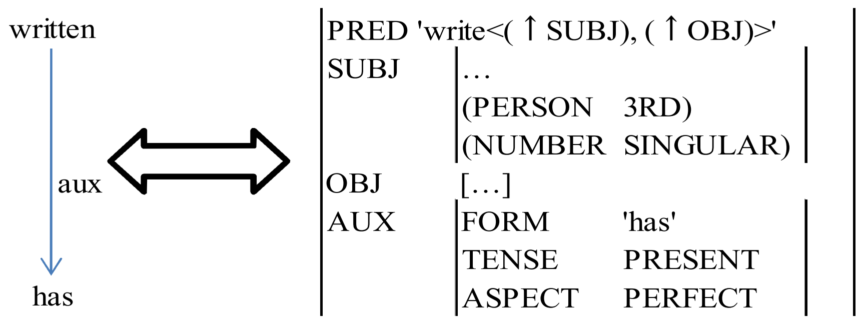


Figure 3.21. Lexicalized schema for the equivalence between the typed-dependency tree and the

[13] In the convention of standard LFG, the verb form at the PRED value must be in its dictionary form; however, this convention does not clearly show the agreement between the auxiliary and the past participle form. This is due to the idea that the agreement relationship should be represented in *morphological structure* (m-structure) (Butt, Niño & Segond 1996, p.117), another attribute-value pair matrix for auxiliaries. M-structure can be similar to Morph-D (See Section 2.4), because both of them are independent from, but related to, the syntactic representation of a sentence. The equivalence between them will be a topic of future research.

f-structure for ‘written’ and ‘has’ in the sentence (3.13)

Lexicalized schema in Figure 3.20 for ‘David,’ ‘written’ and ‘article,’ and that in Figure 3.21 for ‘written’ and ‘has,’ share the same dependency head; hence they are integrated to form one lexicalized schema, as shown in the figure below.

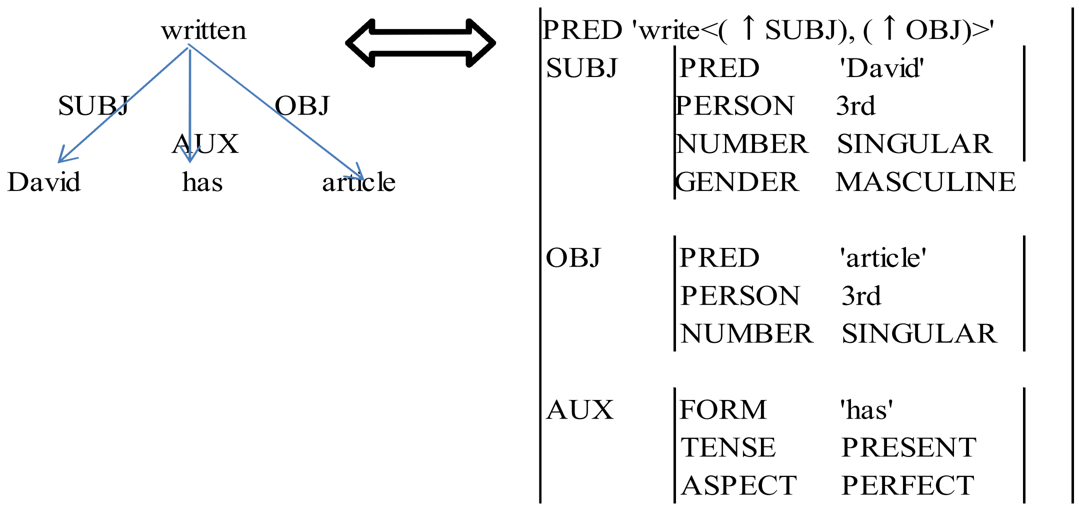


Figure 3.22. Lexicalized schema for the equivalence between the typed-dependency tree and the fragment functional structure for ‘David,’ ‘has,’ ‘written’ and ‘article’ in the sentence (3.13)

The equivalence schema in Figure 3.16 for ‘this’ and ‘article’ can be lexicalized, as shown in Figure 3.23. The determiner ‘this’ provides the noun ‘article’ with the DET value.

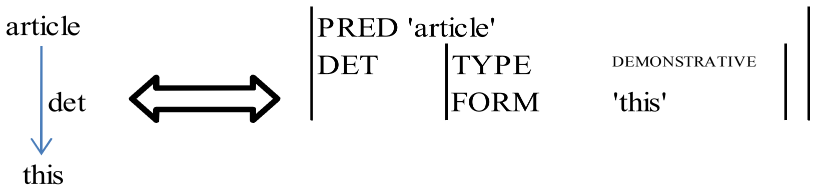


Figure 3.23. Lexicalized schema for the equivalence between the typed-dependency tree and the

f-structure for ‘article’ and ‘this’ in the sentence (3.13)

Lexicalized schema in Figure 3.23 for ‘this’ and ‘article,’ and that in Figure 3.22 for ‘David,’ ‘has,’ ‘written,’ and ‘article,’ share the same word; hence they are integrated to form one lexicalized schema, as shown in the figure below.

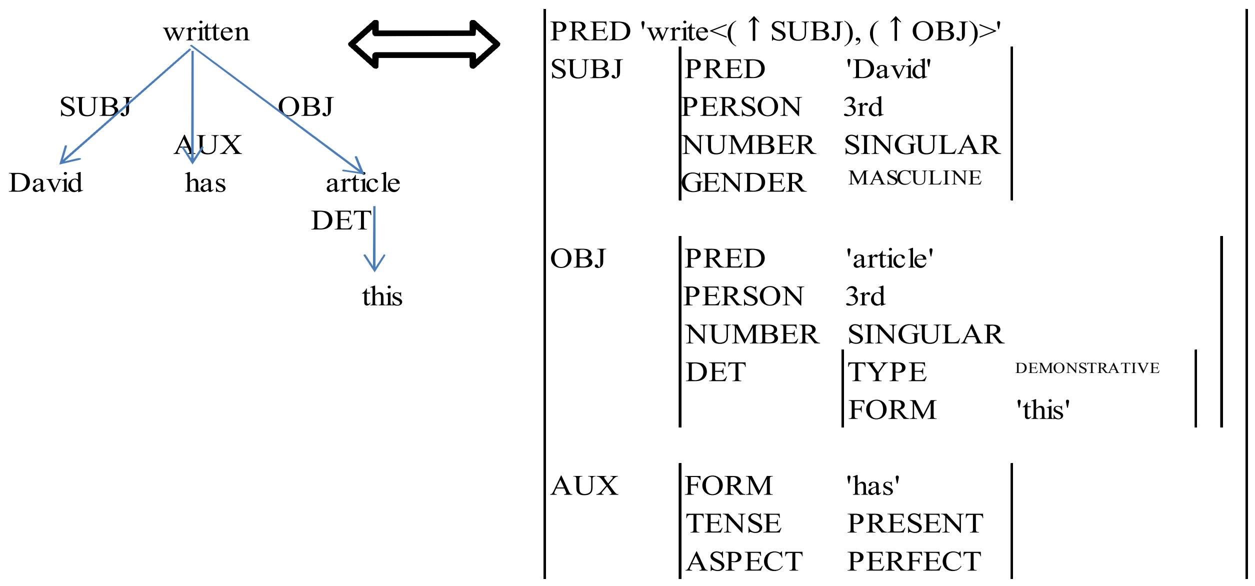


Figure 3.24. Lexicalized schema for the equivalence between the typed-dependency tree and the fragment functional structure for ‘David,’ ‘has,’ ‘written,’ ‘this’ and ‘article’ in the sentence (3.13)

The equivalence schema in Figure 3.17 for ‘written’ and ‘.’ can be lexicalized, as shown in Figure 3.25. The punctuation mark ‘.’ provides the verb ‘written’ with the PUNCT value.

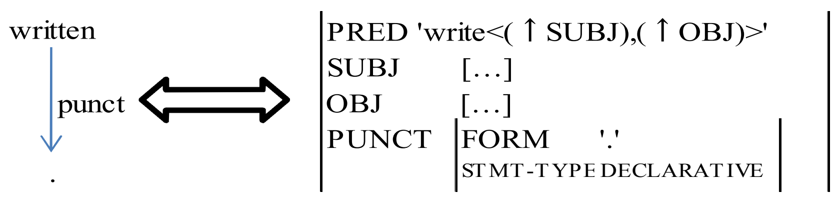


Figure 3.25. Lexicalized schema for the equivalence between the typed-dependency tree and the f-structure for 'write' and '.' in the sentence (3.13)

Lexicalized schema in Figure 3.25 for 'write' and '.,' along with that in Figure 3.24, share the same word; hence they are integrated to form one lexicalized schema, as shown in the figure below.

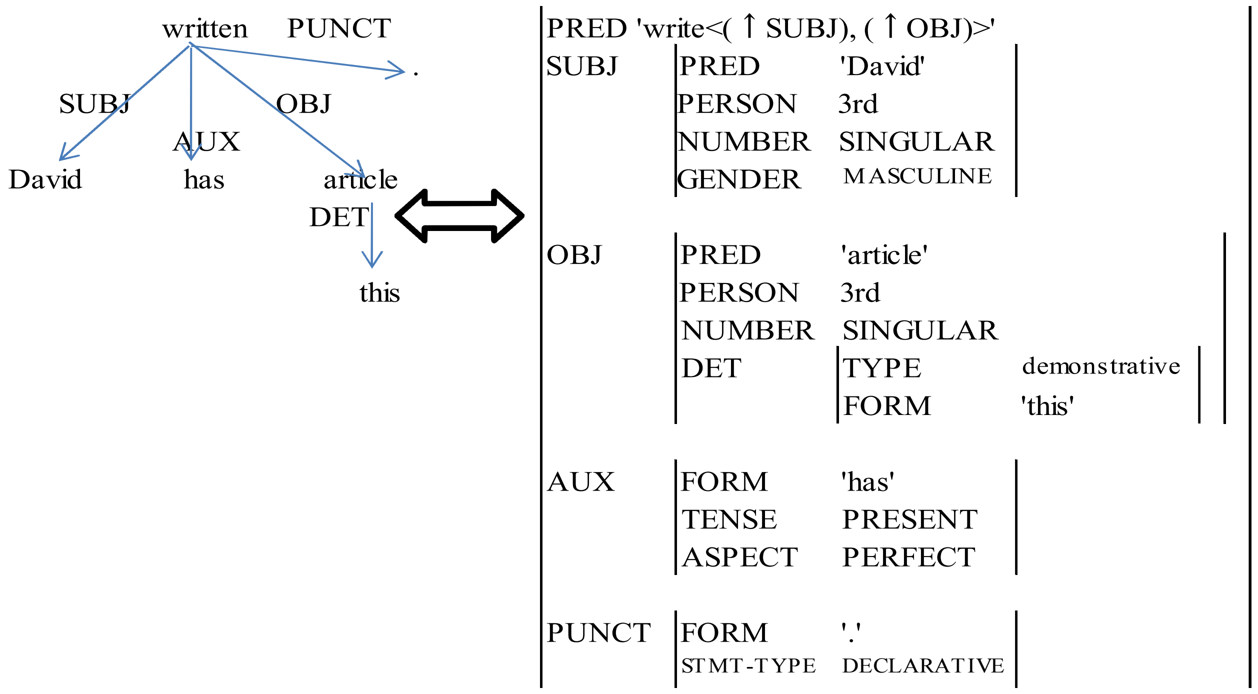
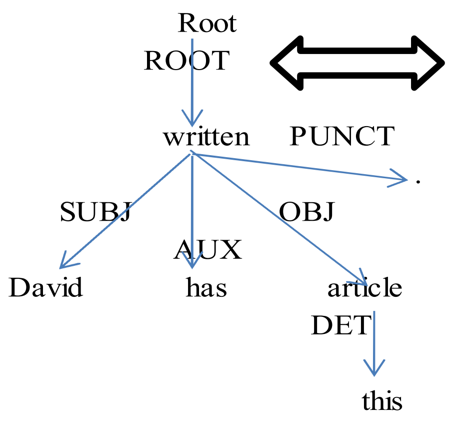


Figure 3.26. Lexicalized schema for the equivalence between the typed-dependency tree and the fragment functional structure for 'David,' 'has,' 'written,' 'this,' 'article' and '.' in the sentence

(3.13)

Finally, the whole sentence depends on an abstract element “Root” (see Section 5.3.1 for the definition of “Root”), as shown in the figure below. The result is the typed-dependency tree for the sentence ‘David has written this article.’ and its functional-structure equivalent.



```
     PRED 'write<(↑SUBJ), (↑OBJ)>'
     SUBJ   PRED      'David'
            PERSON    3rd
            NUMBER    SINGULAR
            GENDER    MASCULINE

     OBJ    PRED      'article'
            PERSON    3rd
            NUMBER    SINGULAR
            DET       TYPE      DEMONSTRATIVE
                      FORM      'this'

     AUX    FORM      'has'
            TENSE     PRESENT
            ASPECT    PERFECT

     PUNCT  FORM      '.'
ROOT        STMT-TYPE DECLARATIVE
```

Figure 3.27. The typed-dependency tree and its functional-structure equivalent for the sentence (3.13)

3.3.4 Fragment functional structures and Mel’čuk’s Criterion A

Notice that not all pairs of words in a sentence are in a dependency relationship in the sentence, and that they do not have equivalent fragment functional structures. For example, the word

'David' and 'article' cannot constitute a typed dependency relationship, because they cannot unify with each other to form a fragment functional structure, as shown in Figure 3.28. The word 'article' cannot be an argument or an adjunct for the word 'David.'

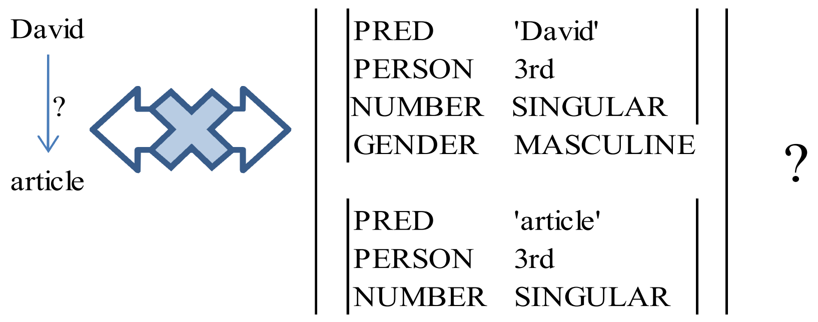


Figure 3.28. Non-equivalence between a wrong dependency and functional structure

This example shows that not all word pairs in a sentence necessarily form a dependency relationship. In other words, the word pair 'David' and 'article' does not constitute one of the "acceptable fragments of an utterance" (Gerdes & Kahane 2011, p.17) in the sentence 'David has written this article.'

Notice that the fragment 'David written' would not be considered to have dependency relationship according to Mel'čuk's Criterion A. This is because this fragment does not always constitute a prosodic unit (see Section 2.4.3.1), for example, when an adverb appears between the subject and the verb (e.g., Sarah *sometimes* read). However, this word pair can constitute a fragment functional structure, as already shown in Figure 3.13 above. Therefore, they are acceptable as one of the fragments of the utterance.

Oya (2013c, p.29) argued that two words in a sentence can constitute a dependency relationship if they can correspond to a fragment functional structure, with respect to the equivalence of a dependency relation between two words and its fragment functional structure. Oya also argued that Mel'čuk's Criteria A (see Section 2.4.3.1) for the presence of a dependency relationship between two words in a sentence (SSyntRel in Mel'čuk's term) can be revised in terms of the fragment functional structure. Note that Criterion A is proposed in Mel'čuk (2011, p.6), and Oya (2013c, p.29) proposed a revision of this criterion (Also see Section 2.4.3.1).

> Criterion A
>
> In a sentence, the lexemes L1 and L2 have a direct Synt-D link, only if L1 and L2 can form in language L an utterance – i.e., a prosodic unit, or a prosodic phrase of L – such as *the window*, *of John*, *spouts water* or *stained glass*, out of any context; the linear position of one of these lexemes in the sentence must be specified with respect to the other. (Mel'čuk 2011, p.6)

> Criterion A (revised):
>
> In a sentence, the lexemes L1 and L2 have a direct Synt-D link, only if L1 and L2 can form *a semantic unit* in language L.

Oya (2013c) argued that the term *semantic unit* in this revised Criterion A can be defined in terms of fragment functional structure. In other words, a word (or *an lexical element* in Oya (2013c)'s term) can form a semantic unit with another word in the same sentence iff the lexical information of these words can be integrated into one fragment functional structure. This study adapts Oya (2013c)'s revision of Mel'čuk's Criterion A.

3.4 Functional Structure as an Extension of Dependency Grammar: an Example from Pseud-Cleft Sentences

Oya (2013c, p. 29) argues that the ungrammaticality of pseudo-cleft sentences can be accounted for by the equivalence between typed-dependency trees and functional structures described in the previous section. Consider the sentence below.

(3.20) (=(18) in Oya (2013c, p.29))

*What the chairman has resigned is *from the board*

The typed-dependency tree for the sentence (3.22) is as follows.

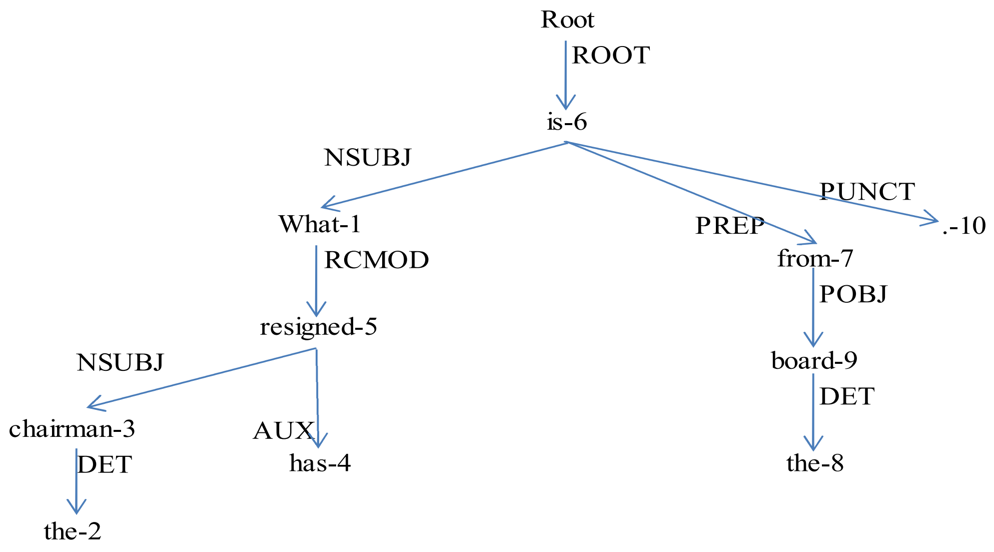


Figure 3.29. The typed-dependency tree for '*What the chairman has resigned is from the board.'

The typed-dependency tree itself is a well-formed one, yet the functional structure shown below for the same sentence violates the completeness constraint. The attribute PREP that is required by the verb ‘resigned’ is not given any value; hence, the sentence is ungrammatical.

```
|PRED      'is<NSUBJ>'                                                    |
|PREP     |PRED 'from<POBJ>'                                     |        |
|         |POBJ      |PRED     'board'                         | |        |
|         |          |DET     |FORM     'the'                  | |        |
|         |          |        |TYPE     DEFINITE               | |        |
|                                                                         |
|NSUBJ    |PRED      'PRO'                                                |
|         |FORM      'what'                                               |
|         |TYPE      RELATIVE                                             |
|         |RCMOD    |PRED     resigned<NSUBJ, PREP>'               |      |
|         |         |NSUBJ   |PRED     'chairman'             |    |      |
|         |         |        |DET     |FORM     'the'      |  |    |      |
|         |         |        |        |TYPE     DEFINITE   |  |    |      |
|         |         |                                            |      |
|         |         |AUX     |FORM     'has'        |             |      |
|         |         |        |TENSE    PRESENT      |             |      |
|         |         |        |ASPECT   PERFECT      |             |      |
|         |         |                                            |      |
|         |         |PREP    |???     |                          |      |
|                                                                         |
|TENSE     PRESENT                                                        |
|PUNCT    |FORM     '.'                          |                        |
|         |STMT-TYPE           DECLARATIVE        |                        |
```

Figure 3.30. The functional structure for ‘*What the chairman has resigned is from the board.’

The typed-dependency tree representation cannot account for the ungrammaticality of the sentence above. On the other hand, the completeness condition can account for the ungrammaticality of the functional structure that is equivalent to the typed-dependency tree. Based on this instance, Oya (2013c, p.30) argues that “the functional-structure representation has more explanatory capability than the typed-dependency representation.” It is also argued that “the equivalence of typed-dependency trees and functional structures can be regarded as a necessary extension of dependency grammar.” (Oya 2013c, p.30).

Obviously, only one instance is not enough to verify Oya’s (2013c) argument for the equivalence of typed-dependency trees and functional structures. What we need here is to explore other instances which seem to support the claim, that is, to find out ungrammatical sentences whose typed-dependency trees are well-formed, while their functional structures are ill-formed. This is one of the research questions in future study.

3.5 Summary

This chapter examined whether a typed-dependency tree for a sentence is equivalent to a functional-structure representation according to Lexical-Functional Grammar (LFG) (Bresnan 1978; Bresnan 1982; Kaplan & Bresnan 1982). The basic architecture of LFG was summarized in Section 3.2, with emphasis on structural correspondence between the constituent structure and the functional structure for a sentence, and on the well-formedness constraints for a functional structure. Next, Section 3.3 showed that a functional-structure representation for a sentence and its typed-dependency tree are equivalent. This equivalence supports the idea that LFG represents one direction of development of dependency grammar. This section also argued that Mel'čuk's Criterion A for the existence of dependency relationship between two words can be revised in terms of their possibility to constitute a fragment functional structure.

4. Typed-Dependency Trees and Graph Theory

4.1 Introduction

The previous chapter investigated the equivalence between typed-dependency syntactic trees and functional-structure syntactic representations according to the principles of LFG. This chapter continues the discussion of typed-dependency by exploring the representation of a typed-dependency syntactic tree as a directed acyclic graph (DAG). Moreover, this chapter also examines the idea of quantifying the structural property in terms of graph centrality. The advantage of dependency grammar representation is that a sentence's dependency can be interpreted as a DAG, allowing the formal syntactic properties to be defined and analyzed mathematically in terms of graph theory (Oya 2010b, 2011, 2013a, 2013b). Dependency grammar makes explicit the connections among the words in a sentence, or the *network* of words (Tesnière 1959). The characteristics of such a network can be quantified in several ways by drawing on approaches in the field of graph theory and network analysis. In other words, the structural properties of networks of words in sentences can be made explicit in dependency grammar and then quantified, using graph theory. Quantified structural properties can be useful for linguistic analyses that have previously relied on the subjective judgment of researchers, such as investigations into stylistic differences across different genres or similarities in syntactic structures for sentences in different languages. Quantitative approaches to syntactic structure can contribute to these types of linguistic analyses by bringing more objectivity.

The structure of this chapter is as follows. Section 4.2 introduces the basic tenets of graph theory. Section 4.3 examines centrality measures, including degree centrality and closeness centrality. The process for employing these centrality measures to analyze structural properties of typed-dependency trees is discussed in Sections 4.4 and 4.5. Specifically, Section 4.4 explores the application of centrality measures to show the similarity of functional-structure representations, and Section 4.5 illustrates how stylistic differences across genres are reflected by different distributions of centralities. Finally, Section 4.6 addresses the role of dependency distance in these representations.

4.2 Graph Theory

In order to understand how a typed-dependency tree can be represented as directed acyclic graphs, we first need to discuss the basic framework of graph theory (de Nooy, Mrvar & Batagelj 2005; Wasserman & Faust 1994; Wilson 1975, among many others). Figure 4.1 presents an example of a graph.

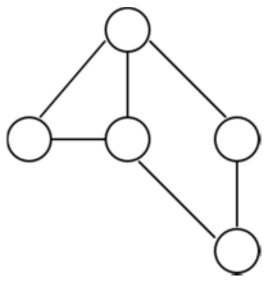


Figure 4.1. An example of a graph (n=5) (Oya 2010b, p. 394)

A graph consists of a set of *nodes* (or *vertices*) and a set of *edges* connecting these nodes (Wasserman & Faust 1994, p.94-95). In the figure above, the circles are the nodes and the lines connecting them are edges. The number of edges connected to a node is called the *degree* of the node (Wasserman & Faust 1994, p.101). Edges are considered *directed* if a direction from one node to the other is specified (directed edges are also called *arcs*) (Wasserman & Faust 1994, p.121). For a directed edge, the node from which an edge is extended is called the *head*, while the node into which an edge enters is called the *tail* (de Nooy et al. 2005, p.7). The number of edges extending from a head is called *outdegree*, while the number of edges entering a tail is called *indegree* (de Nooy et al. 2005, p.74). Nodes and edges can be given *labels*. A path is defined as a sequence of nodes and edges,

$$n_0, e_1, n_1, e_2, n_2, \ldots, n_{r-1}, e_r, n_r$$

where each edge e_i connects the nodes n_{i-1} and n_i ($1 \leq i \leq r$) (Biggs, Lloyd, & Wilson 1999, p.9). A graph is *acyclic* if no path from any node in the graph leads to the same node. Figure 4.2 presents an example of a directed acyclic graph.

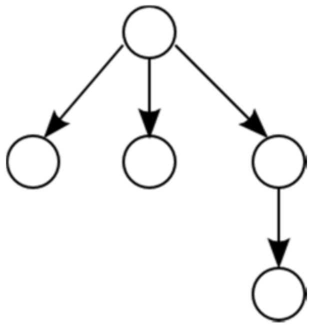


Figure 4.2. An example of a directed acyclic graph (n=5) (Oya 2010b, p. 394)

We can represent each word in a sentence as a node, and the edge connecting them as the dependency relationship between the words. We can label these edges with the grammatical functions that exist between words, i.e., we can categorize the dependency relationships into a number of dependency types. The result would be a typed-dependency directed acyclic graph of the sentence (Debusmann 2003; Debusmann & Kuhlmann 2007; Oya 2009, 2010a, 2010b, 2011, 2012). A typed-dependency directed acyclic graph for the sentence ‘I am studying graph theory’ is presented in Figure 4.3 below.

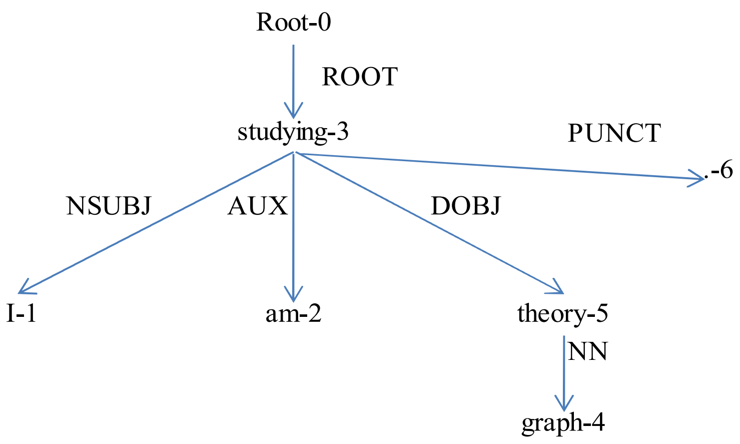


Figure 4.3. The typed-dependency directed acyclic graph for ‘I am studying graph theory.’

Typed-dependency directed acyclic graphs follow a number of constraints on well-formedness (Debusmann & Kuhlmann 2007). Two of these constraints are particularly relevant in this study. One constraint states that the indegree of all the nodes must be one, except for the abstract node “Root” (see Section 5.3.1). This is relevant to the equivalence of the typed-dependency tree representation and the functional-structure representation for a sentence (see Section 3.3), because the lexical information in each word in a sentence cannot be unified if any of the word depends on more than one word. Consider the figure below.

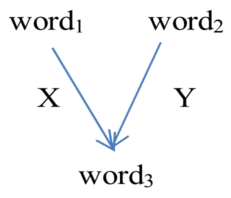


Figure 4.4. An ill-formed typed-dependency tree (more than one indegree)

In the typed-dependency tree above, the lexical information in word$_3$ will not be unified to the lexical information at one unique head; hence, there is no functional structure that is equivalent to the typed-dependency tree above[14].

The other constraint for the well-formedness of typed-dependency trees states that it has no cycle, as the word “acyclic” suggests (Oya 2010b). That is, no path in this graph leads from one node to the same node. Consider the figure below.

$$\begin{tikzcd}
\text{word}_1 \arrow[d, "X"'] \\
\text{word}_2 \arrow[d, "Y"'] \\
\text{word}_3 \arrow[uu, bend right=60, "Z"']
\end{tikzcd}$$

Figure 4.5. An ill-formed typed-dependency tree (cycled)

In the figure above, word$_2$ depends on word$_1$ with the dependency type X, word$_3$ depends on word$_2$ with the dependency type Y, and word$_1$ depends on word$_3$ with the dependency type Z. The lexical information in word$_2$ is unified to that in word$_1$, the lexical information in word$_1$ is unified to that in word$_3$, the lexical information in word$_3$ is unified to that in word$_2$, and this unification process forms an endless loop; hence, this tree has no functional-structure equivalent.

Directed acyclic graphs have been studied in other fields and they are often abbreviated as DAGs.

[14] The basic assumption of dependency grammar is that one word depends on only one word, i.e., one word is dependent on one dependency head. Words in constructions such as coordination or raising also depend on one dependency head. For the dependency structure of coordination, see Section 5.3.3 on the dependency type “conj,” and for the dependency structure of raising verbs, see Section 5.3.3 on the dependency type “acomp.”

4.3 Graph Centrality[15]

Various measures are defined in graph theory to specify the characteristics of a given graph (Freeman 1979; Wasserman & Faust 1994). Among these measures, *centrality* determines the relative importance of a node within a given graph.

Centrality can be defined in various ways, and the different definitions reflect what aspect of a given graph is made salient during the calculation. Following Oya (2010b, 2011, 2012, 2013a, 2013b), I focus on two types of centrality: *degree centrality* and *closeness centrality*. The following sections explore the relevance of these centrality measures for determining the structural property of the typed-dependency tree for a sentence.

4.3.1 Degree Centrality

Degree centrality is defined by the number of edges a given node has, i.e., the *degree* of a given node (Freeman 1979; Wasserman & Faust 1994). According to Wasserman & Faust (1994, p.179), for a graph with g nodes, the degree centrality $C'_D(n_i)$ of node n_i is the degree of this node divided by the number of nodes in the graph minus one. In the formula below, $d(n_i)$ is the degree of the ith node of the graph n, and g is the number of the nodes in the graph.

$$C'_D(n_i) = d(n_i) / (g-1) \tag{4.1}$$

Wasserman & Faust (1994, p.179)

[15] This section is based on Oya (2013a).

This definition can be extended to the whole graph, as shown in Freeman (1979) and Wasserman & Faust (1994). According to these authors, the degree centrality of a given graph is the sum of the maximum degree in the graph minus the degree of each of all the other nodes, divided by the largest possible sum of the maximum degree of the graph minus the degree of all the other nodes (Wasserman & Faust 1994, p.180). This calculation is represented in the following formula.

$$C_D = \frac{\sum_{i=1}^{g} [C_D(n^*) - C_D(n_i)]}{\max \sum_{i=1}^{g} [C_D(n^*) - C_D(n_i)]} \tag{4.2}$$

(Wasserman & Faust 1994, p.180)

The denominator of the formula above can be calculated directly, and equals (g-1)(g-2) (Freeman 1979).

The degree centrality of a given typed-dependency tree indicates the flatness of the typed-dependency tree for a sentence (Oya 2010b, 2011, 2013a). The flatness of a typed-dependency tree means the extent to which the words in a sentence are dependent on one particular word. Larger degree centralities indicate flatter typed-dependency trees.

For example, Oya (2013a) compares the degree centrality of the example sentence 'Sarah has read this book.' to that of 'Sarah would have read this book.' First, the sentence 'Sarah has read this book.' contains seven words, including the Root and the period (the typed-dependency tree for this sentence is Figure 2.2 above). The maximum degree in the typed-dependency graph for this sentence is five at the word 'read'; the sum of the maximum degree in the graph minus the degree of each of all the other vertices is (5-1)+(5-1)+(5-5)+(5-1)+(5-1)+(5-2)+(5-1)=23. The denominator is (7-2)(7-1)=30; hence the degree centrality of the typed-dependency tree for the sentence ‘Sarah has read this book.’ is 23/30≐0.767.

Next, consider the sentence ‘Sarah would have read this book.’ Figure 4.6 presents the typed-dependency tree for this sentence.

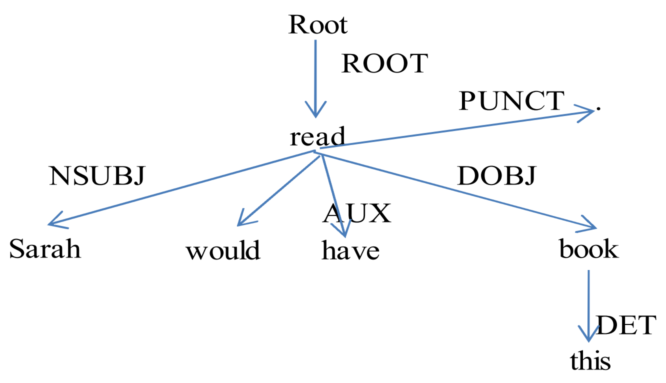


Figure 4.6. The typed-dependency tree for ‘Sarah would have read this book.’ (Oya 2013a, p.44)

This sentence contains eight words, with the Root and the period included in the word count, and the maximum degree is six at the word ‘read’; the sum of the maximum degree in the graph minus the degree of each of all the other vertices is (6-1)+(6-1)+(6-1)+(6-6)+(6-1)+(6-1)+(6-2)+(6-1)=34. The denominator is (8-2)(8-1)=42; hence the degree centrality of the typed-dependency tree for the sentence ‘Sarah would have read this book.’ is 34/42≐0.81. Oya (2013a) concludes that a typed-dependency tree with a flatter setting has a larger degree centrality, as the degree centralities of these examples indicate.

Oya (2013a) also argued that degree centralities of typed-dependency trees can indicate the flatness of sentences across English and Japanese. Consider a typed-dependency tree for an

English sentence in Figure 4.7 and its Japanese equivalent in Figure 4.8.[16]

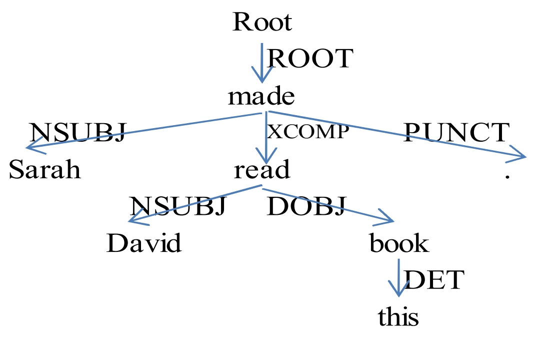


Figure 4.7. The typed-dependency tree for ‘Sarah made David read this book.’ (Oya 2013a, p.44)

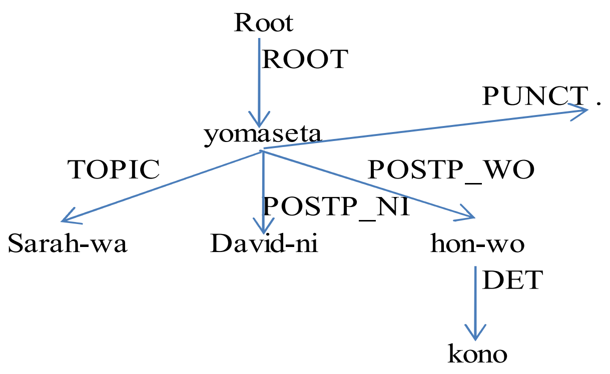


Figure 4.8. The typed-dependency tree for ‘Sarah-wa David-ni kono hon-wo yomaseta (Sarah made David read this book.)’ (Oya 2013a, p.44)

The degree centrality of the typed-dependency tree in Figure 4.7 is approximately 0.428, while that in Figure 4.8 is approximately 0.766. As this illustration suggests, the larger degree centrality of the typed-dependency tree for a Japanese sentence in Figure 4.8 quantitatively indicates that this tree has a flatter structure than its English counterpart.

[16] The dependency types for Japanese sentences used in this study are defined with examples in Section 6.

This study treats complex predicates in Japanese such as causatives (e.g., "yomaseru") or passives (e.g., "yomareru") as *monoclausal* both in typed-dependency trees and in their functional-structure representations. In other words, "yomaseta" in the tree above is one word.

This analysis for Japanese complex predicates can result in non-parallel functional structures for English and Japanese. The English functional structure contains two verbal predicates; hence, it is *biclausal*. The Japanese counterpart functional structure, on the other hand, contains one verbal predicates; hence, it is monoclausal (Matsumoto (1996) and Masuichi & Okuma (2003)). The treatment of zero pronouns will be discussed later in Section 6.4.

```
|         |NSUBJ  |PRED     'Sarah' |                         | |
|         |PRED    'make<NSUBJ, XCOMP>'                       | |
|         |TENSE   PAST                                       | |
|         |XCOMP  |PRED     'read<NSUBJ, DOBJ>'             | | |
|         |       |NSUBJ  |PRED     'David' |               | | |
|         |       |                                         | | |
|         |       |DOBJ   |PRED     'book'              |   | | |
|         |       |       |DET     |TYPE     definite | |   | | |
|         |       |       |        |FORM     'this'   | |   | | |
|         |       |       |PERSON  3rd                |   | | |
|         |       |       |NUM     SINGULAR           |   | | |
|         |                                                   | |
|         |PUNCT  |FORM     '.'         |                     | |
|ROOT     |       |STMT-TYPE DECLARATIVE|                     | |
```

Figure 4.9. The functional-structure representation for 'Sarah made David read this book.'

```
     |PRED       'yomaseta<SUBJ,OBJ,OBJ2>'                          |  |
     |SUBJ      |PRED      'PRO_i'          |                        |  |
     |          |TYPE      ZERO             |                        |  |
     |                                                               |  |
     |OBJ       |PRED      'PRO_j'          |                        |  |
     |          |TYPE      ZERO             |                        |  |
     |                                                               |  |
     |OBJ2      |PRED      'PRO_k'          |                        |  |
     |          |TYPE      ZERO             |                        |  |
     |                                                               |  |
     |TOPIC     |PRED      'Sarah_i'        |                        |  |
     |          |PERSON    1ST              |                        |  |
     |          |NUMBER    SINGULAR         |                        |  |
     |                                                               |  |
     |POSTP_wo  |PRED      'hon'                              |      |  |
     |          |DET      |FORM        'kono'          |      |      |  |
     |          |         |TYPE        DEMONSTRATIVE   |j     |      |  |
     |                                                               |  |
     |POSTP_ni  |PRED      'David_k'        |                        |  |
     |          |PERSON    3RD              |                        |  |
     |          |NUMBER    SINGULAR         |                        |  |
     |TENSE      PAST                                                |  |
     |PUNCT     |FORM      '.'              |                        |  |
ROOT |          |STMT-TYPE DECLARATIVE      |                        |  |
```

Figure 4.10. The functional-structure representation for 'Sarah-wa David-ni kono hon-wo yomaseta (Sarah made David read this book.)'

The biclausal analysis of complex predicates of Japanese sentences will yield the following typed-dependency tree and its functional-structure representation. The causative morpheme "-seta" is the root of the sentence, and it is the head of 'Sarah-wa,' 'David-ni,' and 'yoma.'

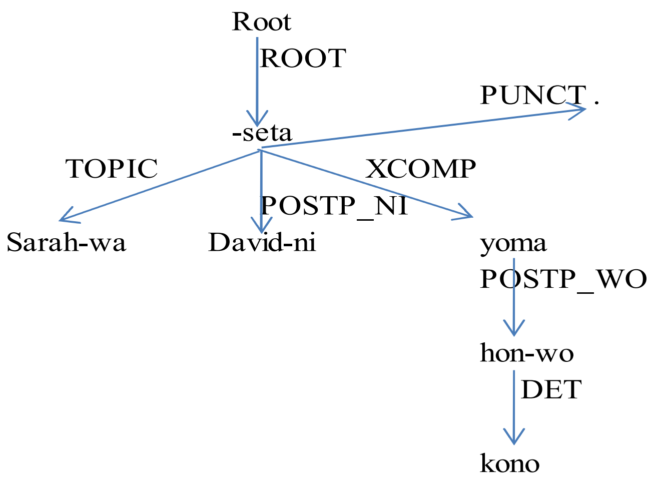


Figure 4.11. The biclausal typed-dependency tree for 'Sarah-wa David-ni sono hon-wo yomaseta.

(Sarah made David read this book.)’

```
     | PRED       '-seta<SUBJ,XCOMP>'                                   ||
     | SUBJ      |PRED      'PRO_i'        |                            ||
     |           |TYPE      ZERO           |                            ||
     |                                                                  ||
     | TOPIC     |PRED      'Sarah_i'      |                            ||
     |           |PERSON    3rd            |                            ||
     |           |NUMBER    SINGULAR       |                            ||
     |                                                                  ||
     | POSTP_NI  |PRED      'David_j'      |                            ||
     |           |PERSON    3rd            |                            ||
     |           |NUMBER    SINGULAR       |                            ||
     |                                                                  ||
     | XCOMP     |'yom<SUBJ, OBJ>'                                 |    ||
     |           |SUBJ      |PRED          'PROj'         ||       |    ||
     |           |          |TYPE          ZERO           ||       |    ||
     |           |                                                 |    ||
     |           |OBJ       |PRED          'PROk'         ||       |    ||
     |           |          |TYPE          ZERO           ||       |    ||
     |           |                                                 |    ||
     |           |POSTP_WO  |PRED          'hon_k'        ||       |    ||
     |           |          |DET           'kono'         ||       |    ||
     | TENSE      PAST                                                  ||
     | PUNCT     |FORM      '.'            |                             |
ROOT |           |STMT-TYPE DECLARATIVE    |                             |
```

Figure 4.12. The biclausal functional-structure representation for ‘Sarah-wa David-ni sono hon-wo yomaseta. (Sarah made David read this book.)’

In addition to the non-parallelism of functional structures for English and Japanese, the monoclausal analysis also yields a degree centrality which is different from the biclausal analysis. The degree centrality of the biclausal typed-dependency tree is approximately 0.618. This is smaller than the degree centrality of the monoclausal typed-dependency tree for the same sentence (approx. 0.766).

This study adopts monoclausal analysis for all the Japanese complex predicates, on the assumption that Japanese complex predicates function as independent syntactic units in the typed-dependency tree representation, while morphemes such as “-eru,” “-areru” or “-seru” cannot. Non-parallelism of functional structures for English and Japanese indicates the typological difference between these two languages (inflecting vs. agglutinative), and this difference can be numerically expressed by the degree centralities calculated from the monoclausal typed-dependency trees for Japanese complex predicates, which are larger than the degree centralities calculated from the biclausal typed-dependency trees.

4.3.2 Closeness Centrality

The distance from one node to another is represented by the number of arcs between them or the *path* between them. Freeman (1979) and Wasserman & Faust (1994) define closeness centrality as the reciprocal of the sum of the length of a path from one node to another in a graph. Closeness centrality of a graph is calculated through the following formula (Sabidussi 1966; Wasserman & Faust 1994, p.184). In this formula, *g* means the number of nodes, and $d(n_i, n_j)$ is the shortest path (*geodesic distance*) between the node n_i and n_j.

$$C_c(n_i)=1/\left[\sum_{j=1}^{g} d(n_i, n_j)\right] \tag{4.3}$$

(Wasserman & Faust 1994, p.184)

Wasserman & Faust (1994, p.185) point out that the maximum value attained by the formula (4.3) above depends on the number of nodes in a graph, and therefore it is difficult to compare values across networks of different sizes. Therefore, they refer to Beauchamp (1965) which suggests using the closeness centralities which are standardized by the following formula.

$$C_c(n_i)=\frac{g-1}{\sum_{j=1}^{g} d(n_i,n_j)} \tag{4.4}$$

(Wasserman & Faust 1994, p.185)

Wasserman & Faust (1994, p.185) point out that the value which is calculated through this formula can be viewed as the inverse average distance between node *i* and all the other nodes. This inverse average distance ranges from 0 to 1. It equals 1 when a node is *adjacent* (connected by one edge) to all the other nodes, and decreases as the nodes are aligned to a line.

Oya (2010b) proposed the term *path length* for the average length of a path from the ROOT to all the other in a graph. However, this figure does not range between 0 and 1. Therefore, Oya (2011) used the concept of closeness centrality as defined by Beauchamp (1965), and this study also follows this idea, expecially in Chapter 7.

The closeness centrality of a given typed-dependency tree indicates the embeddedness of the typed-dependency tree for a sentence (Oya 2010b, 2011, 2013a). The embeddedness of a tree means the extent to which one particular word is distant from other words along the paths in the tree. Larger closeness centralities mean that the words in a sentence are close to each other along the dependency paths, indicating that the typed-dependency tree is less embedded.

For example, Oya (2013a) compares the closeness centralities of the three example sentences as follows. First, there are six paths from the Root in the sentence 'Sarah has read this book.' (the typed-dependency tree for this sentence is Figure 2.2 above). The pahts are as follows: Root-read, Root-read-Sarah, Root-read-has, Root-read-., Root-read-book, and Root-read-book-this. The lengths of these paths are 1, 2, 2, 2, 2, and 3, respectively. The starting vertex is not included in the counting. Their average is 2, and the closeness centrality of this sentence is the inverse of 2, that is, 0.5.

Next, consider another example sentence 'My brother has read this book.' Figure 4.13 is the typed-dependency tree for this sentence.

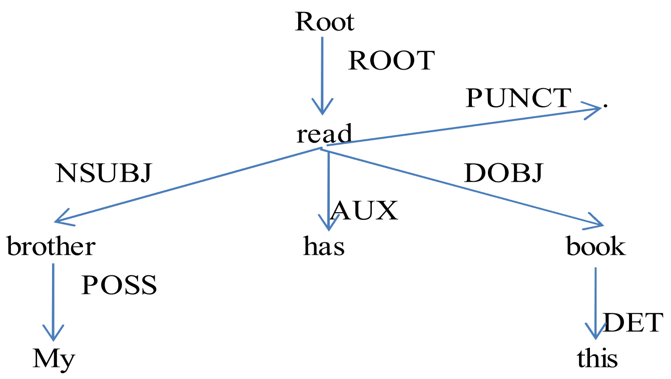


Figure 4.13. The typed-dependency tree for ‘My brother has read this book.’ (Oya 2013a, p.45)

This sentence has seven paths from the Root; Root-read, Root-read-brother, Root-read-brother-My, Root-read-has, Root-read-., Root-read-book, and Root-read-book-this. The lengths of these paths are 1, 2, 3, 2, 2, 2, and 3. The average length of them is 15/7≒2.142, whose inverse is the closeness centrality of this sentence, that is, 1/2.142≒0.467.

Finally, consider an example sentence ‘Sarah read the books David has.’ Figure 4.14 is the typed-dependency tree for this sentence.

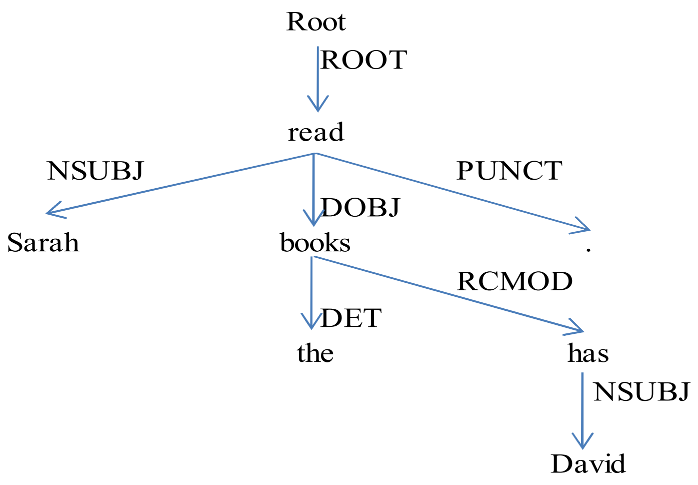


Figure 4.14. The typed-dependency tree for ‘Sarah read the books David has.’ (Oya 2013a, p.45)

This sentence has seven paths from the Root; Root-read, Root-read-Sarah, Root-read-books-the, Root-read-books, Root-read-books-has-David, Root-read-books-has, and Root-read-.. The lengths of these paths are 1, 2, 3, 2, 4, 3, and 2. The average length of them is 17/7≒2.44, whose inverse is the closeness centrality of this sentence, that is, 1/2.44≒0.41. Oya (2013a) concludes that a typed-dependency tree with a more embedded setting has a smaller closeness centrality, as the closeness centralities of these example sentences show.

4.4 Typed-Dependency Tree Centralities as Similarity Measures for Their Functional-Structure Representations[17]

This section discusses how typed-dependency tree centralities function as similarity measures for corresponding functional-structure representations. Dependency graphs are equivalent to functional structures (see Section 3.3). Therefore, the centrality measures of a given graph can indicate the structural characteristics of the corresponding equivalent functional structure. For example, consider the string “W1 W2 W3 W4 W5.” This string can be parsed to show a number of different typed-dependency trees and their equivalent functional structures. On one end, we could produce a typed-dependency tree that would represent the flattest type (Figure 4.15), and on the other end, we could produce a tree that would represent the most embedded type (Figure 4.17). Each of these typed-dependency trees also has an equivalent functional structure, namely the flattest functional structure (Figure 4.16) and the most embedded functional structure (Figure 4.17).

[17] This section is based on Oya (2013b).

The typed-dependency tree in Figure 4.15 and the functional structure in Figure 4.16 both have a flat setting, which means that W1 has the value 'W2' in terms of D1, 'W3' in terms of D2, and so on. This type of graph is called a *star graph*. The pieces of information for each of the words other than W1 are all unified to W1 immediately.

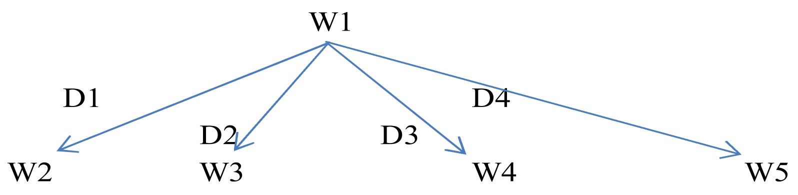


Figure 4.15. The flattest possible typed-dependency tree for a string 'W1 W2 W3 W4 W5'

'W1'	
D1	['W2']
D2	['W3']
D3	['W4']
D4	['W5']

Figure 4.16. The functional structure equivalent to the flattest possible typed-dependency tree for the string 'W1 W2 W3 W4 W5'

The typed-dependency tree in Figure 4.17 and the corresponding functional structure in Figure 4.18 both have a linear setting, which means that W1 has the value 'W2' in terms of D1, 'W2' has the value 'W3' in terms of D2, and so on. This type of graph is called a *line graph*. The pieces of information for each word are all embedded in its head.

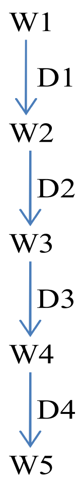


Figure 4.17. The most embedded possible typed-dependency tree for the string ‘W1 W2 W3 W4 W5’

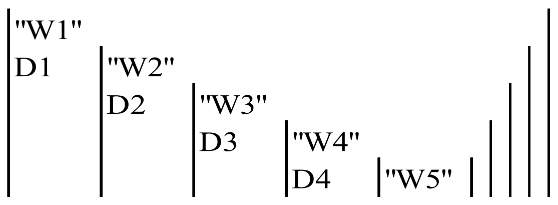


Figure 4.18. The functional structure equivalent to the most embedded possible typed-dependency tree for the string “W1 W2 W3 W4 W5”

Oya (2013b, p.155) claimed that “these two dependency graphs (the star-graph and the line-graph) are two extreme cases that represent the same number of nodes (or words) in terms of centrality.” The star-graph has both the highest degree centrality and the highest closeness centrality, as far as the graphs of the same node numbers are concerned. On the other hand, the line-graph has both the lowest degree centrality and the lowest closeness centrality, also as far as the graphs of the same node numbers are concerned. All the other dependency graphs of the same node numbers fall between these two extreme types of dependency graphs.

Oya (2010b) argued that it is possible to compare the centrality measures of graphs whose node numbers are different, because these centrality measures are calculated with the number of nodes in a given graph as the denominator and, therefore, these measures are normalized across graphs with different node numbers. Later on, however, Oya (2012) showed that longer sentences tend to have smaller degree centralities. Therefore, the argument by Oya (2010b) should be revised, and calculation of degree centrality should take into consideration the difference in word count.

Centrality measures of functional structures can also be employed to show structural similarity between different functional structures in the same language. If two functional structures for two sentences in the same language share a centrality measure, it is probable that these functional structures also share a structural aspect. This argument can be supported by employing more than one centrality measure, such as degree centrality to indicate the flatness of a sentence and closeness centrality to indicate the embeddedness of a sentence. By taking both degree centrality and closeness centrality into consideration, we can objectively determine the similarity of f-structures for different sentences without relying on subjective researcher judgments.

Centrality measures of functional structures can also be employed to indicate the structural similarity among different functional structures in English and Japanese (Oya 2013b). For example, we can choose an English sentence and its Japanese equivalent and calculate their similarity indices in order to quantify their structural similarity. Consider the following English-Japanese sentence pairs in (4.1–4.3). Their corresponding types-dependency trees and functional structures are then presented in Figures 4.19–4.30[18].

[18] The example sentences and their typed-dependency trees in this section are from Oya (2013b).

(4.1)

a. Sarah has already read this book.

b. Sarahwa mou kono honwo yonda.

(4.2)

a. The convenience store is on the other side of the street.

b. Konbiniwa toorino mukougawani arimasu.

(4.3)

a. There seems to be something wrong with this computer.

b. Kono pasokonwa dokoka koshoushiteirumitaida.

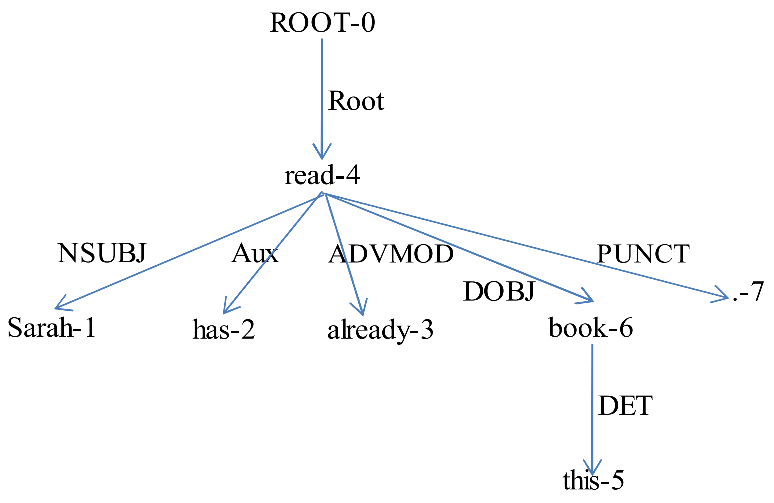


Figure 4.19. The typed-dependency tree for (4.1a) 'Sarah has already read this book.'

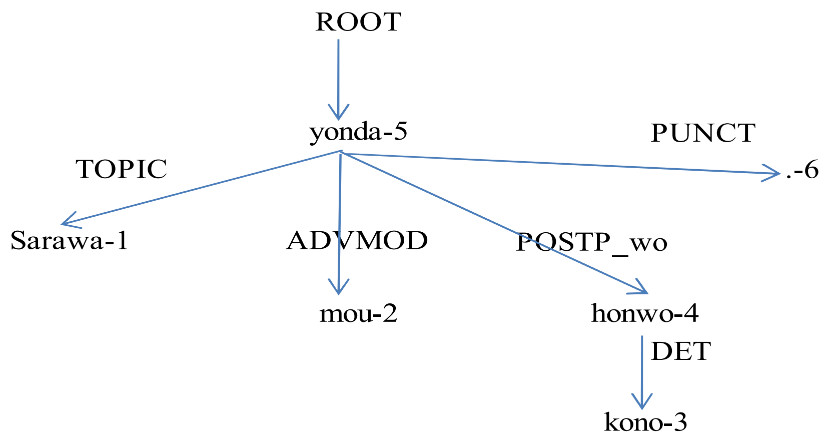


Figure 4.20. The typed-dependency tree for (4.1b) ‘Sarawa mou kono honwo yonda.’

```
|PRED        'read<SUBJ, OBJ>'                                   |
|NSUBJ      |PRED      'Sarah'     |                             |
|           |PERSON    3RD         |                             |
|           |NUMBER    SINGULAR    |                             |
|           |GENDER    FEMININE    |                             |
|                                                                |
|DOBJ       |PRED      'book'                           |        |
|           |PERSON    3RD                              |        |
|           |NUMBER    SINGULAR                         |        |
|           |DET      |FORM        THE        |         |        |
|           |         |TYPE        DET        |         |        |
|                                                                |
|AUX        |TENSE     PRESENT     |                             |
|           |ASPECT    PERFECT     |                             |
|                                                                |
|ADVMOD      {PRED     already'}                                 |
|STMT-TYPE   DECLARATIVE                                         |
```

Figure 4.21. The f-structure for (4.1a) ‘Sarah has already read this book.’

PRED	'yom-<SUBJ, OBJ>'	
TOPIC	PRED	'Sarah'
	CASE	WA
POSTP	PRED	'book'
	CASE	WO
SUBJ	PRED	'PRO'
	TYPE	ZERO
OBJ	PRED	'PRO'
	TYPE	ZERO
TENSE	PAST	
ADV	{PRED	'mou'}
STMT-TYPE	DECLARATIVE	

Figure 4.22. The f-structure for (4.1b) ‘Sarawa mou kono honwo yonda.’

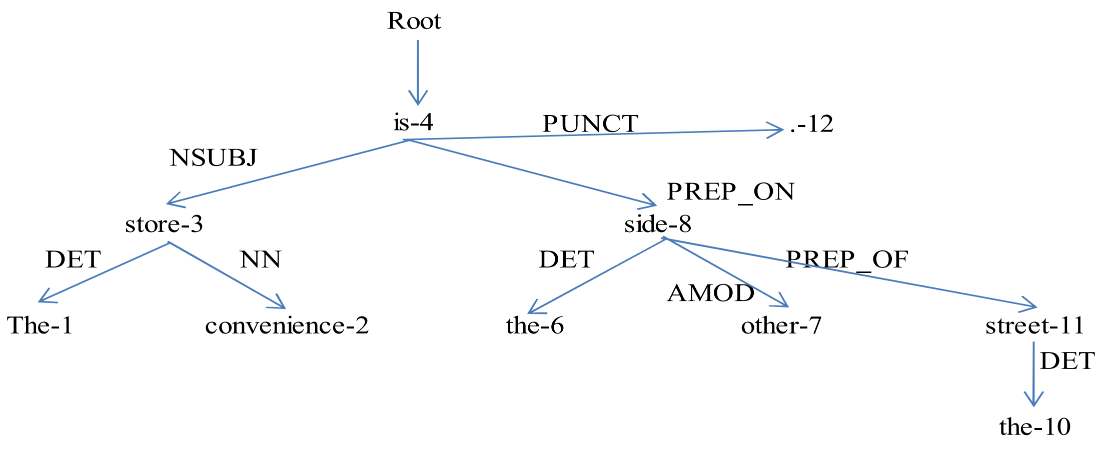


Figure 4.23. The typed-dependency tree for (4.2a) ‘The convenience store is on the other side of the street.’

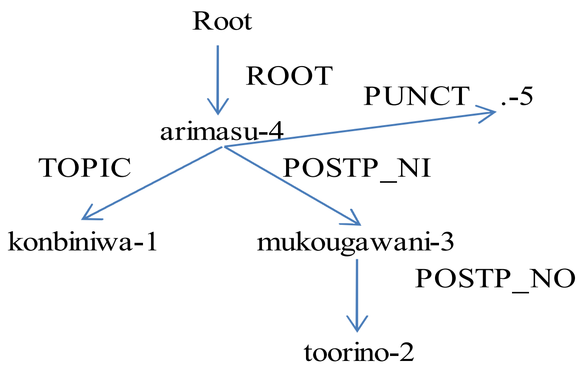


Figure 4.24. The typed-dependency tree for (4.2b) ‘Konbiniwa toorino mukougawani arimasu.’

```
PRED        'be<NSUBJ,PREP>'
NSUBJ       PRED        'store'
            PERSON      3RD
            NUMBER      SINGULAR
            DET         FORM    THE
                        TYPE    DET

            NN          PRED    'convenience'
                        PERSON  3RD
                        NUMBER  SINGULAR

PREP_ON     PRED        'side'
            PERSON      3RD
            NUMBER      SINGULAR
            DET         FORM    THE
                        TYPE    DET

            AMOD        PRED    'other'

            PREP_OF     PRED    'street'
                        PERSON  3RD
                        NUMBER  SINGULAR
TENSE       PRESENT
STMT-TYPE   DECLARATIVE
```

Figure 4.25. The f-structure for (4.2a) ‘The convenience store is on the other side of the street.’

```
|PRED        'arimasu<SUBJ,OBL>'                        |
|TOPIC      |PRED        'konbini'  |                   |
|           |CASE        WA         |                   |
|                                                       |
|POSTP_NI   |PRED        'mukougawa'            |       |
|           |CASE        NI                     |       |
|           |POSTP_NO|PRED       'toori'  |    |       |
|           |        |CASE       NO       |    |       |
|                                                       |
|SUBJ       |PRED        'PRO'      |                   |
|           |TYPE        ZERO       |                   |
|                                                       |
|OBL        |PRED        'PRO'      |                   |
|           |TYPE        ZERO       |                   |
|TENSE       PRESENT                                    |
|STMT-TYPE   DECLARATIVE                                |
```

Figure 4.26. The f-structure for (4.2b) ‘Konbiniwa toorino mukougawani arimasu.’

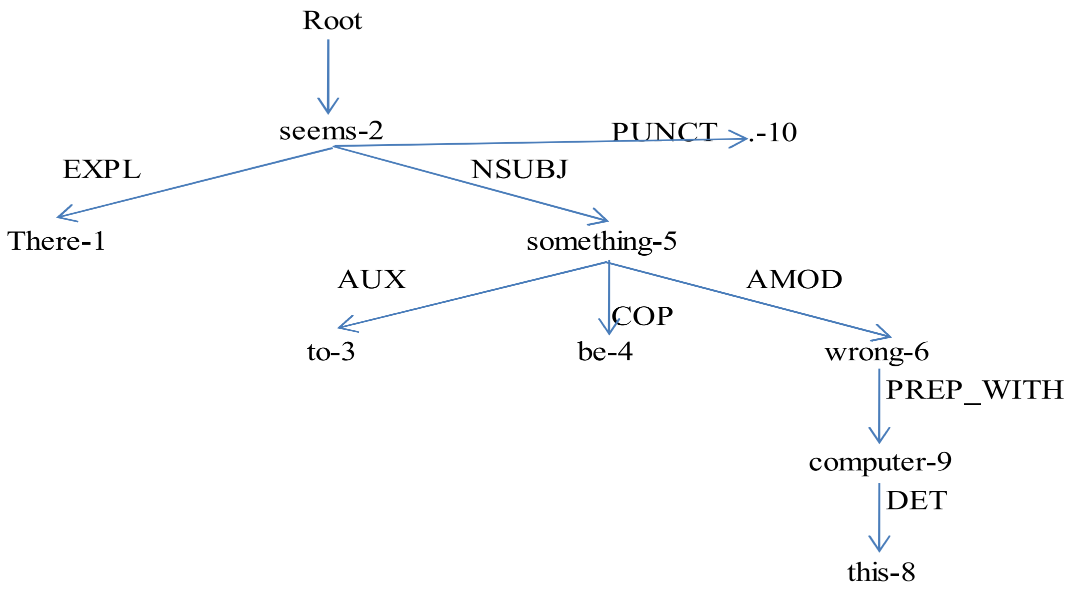


Figure 4.27. The typed-dependency tree for (4.3a) ‘There seems to be something wrong with this computer.’

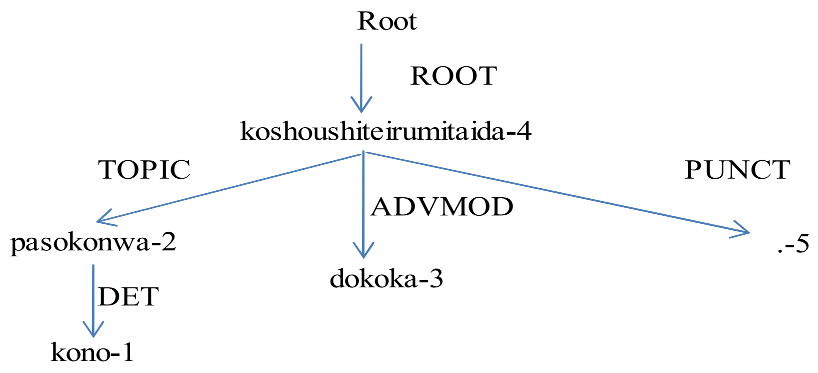


Figure 4.28. The typed-dependency tree for (4.3b) ‘Kono pasokonwa dokoka koshoushiteirumitaida.’

```
|PRED       'seems<NSUBJ>'                                              |
|EXPL       |PRED     'there'    |                                      |
|                                                                       |
|NSUBJ      |PRED     'something'                                     | |
|           |PERSON   3RD                                             | |
|           |NUMBER   SINGULAR                                        | |
|           |AUX      'to'                                            | |
|           |COP      'be'                                            | |
|           |AMOD     |PRED       'wrong'                           | | |
|           |         |PREP_WITH  |PRED    'computer'            | | | |
|           |         |           |DET     |FORM    'the'      | | | | |
|           |         |           |        |TYPE    DET        | | | | |
|TENSE      PRESENT                                                     |
|STMT-TYPE  DECLARATIVE                                                 |
```

Figure 4.29. The f-structure for (4.3a) ‘There seems to be something wrong this this computer.’

```
|PRED        'koshoushiteirumitaida<NSUBJ>'  |
|TOPIC       |PRED     'pasokon'  |          |
|            |CASE     WA         |          |
|            |DET      kono'      |          |
|                                            |
|NSUBJ       |PRED     'PRO'      |          |
|            |TYPE     ZERO       |          |
|ADVMOD      PRED      'dokoka'              |
|STMT-TYPE   DECLARATIVE                     |
```

Figure 4.30. The f-structure for (4.3b) 'Kono pasokonwa dokoka koshoushiteirumitaida.'

The degree centralities and closeness centralities for these English-Japanese sentence pairs are presented in Table 4.1 below.

Table 4.1. The degree centralities and closeness centralities of the typed-dependency trees (Oya 2013b, p.157)

	(4.1a)	(4.1b)	(4.2a)	(4.2b)	(4.3a)	(4.3b)
D	0.81	0.77	0.24	0.7	0.29	0.7
C	0.53	0.53	0.41	0.55	0.38	0.55

D: degree centrality; C: closeness centrality

The degree centralities of the English sentences (4.1a), (4.2a), and (4.3a) show that (4.1a) is the flattest of the three, because larger degree centralities indicate flatter typed-dependency trees. The degree centralities of the Japanese sentences (4.1b), (4.2b), and (4.3b) show that the flatness measures of their typed-dependency trees are quite similar. Comparing the English-Japanese pairs, we find that the difference between (4.1a) and (4.1b) is not as wide as other pairs.

The reason for this difference might have to do with the fact that they have different word counts. The word count of (4.1a) is eight, and that of (4.1b) is seven, including the root. The word count of (4.2a) is 11, and that of (4.2b) is six. Oya (2012) showed that sentences with smaller numbers of words tend to have larger degree centralities. It is difficult to draw any conclusions here, due to the limited data set of only three English-Japanese sentence pairs. In order to fully understand the relationship between degree centrality and structural similarity of sentences, it would be necessary to calculate the degree centralities of English-Japanese pairs in a parallel corpus, with their word counts taken into consideration. The same argument will be applied to closeness centralities. This issue will be discussed later in Section 7.5 of this study.

4.5 Centrality Measures and Different Genres of Texts

Oya (2010b) proposes that different distributions of degree centralities and closeness centralities in more than one set of sentences indicate the tendency of the sentences toward flat or embedded structures, which in turn, reflects genre differences. In order to visually represent these similarities and differences, we can plot these sentences on an x-y plane where the degree centrality of each sentence is indicated on the horizontal axis and the closeness centrality of each sentence is represented on the vertical axis. If the group of sentences has a tendency toward flatter structures (*parataxis*), the plots gather in the top right quadrant of the plane. In contrast, if the group of sentences has more embedded structures (*hypotaxis*), the plots gather in the bottom left quadrant of the plain. If two sets of sentences have different degrees of flatness and embeddedness, we will have two plot graphs on the x-y plane with different distributions, as shown in Figure 4.31 below.

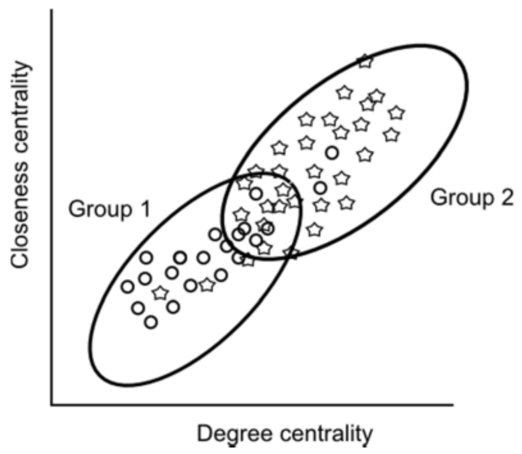


Figure 4.31. The centrality continuums of Group 1 sentences (represented as circles) and Group 2 sentences (represented as stars)

This type of representation shows how groups of sentences or texts written by writers with different levels of ability or for different purposes can produce different distributions on an x-y plane in terms of their degrees of centrality and closeness.

In order to see how the distribution of degree centralities and closeness centralities reflect differences in genres, Oya (2010b) calculated the degree and closeness centralities of the typed-dependency trees acquired from three small-scale corpora: (1) essays in English on the same topics ("self-introduction" and "happiness factors") written by Japanese students studying English (these data were also used in Yoshida et al. (2009)) (henceforth Japanese), (2) abstracts of research articles in *Studies in Second Language Acquisition* vol. 31 (published in 2009) and vol. 32 (published in 2010) (henceforth SLA), and (3) the first chapter of the abridged version of *The Golden Bough* (by Sir James Frazer, published in 1912; henceforth Golden). The descriptive statistics for these data are presented in Table 4.2. The distributional results for the degree centralities and closeness centralities for the Japanese, SLA, and Golden data sets are presented in Figures 4.32, 4.33, and 4.34, respectively.

Table 4.2. The descriptive statistics of the corpora used in Oya (2010b)

	Word per sentence			Degree centrality			Closeness centrality		
	Japanese	SLA	Golden	Japanese	SLA	Golden	Japanese	SLA	Golden
Mean	16.92	24.34	24.01	0.32	0.18	0.23	0.33	0.33	0.37
SD	8.03	8.72	10.78	0.18	0.08	0.16	0.09	0.09	0.14

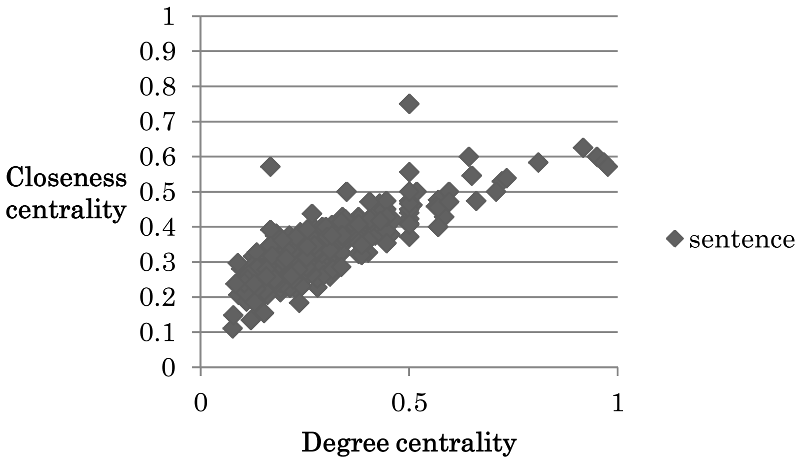


Figure 4.32. The distribution of degree centralities and closeness centralities of the typed-dependency trees in Japanese (n=342)

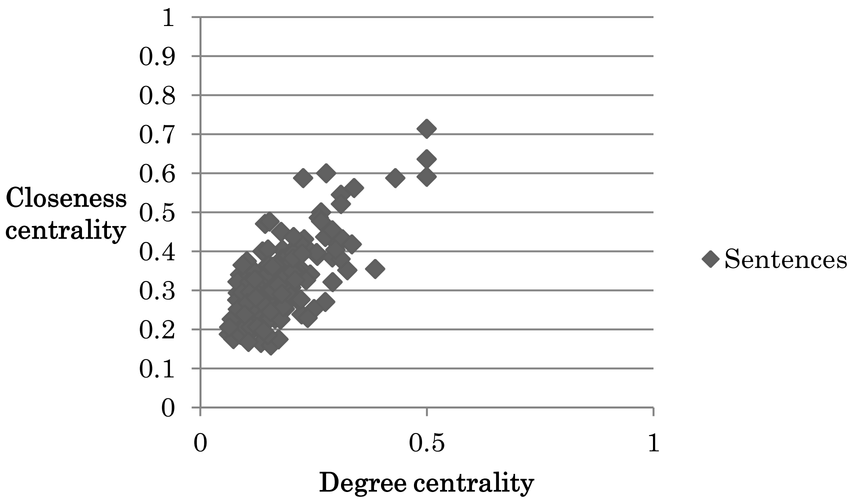


Figure 4.33. The distribution of the degree centralities and closeness centralities of the typed-dependency trees in SLA (n=160)

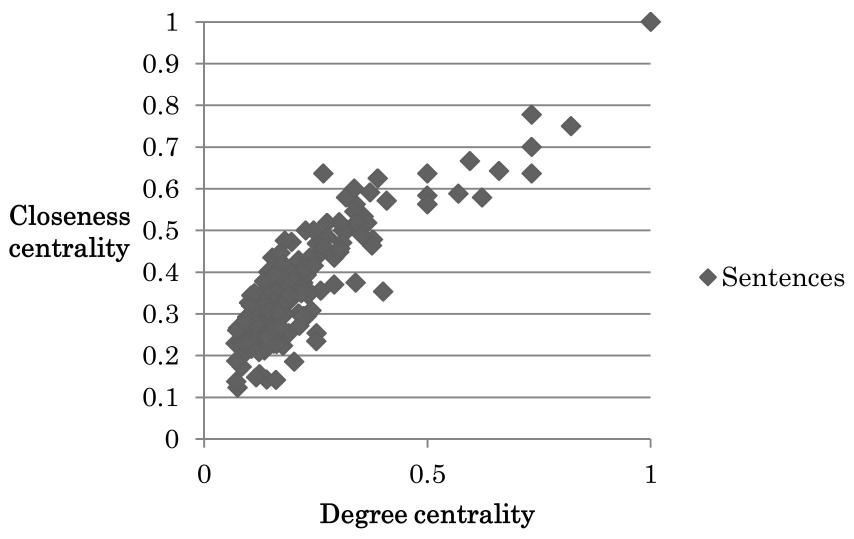


Figure 4.34. The distribution of the degree centralities and closeness centralities of the typed-dependency trees in Golden Bough (n=156)

Oya's (2010b) proposal and analysis has a number of drawbacks. First, it is uncertain whether the degree centrality and closeness centrality constitute rectangular coordinates. If they do not, it is meaningless to consider the distribution of sentences with the degree centrality on the horizontal axis and the closeness centrality on the vertical axis. Second, since the degree centrality of a typed-dependency tree can take only a fixed number of values that are determined by the number of words in the typed-dependency tree, it is also meaningless to compare the degree centralities of sentences with different word counts.

To address these drawbacks, Oya (2012) focused on the relationship between the centrality measure and the word count of each sentence in different sets of sentences, as well as the distribution of degree centralities for sentences with fixed word counts. The results of Oya (2012) will be discussed and replicated later in Section 7.6.

4.6 Dependency Distance

Another drawback of these centrality measures is that they abstract away the linear order of words in a sentence, and in this way, they do not show the surface dependency distance between a head and its dependent. Dependency distance is another feature that can be extracted from the typed-dependency tree representation of a sentence. This feature takes into consideration the surface dependency distance between a head and its dependent (Oya 2011). The dependency distance of a dependency relation is the number of words between its head and tail in the surface word order of a sentence. Consider the typed-dependency tree shown below in Figure 4.31.

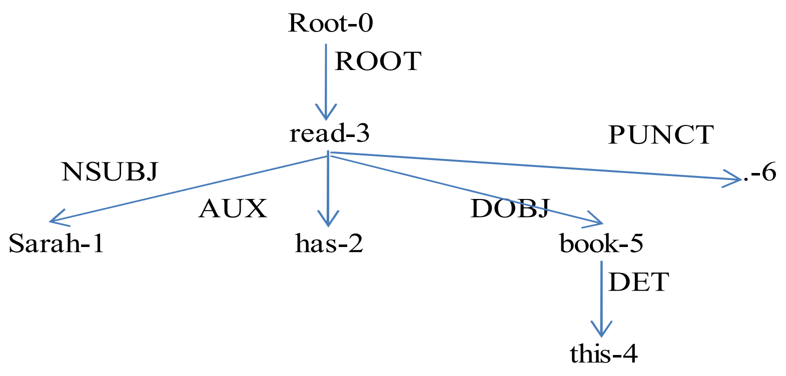


Figure 4.35. The typed-dependency tree representation for ‘Sarah has read this book.’

The number that corresponds to each word represents its place in the surface word order. Thus, the dependency distance between ‘Sarah’ and ‘read’ is two, ‘has’ and ‘read’ is one, and so on. We can obtain the average dependency distance of the typed-dependency tree above as the sum of all the dependency distances divided by the number of dependency relations. In the typed-dependency tree above, the average dependency distance is (3 + 2 + 1 + 2 + 1 + 3) / 4 = 3.

This calculation can serve as one of the measures for sentence complexity, with the assumption that a sentence becomes more complex in proportion to its average dependency distance. Dependency Locality Theory (DLT) (Gibson 1998, 2000) proposes that the syntactic complexity of sentences increases in proportion to the length of syntactic dependency, and that syntactic complexity can be predicted by two factors: "storage cost" and "integration cost." Storage cost refers to the cost of keeping the previous words in a speaker's memory. Integration cost refers to the cost of connecting the words in a speaker's memory. Longer dependency lengths require more storage cost, thus increasing the difficulty of processing the dependency relationships.

Temperley (2007) proposes that DLT can be applied to the production of sentences. He suggests that there are different preferences for longer or shorter dependency distances with respect to the syntactic environments in which dependency relations appear. For example, he shows that subject noun phrases in S-V order quotations tend to have a smaller number of words than subject noun phrases in V-S order quotations (Temperley 2007, p.307). Consider the examples below in (4.3a-d), taken from Oya (2011).

(4.3)
a. "I've read this book," Sarah said.
b. "I've read this book," said Sarah.
c. "I've read this book," my supervisor said.
d. "I've read this book," said my supervisor.

The subject noun phrase in the S-V order quotation in (4.3a) has one word, while the subject noun phrase in the V-S order quotation in (4.3d) has two words. The dependency distance of the main verb 'said' and its indirect-speech complement is shorter in (4.3d) than that in (4.3c), and (4.3d) is preferred to (4.3c).

Temperley's (2007) observation about the differences in dependency-length preference according to syntactic environments can be applied to our methods for calculating sentence complexity. For example, if Japanese learners of English prefer less complex sentences to more complex ones when they produce English sentences, the average dependency distance of their written productions will be shorter than that of native speakers. In this way, the average dependency distance in a given text can reflect the writer's preference in terms of sentence complexity. This point will be further explored in Chapter 7 where this assumption is tested with a large amount of corpus data.

4.7 Summary

This chapter continued the discussion of typed-dependency by exploring the representation of a typed-dependency syntactic tree as a directed acyclic graph (DAG). Moreover, this chapter also examined the idea of quantifying structural properties in terms of graph centrality. The advantage of dependency grammar representation is that a sentence's dependency can be interpreted as a DAG, allowing the formal syntactic properties to be defined and analyzed mathematically in terms of graph theory (Oya 2010b, 2011). Section 4.2 introduced the basic tenets of graph theory. Section 4.3 examined centrality measures, including degree centrality and closeness centrality. The process for employing these centrality measures to analyze structural properties of typed-dependency trees was discussed in Sections 4.4 and 4.5. Specifically, Section 4.4 explored the application of centrality measures to show the similarity of functional-structure representations by comparing the typed-dependency trees of English and Japanese sentences in terms of their centrality measures. It is pointed out that it would be necessary to calculate the centrality measures of English-Japanese pairs, not just only three sentence pairs, but an English-Japanese parallel corpus with their word counts taken into consideration, so as to understand the relationship between centrality measures and structural similarity of sentences. This issue will be the topic of Section 7. Section 4.5 illustrated how stylistic differences across genres are reflected by different distributions of centralities (Oya 2010b). It is pointed out that Oya's (2010b) analysis on different distribution of centralities had some drawbacks (degree centralities and closeness centralities may not be rectangular coordinates, and the number of different values of the degree centrality of a typed-dependency tree is determined by the number of words in the tree), and that these drawbacks will be addressed in Section 7. Finally, Section 4.6 addressed the role of dependency distance in these representations.

5. Dependency Parsing of English Sentences by Stanford Parser

5.1 Introduction

The preceding chapters have examined the significance of typed-dependency representations for investigating syntactic structure. This chapter furthers this discussion by introducing the Stanford Parser (de Marneffe & Manning 2012), which is a state-of-the-art parser used in this study for acquiring typed-dependency tree representations for English sentences. Chapter 4 explored typed-dependency trees and their graph-theoretical analyses, showing that graph centrality measures can be employed for more objective linguistic analyses. However, it is time-consuming for the researcher to construct typed-dependency trees for each sentence in a corpus and manually calculate their centrality measures. This chapter proposes a more efficient method, namely a *syntactic parser,* to obtain typed-dependency trees for individual sentences in large corpora.

Syntactic parsers are computer applications that are designed to obtain the syntactic parse for an input sentence. The Stanford Parser is a state-of-the-art dependency parser that extracts typed-dependency parses from phrase-structure parses (de Marneffe, MacCartney & Manning 2006). It was created in 2005 because the other available dependency parsers, such as Minipar (Lin 1998) and the Link Parser (Sleator & Temperley 1993), were not as robust and accurate as phrase-structure statistical parsers, such as Collins (1999) and Charniak (2000). Since then, the Stanford Parser has been implemented by researchers in a diverse set of fields, such as biomedical text mining (Pyysalo, Ginter, Haverinen, Heimonen, Salakoski, & Laippala 1997), sentiment extraction (Kessler 2008), textual entailment recognition (Adams, Nicolae, Nicolae, & Harabagiu 2007), machine translation (Genzel 2010), and sentence complexity calculation (Lu 2010).

This chapter is structured as follows. Section 5.2 describes the output format of the

Stanford Parser. Section 5.3 provides the definition of each dependency type used in the parsed output of the Stanford Parser. These definitions are based on the revised version of Mel'čuk's Criteria introduced in Chapter 3. By doing this, these dependency types are provided with a theoretical backbone, viz. a tradition of dependency grammar which was started by Tesnière and developed by Mel'čuk. In addition to this, each dependency type is defined with respect to example sentences which contain the dependency type. For each example sentence, its typed-dependency tree and its equivalent functional-structure representation are provided, so that it can be shown that the typed-dependency tree is equivalent to the functional-structure representation, as indicated in Section 3.3. Section 5.4 explains the differences among the different output styles of the Stanford Parser.

5.2 The Output Format of the Stanford Parser

This section introduces the output format of the Stanford Parser. The Stanford Parser's parsed output for a sentence consists of a number of *triples* and has the following format presented in (5.1).

(5.1)
Dependency-type(head-x, tail-y)

This format is called a triple because it is made up of three parts. For example, the parsed output for the sentence 'Sarah has read this book' is shown below in (5.2).

(5.2)
nsubj(read-3, Sarah-1)

```
aux(read-3, has-2)
root(ROOT-0, read-3)
det(book-5, this-4)
dobj(read-3, book-5)
punct(read-3, .-6)
```

Each line of the output represents a dependency relationship between two words in the input sentence. The first line of this parsed output shows (1) that the head of the dependency relation is ‘read,’ which is the third word of this sentence, (2) that the tail of the dependency relation is ‘Sarah,’ which is the first word of the sentence, and (3) that their dependency type is ‘nsubj’ (the definition of each dependency type used in the Stanford Parser is discussed in the following section).

The parsed output of the Stanford Parser for a sentence is equivalent to the typed-dependency tree of the same sentence. The dependency-tree representation of the sentence ‘Sarah will read the book.’ is shown in the figure below.

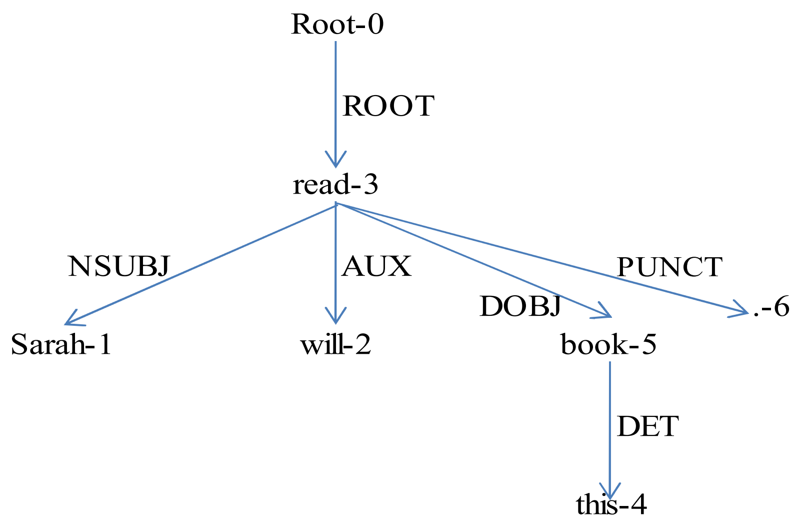


Figure 5.1. The typed-dependency tree for ‘Sarah will read this book.’

The typed-dependency tree is equivalent to the functional-structure representation (see Section 3.3). The functional-structure representation of the sentence ‘Sarah will read the book.’ is shown below in the figure below.

```
        | PRED     'read<NSUBJ,DOBJ>'                              | |
        | NSUBJ   | PRED 'Sarah'              |                    | |
        |         | PERSON     3RD            |                    | |
        |         | NUMBER     SINGULAR       |                    | |
        |         | GENDER     FEMININE       |                    | |
        |                                                          | |
        | DOBJ    | PRED 'book'                       |            | |
        |         | PERSON     3RD                    |            | |
        |         | NUMBER     SINGULAR               |            | |
        |         | GENDER     NEUTRAL                |            | |
        |         | DET       | FORM 'the'      |     |            | |
        |         |           | TYPE DEFINITE   |     |            | |
        |                                                          | |
        | AUX     | FORM       'will'         |                    | |
        |         | TENSE      FUTURE         |                    | |
        |                                                          | |
        | PUCNT   | FORM       '.'            |                    | |
| ROOT  |         | STMT-TYPE  DECLARATIVE    |                    | |
```

Figure 5.2. The functional structure for ‘Sarah will read the book.’

5.3 The Definition of Each Dependency Type used in Stanford Dependencies

This section introduces the definitions for each dependency type in Stanford Dependencies (de Marneffe & Manning 2012), following Mel’čuk’s dependency grammar. Stanford Dependencies contain 55 dependency types for the parsing output of the Stanford Parser. The dependency types represent the different functions of dependents in terms of their heads. Each dependency type is defined with example sentences and dependency trees. The hierarchy of dependency types is shown in Appendix I. Some of these types (e.g., agent, xdep, and xsubj) are not implemented in the default setting of the Stanford Parser (for different output format options of the Stanford Parser, see Section 5.4). In terms of the output of the Stanford Parser, the dependency type "arg" and "mod" are always realized as their subtypes, and they do not appear in the parsed output. In addition, the subtypes "comp," "obj," and "subj" are also realized as their subtypes (e.g., "nsubj" and "iobj") and do not appear in the parsed output. Though the category "aux" has two subcategories (namely, "auxpass" and "cop"), "aux" itself appears in the parsed output if the tail of the given dependency is a modal auxiliary. If the parser cannot determine a given dependency type, it yields "dep."

The definitions of the dependency types are presented in this section in almost the same order as the dependency hierarchy shown in Appendix I, i.e., "root," "aux," "arg," "cc," "conj," "expl," "mod," "parataxis," and "punct." Note that "ref" is defined along with "rcmod" and "rel," and "sdep" is omitted because it is not implemented in the Stanford Parser. The subcategories of "aux," "arg," and "mod" are also defined. There are some instances in which the actual parsed output for a sentence by the Stanford Parser does not yield a correct typed-dependency tree. In this section, these instances are pointed out as much as possible. The parsing accuracy of the Stanford Parser will be dealt with in Section 7.3.

These dependency types are categorized according to the deep syntactic relations (DSyntRels) in Mel'čuk's dependency grammar, as shown in Table 5.1.

Table 5.1. English dependency types in Stanford Dependencies categorized according to the inventory of DSynt-Rels in Mel'čuk (2011, p.6)

coordinate DSyntRels		subordinate DSyntRels									
		weak subordi-nate DSyntRel	strong subordinate DSyntRels								
			modification		complementation						
COORD	QUASI-COORD	APPEND	ATTR	ATTRdescr	I	II	III	IV	V	VI	IIdir-sp
conj	prep	advmod	Those typed as "mod"	rcmod	nsubj, nsubjpass, csubj, csubjpass	dobj, acomp, ccomp,xcomp	iobj	prep, pobj, pcomp			ccomp

In the following sections, each dependency type is defined with respect to the criteria for surface syntactic relations (SSyntRels) proposed by Mel'čuk (2009, 2011), i.e., Criterion A for the presence of the surface syntactic relation between two words (or elements), Criterion B for the orientation of the surface syntactic relation, and Criterion C for the type of the surface syntactic relation. Along with these definitions, typed-dependency trees and functional-structure representations for the example sentences are also illustrated, in order to clarify the equivalence between typed-dependency trees and functional-structure representations (see Section 3.3).

Some dependency types in Stanford Dependencies are used to cover the grammatical functions used in the standard LFG, e.g., "nsubj" for SUBJ, "dobj" for OBJ, or "prep" and "pcomp" for PCASE[19]. This fact, however, does not necessarily lead Stanford Dependencies to yield the same analysis of a given sentence as the standard LFG does, and therefore, the functional structure which is equivalent to the typed-dependency tree of a given sentence can be different from the f-structure within the framework of the standard LFG. This study does not account for the difference between them in detail, because the main theme of this section is to define each of the dependency types in Stanford Dependencies in terms of Mel'čuk's criteria.

[19] The grammatical function PCASE is introduced by Kaplan & Bresnan (1982, p.197). This specifies that a preposition has the function to assign a case to a noun.

5.3.1 Root

This dependency type is one where the dependency head is an abstract element “Root” and the dependent is the root word of the sentence. The presence of the abstract element “Root” enables us to produce an analysis in which all the words in a sentence are dependent on another entity. In the example sentence below, the root of the sentence is the verb ‘read.’

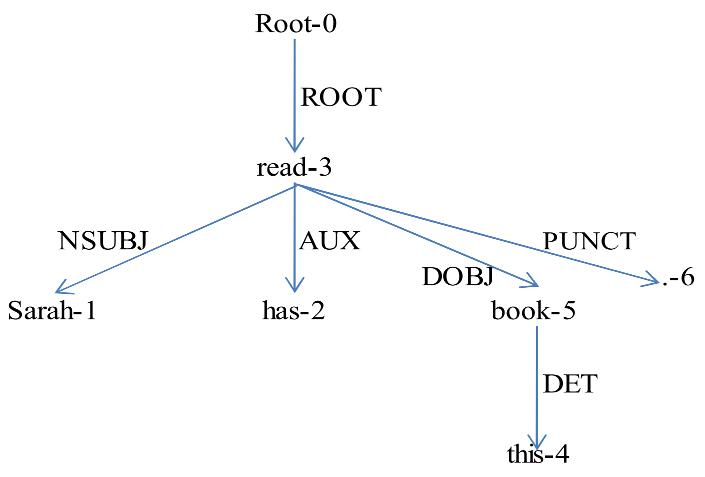


Figure 5.3. The typed-dependency tree for ‘Sarah has read this book.’

This dependency type is exceptional because the head of this dependency is not a word. This is because the main predicate of a sentence must depend on some element in a level of representation higher than the sentence level, i.e., the discourse level. Moreover, all the sentences in a text must depend on some element in that higher level of representation.[20] The

[20] Gerdes & Kahane (2011, p.22) also argue that we can define discourse structures for a whole text if we consider the sentences in the text as “discourse units”, which are minimal fragments of “a discourse connection graph” for the text.

typed-dependency tree shown below illustrates how these two sentences depend on the same node.

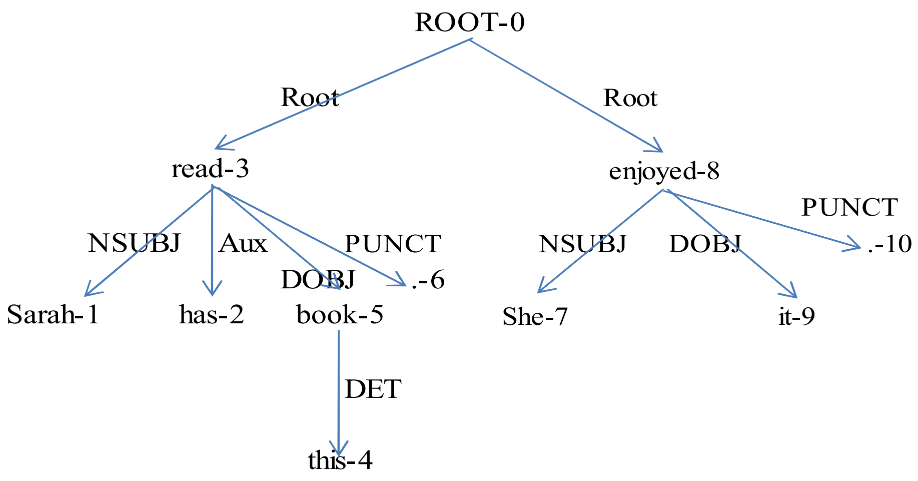


Figure 5.4. The typed-dependency tree for ‘Sarah has read this book. She enjoyed it.’

Thus, it is possible to consider the discourse structure as a typed-dependency tree, i.e., each sentence in a discourse depends on the discourse root with a *discourse dependency type*. For example, the first sentence in the typed-dependency tree above depends on the discourse root with the discourse dependency type “TOPIC,” and the second sentence depends on the discourse root with the discourse dependency type “RESULT,” as shown in the figure below.

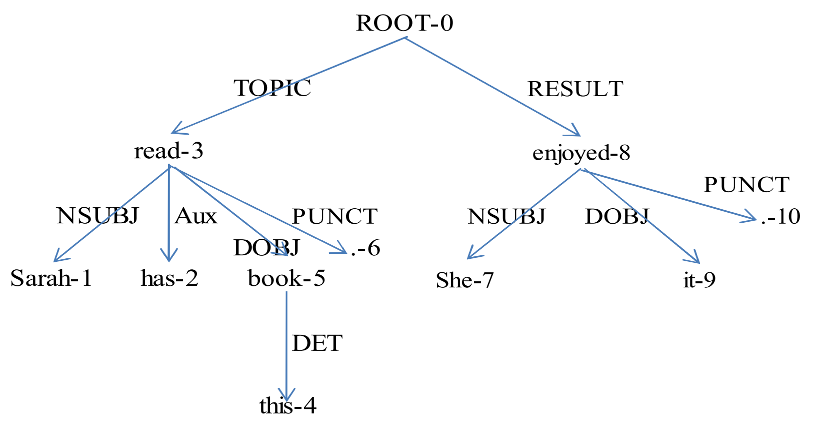


Figure 5.5. The typed-dependency tree with discourse dependency types for ‘Sarah has read this book. She enjoyed it.’

The typed-dependency tree above is equivalent to the functional structure below.

```
|          |PRED      'read<NSUBJ, DOBJ>'                          | |
|          |NSUBJ    |PRED      'Sarah'              |              | |
|          |         |PERSON    3RD                  |              | |
|          |         |NUMBER    SINGULAR             |              | |
|          |         |GENDER    FEMININE             |              | |
|          |                                                        | |
|          |DOBJ     |PRED      'book'                 |            | |
|          |         |PERSON    3RD                    |            | |
|          |         |NUMBER    SINGULAR               |            | |
|          |         |GENDER    NEUTRAL                |            | |
|          |         |DET      |FORM   'the'       |  |            | |
|          |         |         |TYPE   DEFINITE    |  |            | |
|          |                                                        | |
|          |AUX      |FORM      'has'      |                        | |
|          |         |TENSE     PRESENT    |                        | |
|          |         |ASPECT    PERFECT    |                        | |
|          |                                                        | |
|          |PUNCT    |FORM      '.'                |               | |
|TOPIC     |         |STMT-TYPE DECLARATIVE        |               | |
|                                                                    |
|          |PRED      'enjoy<NSUBJ, DOBJ>'                         | |
|          |NSUBJ    |PRED      'PRO'                |             | |
|          |         |PERSON    3RD                  |             | |
|          |         |NUMBER    SINGULAR             |             | |
|          |         |GENDER    FEMININE             |             | |
|          |         |CASE      NOMINATIVE           |             | |
|          |         |FORM      'she'                |             | |
|          |                                                        | |
|          |DOBJ     |PRED      'PRO'                |             | |
|          |         |PERSON    3RD                  |             | |
|          |         |NUMBER    SINGULAR             |             | |
|          |         |GENDER    NEUTRAL              |             | |
|          |         |CASE      ACCUSATIVE           |             | |
|          |         |FORM      'it'                 |             | |
|          |                                                        | |
|          |TENSE     PAST                                          | |
|          |                                                        | |
|          |PUNCT    |FORM      '.'                  |             | |
|RESULT    |         |STMT-TYPE DECLARATIVE          |             | |
```

Figure 5.6. The functional structure for ‘Sarah has read the book.   She enjoyed it.’

This example shows that representing discourse structure as a typed-dependency tree enables us to illustrate the function of each sentence in a discourse, just like the function of each word in a sentence. However, the specification of discourse-structure representation and the definitions of possible discourse dependency types are beyond the scope of this dissertation and I leave this topic for further research.[21]

In terms of the definition for the dependency type "root," not all of Mel'čuk's original criteria for SSyntRels seem to apply in this case. In terms of Criterion A (Section 2.4.3.1), the node "root" and the verb do not seem to constitute a prosodic unit. However, considering that the node "root" is a discourse element rather than a syntactic one, we can safely argue that the node "root" is not applicable to the criteria that concern syntactic, morphological, and semantic dependencies.

5.3.2 Aux - auxiliaries

This dependency type is used for cases where the dependency head is the main predicate of a clause and the dependent is a modal auxiliary or a progressive 'be.' This dependency has two subcategories: auxpass (passive auxiliary) and cop (copula). As previously mentioned, the category 'aux' is also used in the parsed output. If a sentence has more than one auxiliary for a verb, then each auxiliary depends on the verb.

[21] The sentences I use as examples in this dissertation are actually *one-sentence texts*, which are a type of text that contains only one sentence. They are analogous to *one-word sentences*, i.e., a type of sentence that contains only one word. Studying one-word sentences provides interesting insights into their function, structure, and use; however, not into more-than-one-word sentences. By analogy, studying one-sentence texts can give us interesting insights into the function, structure, and use of them as we have already seen especially in syntax, but not into more-than-one-sentence texts.

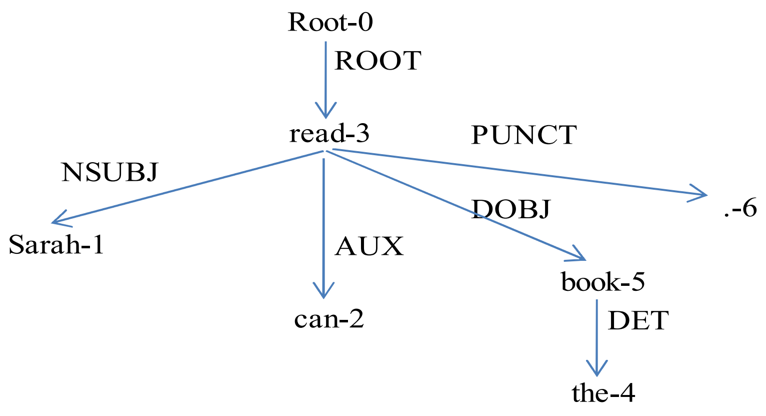


Figure 5.7. The typed-dependency tree for ‘Sarah can read the book.’

The functional structure that is equivalent to the typed-dependency tree above is as follows; the modal auxiliary ‘can’ adds the modal meaning ‘possible’[22] to the sentence.

[22] This study follows Palmer’s (2001) categorization of modality, in which modal meanings are aligned according to two major axes: *propositional modality* and *event modality*. Propositional modality is divided into *epistemic modality* and *evidential modality*, which “are concerned with the speaker’s attitude to the truth-value or factual status of the proposition” (Ibid 2001, p.7), while event modality is divided into *deontic modality* and *dynamic modality*, which “refer to events that are not actualized” (Ibid 2001, p.7). As for the sentence ‘Sarah can read this book.,” the modal meaning of this sentence is not unique; it can mean either that Sarah has the ability to read the book (dynamic meaning), or that Sarah is allowed to read the book (deontic meaning). In this study, the term “possible” is used to cover both of these modality meanings.

```
          | PRED     'read<NSUBJ,DOBJ>'                                 | |
          | NSUBJ    | PRED 'Sarah'            |                        | |
          |          | PERSON    3RD           |                        | |
          |          | NUMBER    SINGULAR      |                        | |
          |          | GENDER    FEMININE      |                        | |
          |                                                           | |
          | DOBJ     | PRED 'book'                                 |  | |
          |          | PERSON    3RD                               |  | |
          |          | NUMBER    SINGULAR                          |  | |
          |          | GENDER    NEUTRAL                           |  | |
          |          | DET       | FORM       'the'        |       |  | |
          |          |           | TYPE       DEFINITE     |       |  | |
          |                                                           | |
          | AUX      | FORM       'can'        |                      | |
          |          | TENSE      PRESENT      |                      | |
          |          | MODALITY   POSSIBILITY  |                      | |
          |                                                           | |
          | PUCNT    | FORM       '.'          |                      | |
| ROOT    |          | STMT-TYPE  DECLARATIVE  |                      | |
```

Figure 5.8. The functional structure for ‘Sarah can read the book.’

In the figure below, the copula depends on the present participle of a verb, and this construction indicates the present progressive aspect.

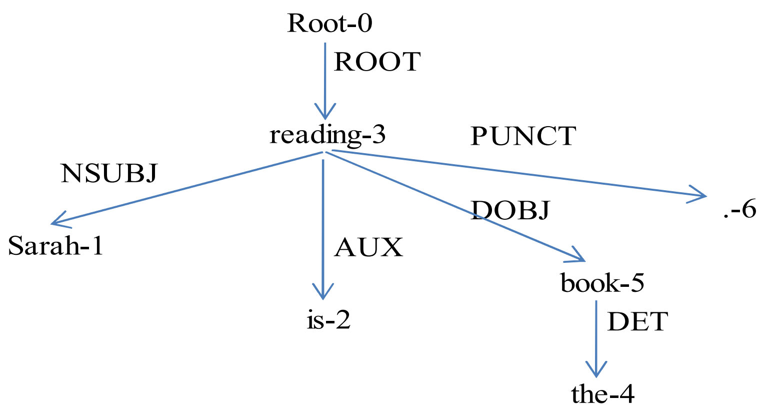


Figure 5.9. The typed-dependency tree for ‘Sarah is reading the book.’

The figure below is the functional structure that is equivalent to the typed-dependency tree above.

Notice that the auxiliary ‘is’ specifies the tense and aspect values of this sentence.

$$\left[\begin{array}{ll} \text{ROOT} & \left[\begin{array}{ll} \text{PRED} & \text{'read<NSUBJ,DOBJ>'} \\ \text{NSUBJ} & \left[\begin{array}{ll} \text{PRED} & \text{'Sarah'} \\ \text{PERSON} & \text{3RD} \\ \text{NUMBER} & \text{SINGULAR} \\ \text{GENDER} & \text{FEMININE} \end{array}\right] \\ \text{DOBJ} & \left[\begin{array}{ll} \text{PRED} & \text{'book'} \\ \text{PERSON} & \text{3RD} \\ \text{NUMBER} & \text{SINGULAR} \\ \text{GENDER} & \text{NEUTRAL} \\ \text{DET} & \left[\begin{array}{ll} \text{FORM} & \text{'the'} \\ \text{TYPE} & \text{DEFINITE} \end{array}\right] \end{array}\right] \\ \text{AUX} & \left[\begin{array}{ll} \text{FORM} & \text{'is'} \\ \text{TENSE} & \text{PRESENT} \\ \text{ASPECT} & \text{PROGRESSIVE} \end{array}\right] \\ \text{PUNCT} & \left[\begin{array}{ll} \text{FORM} & \text{'.'} \\ \text{STMT-TYPE} & \text{DECLARATIVE} \end{array}\right] \end{array}\right] \end{array}\right]$$

Figure 5.10. The functional structure for ‘Sarah is reading the book.’

A single sentence can have more than one auxiliary, as shown in the typed-dependency tree in the figure below.

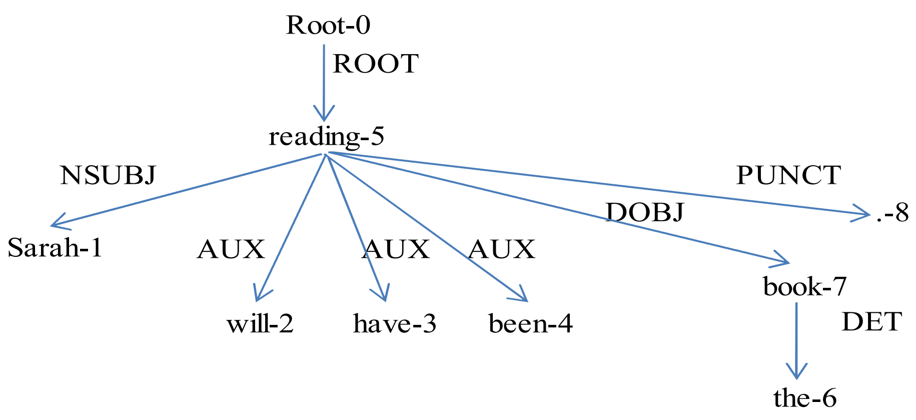


Figure 5.11. The typed-dependency tree for ‘Sarah will have been reading the book.’

Each of the different auxiliaries carries different types of information, which are unified at the root level of the functional structure. In the functional structure below, ‘will’ and ‘have’ indicate the future time[23] and the perfective aspect of the sentence respectively, and ‘been’ and the present participle ‘reading’ indicate the progressive aspect.[24]

[23] Some grammarians (e.g., Quirk, Greenbaum, Leech, & Svartvik 1985, p.176) assume that English does not have future tense as a formal category. The background of this assumption is that they consider the tense in general as a category expressed by the morphology of a verb. In this dissertation, I do not share this assumption with these grammarians, and consider the tense as a category expressed not only by the morphology of verbs, but also by various constructions which may involve more than one word.

[24] The attribute AUX can have a set of values; the curly brackets around the values of the attribute AUX in Figure 5.12 indicate that they are the elements of the value set for the attribute AUX. In this study, curly brackets around the value(s) of an attribute indicate that the attribute can have a set of values. In addition, m-structure representation of auxiliaries (Sadler & Spencer 2004) is not used here, in order to highlight the equivalence of the functional-structure representation and the typed-dependency tree representation for a sentence. See Section 3.3.

$$\left[\begin{array}{ll}
\text{ROOT} & \left[\begin{array}{ll}
\text{PRED} & \text{'read<NSUBJ,DOBJ>'} \\
\text{NSUBJ} & \left[\begin{array}{ll}
\text{PRED} & \text{'Sarah'} \\
\text{PERSON} & \text{3RD} \\
\text{NUMBER} & \text{SINGULAR} \\
\text{GENDER} & \text{FEMININE}
\end{array}\right] \\
\text{DOBJ} & \left[\begin{array}{ll}
\text{PRED} & \text{'book'} \\
\text{PERSON} & \text{3RD} \\
\text{NUMBER} & \text{SINGULAR} \\
\text{GENDER} & \text{NEUTRAL} \\
\text{DET} & \left[\begin{array}{ll}
\text{FORM} & \text{'the'} \\
\text{TYPE} & \text{DEFINITE}
\end{array}\right]
\end{array}\right] \\
\text{AUX} & \left\{\begin{array}{l}
\left[\begin{array}{ll}
\text{FORM} & \text{'will'} \\
\text{TENSE} & \text{FUTURE}
\end{array}\right] \\
\left[\begin{array}{ll}
\text{FORM} & \text{'have'} \\
\text{ASPECT} & \text{PERFECT}
\end{array}\right] \\
\left[\begin{array}{ll}
\text{FORM} & \text{'been'} \\
\text{ASPECT} & \text{PROGRESSIVE}
\end{array}\right]
\end{array}\right\} \\
\text{PUNCT} & \left[\begin{array}{ll}
\text{FORM} & \text{'.'} \\
\text{STMT-TYPE} & \text{DECLARATIVE}
\end{array}\right]
\end{array}\right]
\end{array}\right]$$

Figure 5.12. The functional structure for ‘Sarah will have been reading the book.’

In the Stanford Parser output, the auxiliaries that include to-infinitive (e.g., ‘have to,’ ‘be to,’ ‘ought to’) are analyzed as the root of the sentence, and the ‘to’ is considered an auxiliary depending on the verb, which depends on the root with the type “XCOMP.” This study follows this analysis, and the example sentences that contain to-infinitive auxiliaries will be shown in the subsection for “XCOMP.”

The dependency type “aux” follows Mel’čuk’s Criteria. First, a verb and an auxiliary (or at most two auxiliaries) clearly form a prosodic unit (e.g., ‘can take,’ ‘may have seen’) and the auxiliary always precedes the verb, following Mel’čuk’s original Criterion A. In addition, a verb and an auxiliary form a semantic unit, which follows the revised Criterion A discussed in

Section 2.4.3.1.

Second, this dependency type follows Mel'čuk's Criterion B. The verb, not the auxiliary, serves as the head of the dependency between the verb and the auxiliary. This is the case because the verb-auxiliary phrase is a form of the verb, which follows Mel'čuk's Criterion B3. For example, the phrase 'can take' represents a form of the verb 'take,' not of the auxiliary 'can.'

Third, this dependency type implies a unique semantic relationship between the governor and its dependent, and the prototypical dependent of this type is an auxiliary. Therefore, this dependency type follows the revised Criteria C1 and C2 proposed in Section 2.4.3.3.

Auxpass - passive auxiliaries

The dependency type "auxpass" is used when the dependency head is a verb and the dependent is an auxiliary for sentences in the passive voice.

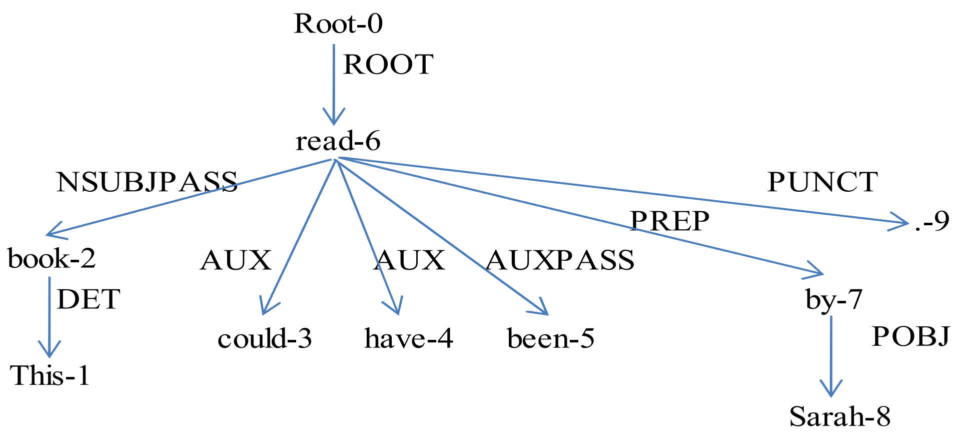


Figure 5.13. The typed-dependency tree for 'This book could have been read by Sarah.'[25]

[25] The dependency type "prep" is used for prepositional phrases, indicating the agent of the action. The dependency type "agent" is not used here. The dependency type "agent" will be discussed later in this section.

The figure below is the functional structure that is equivalent to the typed-dependency tree above. Notice that the AUXPASS attribute has the attribute VOICE whose value is PASSIVE.

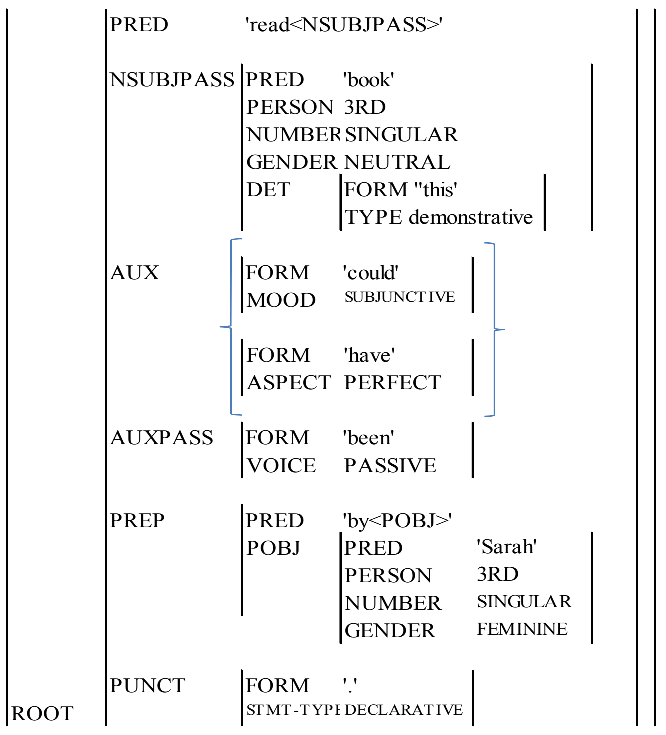


Figure 5.14. The functional structure for ‘This book could have been read by Sarah.’

The dependency type “auxpass” follows Mel’čuk’s Criteria for the same reasons as the ones presented for the type “aux.” First, a verb and a passive auxiliary clearly form a prosodic unit (e.g., ‘was taken’ and ‘may have been taken’), and the auxiliary precedes the verb. Therefore, this type follows Mel’čuk’s Criterion A.

Second, this dependency type follows Mel’čuk’s Criterion B. The verb, not the passive auxiliary, functions as the head of the dependency because the verb-auxiliary phrase represents a form of the verb. For example, the phrase ‘was taken’ represents a form of the verb ‘take,’ not the passive auxiliary 'was.'

Third, this dependency type implies a unique semantic relationship between the governor and its dependent, and the prototypical dependent of this type is a passive auxiliary. Therefore, this dependency type follows the revised Criteria C1 and C2 proposed in Section 2.4.3.3.

Cop - copula

This dependency type is used for cases where the dependent is a form of the verb 'be.' This type is used for sentences in which an adjective or a noun serves as the root of the sentence, and the verb 'be' is the auxiliary modifier of the root[26]. In Stanford Dependencies, the nominal or adjectival complement is considered as the root of a clause and the copula is an auxiliary modifier, and the prepositional complement as the dependent on the copula. This analysis distinguishes the existential usage of the copula 'be' from the other usages. This distinction is reminiscent of Halliday's (1967, p.66) classification of 'be.' He argues that the copula 'be' is not different from other verbs except for the fact that it can be categorized into more than one class. The three classes he assigns to the copula 'be' are as follows.

[26] There are two different analyses for copular constructions in LFG (Butt et al. 1999; Dalrymple, Dyvik, & King 2004). One is an *open complement* analysis, in which the complement of a copula is an XCOMP whose subject is also the subject of the copula. The other is a *closed complement* analysis, in which the complement of a copula is PREDLINK which does not have its subject. Open complement analysis is appropriate for constructions in which the complement of a copula is a verbal predicate (e.g., Sarah is reading a book), while closed complement analysis is appropriate for constructions in which the complement of a copula is a nominal predicate (e.g., Sarah is a student), adjectival predicate (e.g., Sarah is intelligent), or a prepositional predicate (e.g., Sarah is in the classroom), because these non-verbal complements do not need to have subject arguments (Butt et al. 1999, p.70). Both analyses presuppose that the copula "be" is always the root of a clause.

(5.3)

Class 0 be means “can be characterized as, has the attribute of being”

Class 1 be means “exists, happens, is found or located”

Class 2 be means “identifies or is identifiable as, can be equated with”

Halliday (1967, p.67) states that *Class 0* be is *intensive*, while *Class 2* be is *extensive effective*: intensive ‘be’ clauses answer questions about the attribute or quality of the subject (e.g., ‘What is Sarah?’ ‘She is a student’); extensive effective ‘be’ clauses answer questions about the identity of the subject (e.g., ‘Who is Sarah?’ ‘She is David’s wife.’) *Class 1* ‘be’ is *descriptive*, and used with a locative adjunct (e.g., ‘Where is Sarah?’ ‘She is in her room.’) (Halliday 1967, p.71).

Although the Stanford Parser has an option for the output format in which the copula is the root of the sentence (de Marneffe & Manning 2012, p.17), this study does not use this option, because the default option of the Stanford Parser reflects the syntactic difference between *Class 1* be and *Class 0* and *Class 2* be; *Class 1* be requires a locative adjunct, while *Class 0* and *Class 2* be do not.

The figure below is the typed-dependency tree for ‘Sarah is a student.’ The complement of ‘is’ in this sentence is a noun.

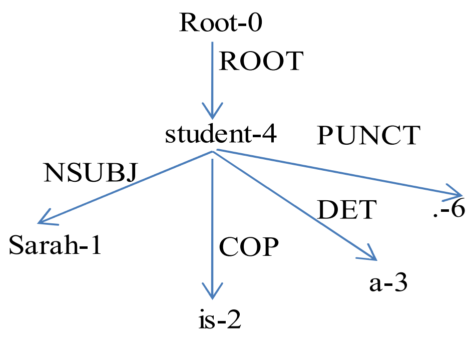


Figure 5.15. The typed-dependency tree for ‘Sarah is a student.’

The typed-dependency tree above is equivalent to the functional structure below.

```
     | PRED      'student<NSUBJ>'                  | |
     | PERSON    3RD                               | |
     | NUMBER    SINGULAR                          | |
     | GENDER    MASCULAR/FEMININE                 | |
     | DET      |FORM       'a'           |        | |
     |          |TYPE       INDEFINITE    |        | |
     |                                             | |
     | NSUBJ    |PRED       'Sarah'       |        | |
     |          |PERSON     3RD           |        | |
     |          |NUMBER     SINGULAR      |        | |
     |          |GENDER     FEMININE      |        | |
     |                                             | |
     | COP      |FORM       'is'          |        | |
     |          |TENSE      PRESENT       |        | |
     |                                             | |
     | PUNCT    |FORM       '.'           |        | |
ROOT |          |STMT-TYPE  DECLARATIVE   |        | |
```

Figure 5.16. The functional structure for ‘Sarah is a student.’[27]

The figure below is the typed-dependency tree for ‘Sarah is intelligent.’ The root of this sentence is an adjective.

[27] This functional-structure representation raises a problem of whether a noun can have a nominal subject. The solution for this problem is to treat the noun modified by a copular auxiliary as subcategorizing for its subject. Chafe (1970, p.201-202) states that predicative nouns are “stative verbs derived from nouns through a derivational unit that was labeled *predicativizer*”. Predicativizer in this context can be understood as a copular auxiliary. Being stative verbs necessitates predicate nouns to have their subjects. He also argues that the semantics of predicate nouns are such that the subject is a “proper subset of the objects specified by the root of the predicate noun” (Ibid 1970, p.202). According to this view, ‘Sarah’ in the sentence ‘Sarah is a student’ is a proper subset of all students.

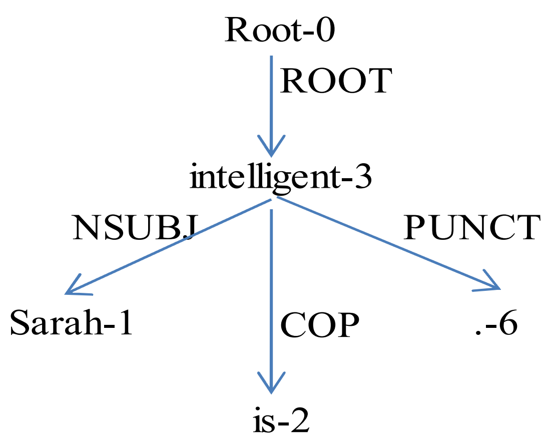


Figure 5.17. The typed-dependency tree for ‘Sarah is intelligent.’

The figure below is the functional structure that is equivalent to the typed-dependency tree above.

	PRED	'intelligent<NSUBJ>'	
	NSUBJ	PRED	'Sarah'
		PERSON	3RD
		NUMBER	SINGULAR
		GENDER	FEMININE
	COP	FORM	'is'
		TENSE	PRESENT
	PUNCT	FORM	'.'
ROOT		STMT-TYPE	DECLARATIVE

Figure 5.18. The functional structure for ‘Sarah is intelligent.’

The dependency type “cop” follows Mel’čuk’s Criteria for SSyntRel. First, a noun or an adjective and a copula clearly form a prosodic unit (e.g., ‘is intelligent’), and the copula precedes the verb. Therefore, this type follows Mel’čuk’s Criterion A. In addition, a noun or an adjective and a copula can form a semantic unit, which follows the revised Criterion A proposed in Section 2.4.3.1.

Second, this dependency type follows Mel'čuk's Criterion B3 because a copula-noun phrase or a copula-adjective phrase represents a form the noun or the adjective. For example, the phrase 'is intelligent' is considered a form of the adjective 'intelligent,' not the auxiliary 'is.'

Third, this dependency type implies a unique semantic relationship between the governor and its dependent, and the prototypical dependent of this type is a copula. Therefore, this dependency type follows the revised Criteria C1 and C2 proposed in Section 2.4.3.3.

5.3.3 Arg - arguments

This dependency type describes cases in which the dependency head is an argument-taking element (verbs or prepositions) and the dependent is its argument. This dependency type has three subtypes ("agent," "comp," and "subj"), and two of them ("subj" and "comp") have sub-subtypes. Each type is discussed in turn below.

Agent

This dependency type is a subtype of "arg" and describes cases in which the dependency head is a verb in the passive voice and the dependent is the agent of the verb. This dependency type is realized in the Stanford Parser when the parsing option is not set to the default option (basicDependencies); see Section 5.4 for the output options of the Stanford Parser. The preposition 'by' is not realized in the parsed output.

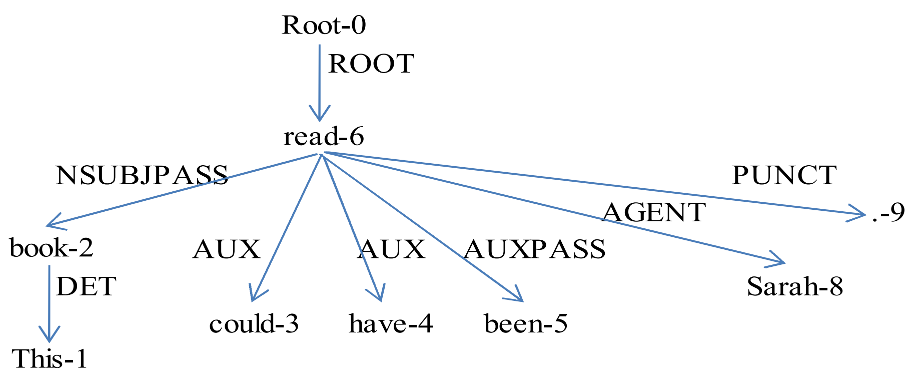


Figure 5.19. The typed-dependency tree for ‘This book could have been read by Sarah’ with the dependency type “agent”

The typed-dependency tree above is equivalent to the functional structure below.

```
         PRED       'read<NSUBJPASS>'
         NSUBJPASS  PRED     'book'
                    PERSON   3RD
                    NUMBER   SINGULAR
                    GENDER   NEUTRAL
                    DET      FORM "this'
                             TYPE demonstrative

         AUX        { FORM     'could'
                      MOOD     SUBJUNCTIVE

                      FORM     'have'
                      ASPECT   PERFECT }

         AUXPASS    FORM     'been'
                    VOICE    PASSIVE

         AGENT      PRED     'Sarah'
                    PERSON   3RD
                    NUMBER   SINGULAR
                    GENDER   FEMININE

         PUNCT      FORM     '.'
ROOT                STMT-TYPE  DECLARATIVE
```

Figure 5.20. The functional structure for ‘This book could have been read by Sarah.’ with the

dependency type “agent.”

Comp - complement

This dependency type is a subtype of “arg” and describes cases in which the dependent is the complement of the head of a clause. This dependency type is further subcategorized into eleven subtypes, discussed in turn below.

Acomp - adjectival complement

The dependency type “acomp” is a subtype of “comp” used for cases where the dependent is an adjective. The verbs that take adjectives as their complement are divided into the following four subcategories (Nakano 1998, p.33-34): (1) verbs that express a change of state (e.g., become, fall, get, turn); (2) perception verbs (e.g., look, sound, smell, taste, feel); (3) stative verbs (e.g., remain, stay, keep); and (4) *contingent* verbs (e.g., seem, appear). Contingent verbs express an accidental, not essential, state of an entity.

Change-of-state verbs

The figure below presents an example typed-dependency tree in which an adjective depends on a change-of-state verb, with the dependency type “acomp.”

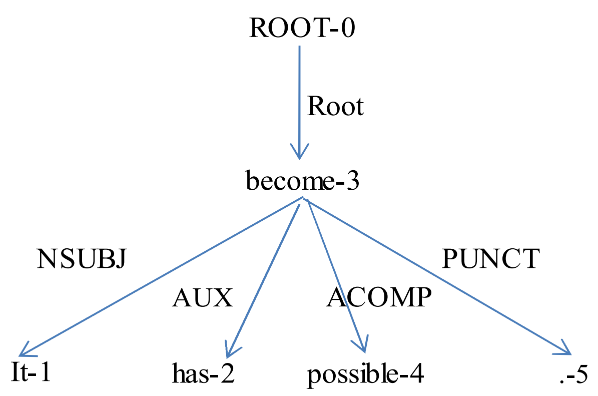


Figure 5.21. The typed-dependency tree for ‘It has become possible.’

The typed-dependency tree above is equivalent to the following functional structure. Notice that the dependency type “acomp” is subcategorized for by the main predicate ‘become.’

	PRED	'become<NSUBJ, ACOMP>'	
	NSUBJ	PRED	'PRO'
		FORM	it
		PERSON	3RD
		NUMBER	SINGULAR
		GENDER	NEUTRAL
	AUX	FORM	'has'
		TENSE	PRESENT
		ASPECT	PERFECT
	ACOMP	PRED	'possible'
	PUNCT	FORM	'.'
ROOT		STMT-TYPE	DECLARATIVE

Figure 5.22. The functional structure for ‘It has become possible.’

The typed-dependency tree below is another example in which an adjective depends on a change-of-state verb with the dependency type “acomp.”

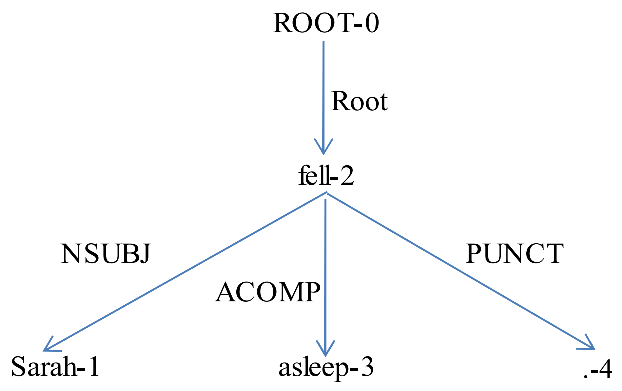


Figure 5.23. The typed-dependency tree for 'Sarah fell asleep.'

The typed-dependency tree above is equivalent to the functional structure shown below. The dependency type "acomp" is subcategorized for by the verb 'fell.'

```
|      |PRED     'fall<NSUBJ, ACOMP>'        ||
|      |NSUBJ   |PRED       'Sarah'       |  ||
|      |        |PERSON     3RD           |  ||
|      |        |NUMBER     SINGULAR      |  ||
|      |        |GENDER     FEMININE      |  ||
|      |TENSE    PAST                        ||
|      |ACOMP   |PRED       'asleep'      |  ||
|      |                                     ||
|      |PUNCT   |FORM       '.'           |  ||
|ROOT  |        |STMT-TYPE  DECLARATIVE   |  ||
```

Figure 5.24. The functional structure for 'Sarah fell asleep.'

Perception verbs

The figure below presents an example typed-dependency tree in which an adjective depends on a perception verb.

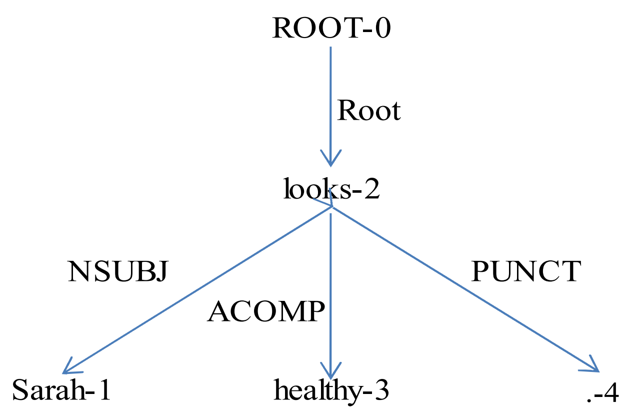


Figure 5.25. The typed-dependency tree for ‘Sarah looks healthy.’

The typed-dependency tree above is equivalent to the functional structure shown below. The dependency type “acomp” is subcategorized for by the verb ‘looks.’

	PRED	'look<NSUBJ, ACOMP>'	
	NSUBJ	PRED	'Sarah'
		PERSON	3RD
		NUMBER	SINGULAR
		GENDER	FEMININE
	TENSE	PRESENT	
	ACOMP	PRED	'healthy'
	PUNCT	FORM	'.'
ROOT		STMT-TYPE	DECLARATIVE

Figure 5.26. The functional structure for ‘Sarah looks healthy.’

The figure below is another example in which an adjective depends on a perception verb.

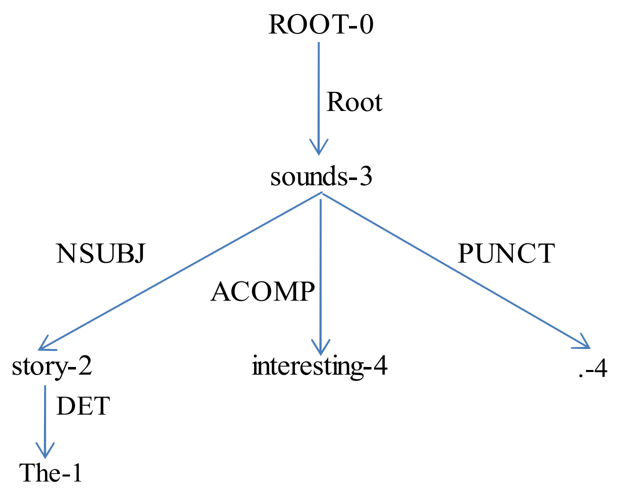


Figure 5.27. The typed-dependency tree for ‘The story sounds interesting.’

The typed-dependency tree above is equivalent to the functional structure below.

```
|          |PRED     'sound<NSUBJ, ACOMP>'                            | |
|          |NSUBJ   |PRED       'story'                            |  | |
|          |        |PERSON     3RD                                |  | |
|          |        |NUMBER     SINGULAR                           |  | |
|          |        |GENDER     NEUTRAL                            |  | |
|          |        |DET       |FORM        'the'         |        |  | |
|          |        |          |TYPE        DEFINITE      |        |  | |
|          |TENSE    PRESENT                                          | |
|          |ACOMP   |PRED       'interesting'  |                      | |
|          |                                                          | |
|          |PUNCT   |FORM       '.'            |                      | |
|ROOT      |        |STMT-TYPE  DECLARATIVE    |                      | |
```

Figure 5.28. The functional structure for ‘The story sounds interesting.’

Stative verbs

The figure below presents an example typed-dependency tree in which an adjective depends on a stative verb.

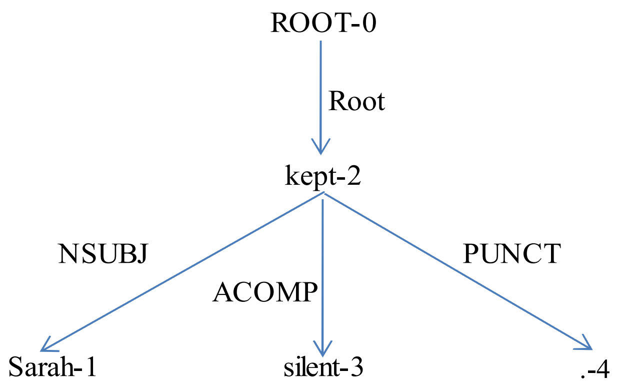


Figure 5.29. The typed-dependency tree for ‘Sarah kept silent.’

The typed-dependency tree above is equivalent to the functional structure below.

```
|        |PRED     'keep<NSUBJ, ACOMP>'          ||
|        |NSUBJ   |PRED       'Sarah'       |    ||
|        |        |PERSON     3RD           |    ||
|        |        |NUMBER     SINGULAR      |    ||
|        |        |GENDER     FEMININE      |    ||
|        |TENSE    PAST                          ||
|        |ACOMP   |PRED       'silent'      |    ||
|        |                                       ||
|        |PUNCT   |FORM       '.'           |    ||
|ROOT    |        |STMT-TYPE  DECLARATIVE   |    ||
```

Figure 5.30. The functional structure for ‘Sarah kept silent.’

The figure below is another example in which an adjective depends on a stative verb. The adjective ‘still’ depends on the stative verb ‘stood.’

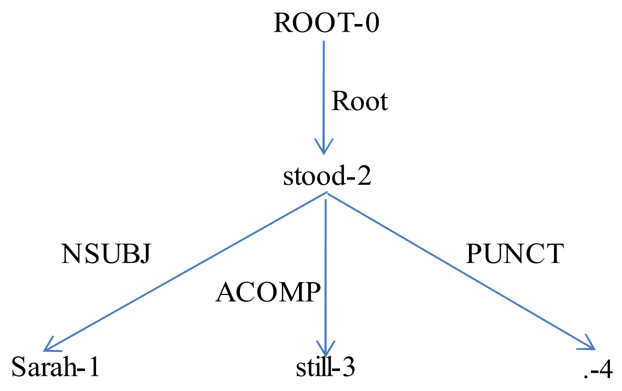


Figure 5.31. The typed-dependency tree for ‘Sarah stood still.’

The typed-dependency tree above is equivalent to the functional structure below.

```
|      |PRED    'stand<NSUBJ, ACOMP>'   ||
|      |NSUBJ   |PRED      'Sarah'    | ||
|      |        |PERSON    3RD        | ||
|      |        |NUMBER    SINGULAR   | ||
|      |        |GENDER    FEMININE   | ||
|      |TENSE   PAST                    ||
|      |ACOMP   |PRED      'still'    | ||
|      |                                ||
|      |PUNCT   |FORM      '.'        | ||
|ROOT  |        |STMT-TYPE DECLARATIVE| ||
```

Figure 5.32. The functional structure for ‘Sarah stood still.’

Contingent verbs

The figure below presents an example of a typed-dependency tree which contains a contingent verb. The adjective ‘healthy’ depends on the contingent verb ‘seems.’

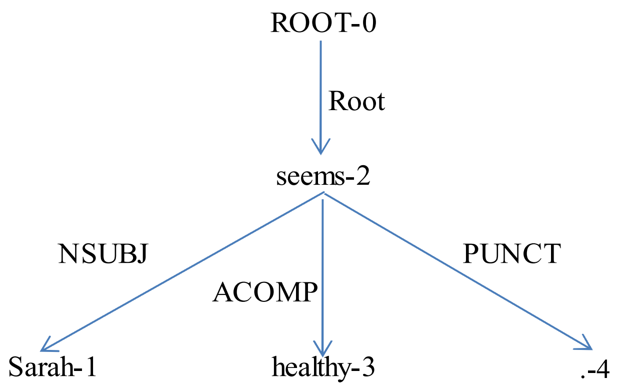


Figure 5.33. The typed-dependency tree for ‘Sarah seems healthy.’

The typed-dependency tree above is equivalent to the functional structure below.

$$\left[\begin{array}{ll} & \left[\begin{array}{ll} \text{PRED} & \text{'seem<NSUBJ, ACOMP>'} \\ \text{NSUBJ} & \left[\begin{array}{ll} \text{PRED} & \text{'Sarah'} \\ \text{PERSON} & \text{3RD} \\ \text{NUMBER} & \text{SINGULAR} \\ \text{GENDER} & \text{FEMININE} \end{array}\right] \\ \text{TENSE} & \text{PRESENT} \\ \text{ACOMP} & \left[\begin{array}{ll} \text{PRED} & \text{'healthy'} \end{array}\right] \\ \text{PUNCT} & \left[\begin{array}{ll} \text{FORM} & \text{'.'} \\ \text{STMT-TYPE} & \text{DECLARATIVE} \end{array}\right] \end{array}\right] \\ \text{ROOT} & \end{array}\right]$$

Figure 5.34. The functional structure for ‘Sarah seems healthy.’

The figure below is another example of a typed-dependency tree which contains a contingency verb. In this case, the adjective ‘healthy’ depends on the contingency verb ‘appears.’

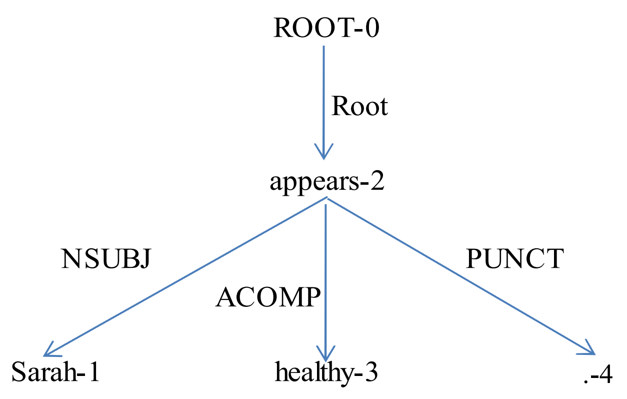


Figure 5.35. The typed-dependency tree for ‘Sarah appears healthy.’

The typed-dependency tree above is equivalent to the functional structure below.

```
|          |PRED     'appear<NSUBJ, ACOMP>'    | |
|          |NSUBJ   |PRED       'Sarah'      |  | |
|          |        |PERSON     3RD          |  | |
|          |        |NUMBER     SINGULAR     |  | |
|          |        |GENDER     FEMININE     |  | |
|          |TENSE    PRESENT                    | |
|          |ACOMP   |PRED       'healthy'    |  | |
|          |                                    | |
|          |PUNCT   |FORM       '.'          |  | |
|ROOT      |        |STMT-TYPE  DECLARATIVE  |  | |
```

Figure 5.36. The functional structure for ‘Sarah appears healthy.’

The dependency type “acomp” follows Mel’čuk’s Criteria for SSyntRel. First, a verb and its adjectival complement clearly form a prosodic unit (e.g., ‘looks intelligent,’ ‘kept silent,’ etc.), and the verb precedes its adjectival complement. Therefore, this type follows Mel’čuk’s Criterion A. In addition, a verb and its adjectival complement form a semantic unit, which follows the revised Criterion A proposed in Section 2.4.3.1.

Second, this dependency type follows Criterion B3 because the verb-complement phrase represents a form, or denotation, of the verb. For example, the phrase ‘looks beautiful’ denotes the verb ‘looks,’ not the adjective ‘beautiful.’

Third, this dependency type implies a unique semantic relationship between the governor and its dependent, and the prototypical dependent of this dependency type is an adjective. Therefore, this dependency type follows the revised Criteria C1 and C2 proposed in Section 2.4.3.3.

Attr – attributive

The dependency type “attr” is a subtype of “comp” and represents cases where the dependency head is a copula and the dependent is an interrogative pronoun. The figure below is an example typed-dependency tree in which the interrogative pronoun ‘what’ depends on the copula ‘is’ with the type “attr.”

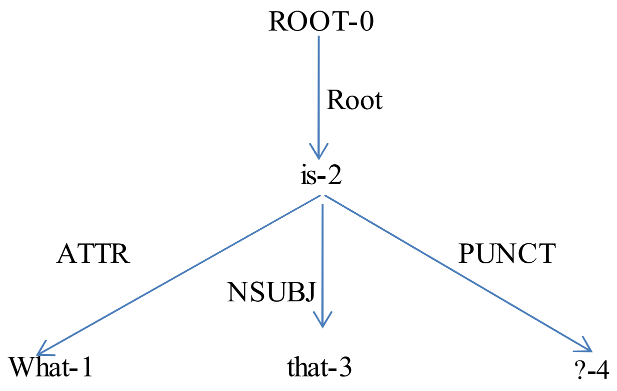


Figure 5.37. The typed-dependency tree for ‘What is that?’

The typed-dependency tree above is equivalent to the functional structure below. The pronoun ‘what’ is typed as INTERROGATIVE, and the pronoun ‘that’ is typed as DEMONSTRATIVE.

The question mark adds to the sentence the value of the statement-type attribute.

```
     | PRED    'be<NSUBJ>'                          | |
     | ATTR   | PRED       'PRO'               |    | |
     |        | FORM       What                |    | |
     |        | TYPE       INTERROGATIVE       |    | |
     |                                              | |
     | NSUBJ  | PRED       'PRO'               |    | |
     |        | FORM       that                |    | |
     |        | TYPE       DEMONSTRATIVE       |    | |
     |                                              | |
     | PUNCT  | FORM       '?'                 |    | |
|ROOT  |        | STMT-TYPE  INTERROGATIVE       |    | |
```

Figure 5.38. The functional structure for ‘What is that?’

The dependency type “attr” seems to be an ad hoc type because it does not follow all of Mel’čuk’s criteria. On one hand, a copula and an interrogative pronoun form a prosodic unit (e.g., ‘Where is,’ ‘How are,’ etc.), thus following Mel’čuk’s Criterion A. However, the copula, not the interrogative pronoun, serves as the head of the dependency. This contrasts with the analysis for the dependency type “cop,” in which the adjective or noun functions as the head while the copula is the dependent. There is no well-established reason to treat the copula as the head if it exists in a dependency relationship with an interrogative pronoun. Therefore, this dependency type seems to be ad hoc, and a better analysis would interpret the interrogative pronoun as the head and the copula as the dependent.

Ccomp - Clausal Complement with Internal Subject

Complm – complementizer

The dependency type “ccomp” is a subtype of “comp” and represents cases where the dependent clause has an internal subject. The dependency type “complm” is another subtype of “comp” used to represent cases where the dependent is the word introducing the dependent clause, such as ‘that’ in the example sentence below.

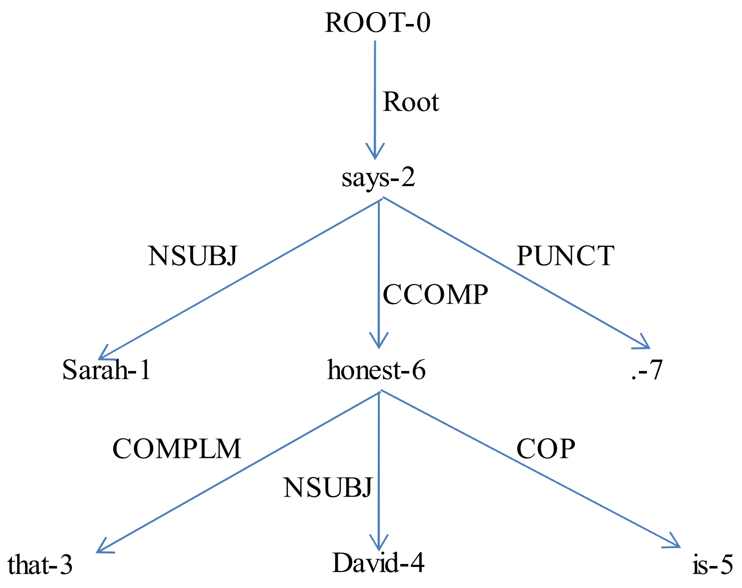


Figure 5.39. The typed-dependency tree for ‘Sarah says that David is honest.’

The typed-dependency tree above is equivalent to the functional structure below. The attribute COMPLM has a local functional structure, which has an attribute FORM whose value is ‘that.’

```
|      |PRED    'say<NSUBJ,CCOMP>'                                   ||
|      |NSUBJ  |PRED       'Sarah'    |                              ||
|      |       |PERSON     3RD        |                              ||
|      |       |NUMBER     SINGULAR   |                              ||
|      |       |GENDER     FEMININE   |                              ||
|      |                                                             ||
|      |CCOMP  |PRED       'honest'                          |       ||
|      |       |NSUBJ     |PRED       'David'     |          |       ||
|      |       |          |PERSON     3RD         |          |       ||
|      |       |          |NUMBER     SINGULAR    |          |       ||
|      |       |          |GENDER     MASCULINE   |          |       ||
|      |       |                                             |       ||
|      |       |COP       |FORM       'is'        |          |       ||
|      |       |          |TENSE      PRESENT     |          |       ||
|      |       |                                             |       ||
|      |       |COMPLM    |FORM       'that'      |          |       ||
|      |                                                             ||
|      |PUNCT  |FORM       '.'          |                            ||
|ROOT  |       |STMT-TYPE  DECLARATIVE  |                            ||
```

Figure 5.40. The functional structure for ‘Sarah says that David is honest.’

The typed-dependency tree below is an example which contains the complementizer ‘whether.’

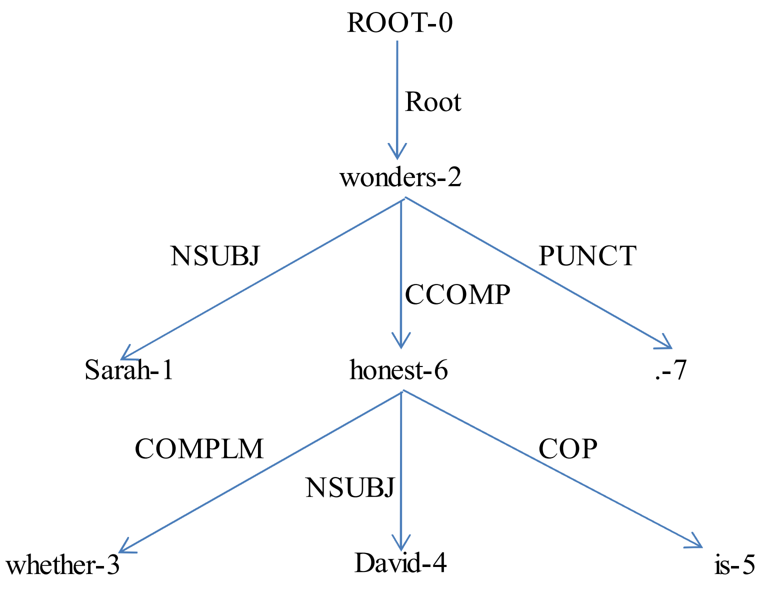


Figure 5.41. The typed-dependency tree for ‘Sarah wonders whether David is honest.’

The typed-dependency tree above is equivalent to the functional structure below. Notice that the word ‘whether’ is the value of the attribute FORM in the local functional structure, and this local functional structure is the value of the attribute COMPLM.

```
|      |PRED    'wonder<NSUBJ,CCOMP>'                               ||
|      |NSUBJ  |PRED     'Sarah'    |                               ||
|      |       |PERSON   3RD        |                               ||
|      |       |NUMBER   SINGULAR   |                               ||
|      |       |GENDER   FEMININE   |                               ||
|      |                                                            ||
|      |CCOMP  |PRED     'honest'                          |        ||
|      |       |NSUBJ   |PRED      'David'     |           |        ||
|      |       |        |PERSON    3RD         |           |        ||
|      |       |        |NUMBER    SINGULAR    |           |        ||
|      |       |        |GENDER    MASCULINE   |           |        ||
|      |       |                                          |        ||
|      |       |COP     |FORM      'is'        |           |        ||
|      |       |        |TENSE     PRESENT     |           |        ||
|      |       |                                          |        ||
|      |       |COMPLM  |FORM      'whether'   |           |        ||
|      |                                                            ||
|      |PUNCT  |FORM       '.'          |                           ||
|ROOT  |       |STMT-TYPE  DECLARATIVE  |                           ||
```

Figure 5.42. The functional structure for ‘Sarah wonders whether David is honest.’

The dependency type “ccomp” does not follow all of Mel’čuk’s original criteria for SSyntRel. The main verb precedes the dependent clause; however, the main verb and the subordinate verb do not form a prosodic unit (e.g., ‘…says…honest’), in contrast to Mel’čuk’s original Criterion A. However, the verb of the main clause and the subordinate verb definitely form a semantic unit, thus following the revised Criterion A discussed in Section 2.4.3.1.

This dependency type follows Mel’čuk’s Criterion B because the main verb determines the passive valence of the subsequent phrase. In other words, in the sentence, the subsequent verb has the “ability to be subordinated, in a specified role, to lexemes of a certain class” (Mel’čuk & Pertsov 1987, p.80). For example, a verb taking a subordinate clause can be dependent on another verb, such as 'John believes Sarah has said that David was honest.' Therefore, the main verb and the additional verb in an internal-subject clause both follow Mel'čuk's Criterion B1.

This dependency type also follows the revised Criterion C proposed in Section 2.4.3.3 because it implies a certain kind of semantic relationship between the verbs, and this semantic relationship cannot be expressed by any other dependency type.

Xcomp - clausal complement with external subject[28]

The dependency type "xcomp" is a subtype of "comp" and represents cases where the clausal complement does not have its overt subject; in other words, the subject of the verb in the complement is external to the complement. This type of clausal complements is called 'open complement.' The term "xcomp" is an abbreviation of 'an e**x**ternal open **comp**lement.' Verbs that take a clausal complement are divided into three categories[29]: (1) intransitive subject-control verbs, such as 'try' in 'Sarah tries to go there'; (2) transitive subject-control verbs, such as 'promise' in 'Sarah promised David to go there'; (3) transitive object-control verbs, such as 'persuade' in 'Sarah persuaded David to go there.'

As for an intransitive subject-control verb, the external subject of its open complement is the

[28] This study follows de Marneffe & Manning's (2012) terminology for this dependency type, yet it must be pointed out here that it is appropriate to use the name "vcomp" rather than "xcomp" for this dependency type, because the dependent of this dependency type is a verb, and the term "xcomp" breaks the parallelism with "acomp" whose dependent is an adjective.

[29] This study does not follow the tradition of standard LFG in terms of the distinction between *functional control* and *anaphoric control* (Mohanan 1983, p.641; Dalrymple 2001, p.325). In functional control, the control equation in the lexical entry of a control verb identifies the controllee, and the control relationship is obligatory. In anaphoric control, on the other hand, the controller is optional. The assumption in this study is that we need not distinguish between them, because they both involve the subject zero pronoun of a complement.

same as its subject. An example of an intransitive subject-control verb is shown below. In this sentence, the subject of 'tried' and the subject of 'go' are both 'Sarah.' The subject of 'tried' is 'PRO,' which refers to 'Sarah' (this relationship is represented in a functional structure by the same index 'i' on 'PRO' and 'Sarah').

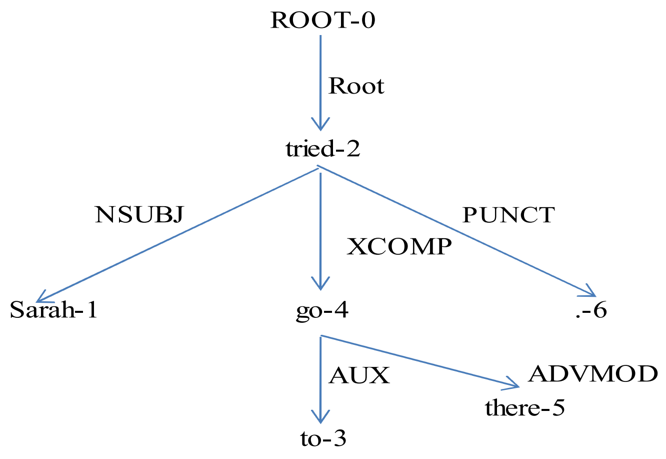


Figure 5.43. The typed-dependency tree for 'Sarah tried to go there.'

The typed-dependency tree above is equivalent to the functional structure below. Notice that the same index 'i' is assigned to the subject zero pronoun of the verb 'go' and to 'Sarah.' This indicates that the subject zero pronoun of the verb 'go' refers to 'Sarah.'

```
     | PRED      'try<NSUBJ, XCOMP>'
     | NSUBJ   | PRED      'Sarahi'
     |         | PERSON    3RD
     |         | NUMBER    SINGULAR
     |         | GENDER    FEMININE
     |
     | XCOMP   | PRED      'go<NSUBJ>'
     |         | NSUBJ   | PRED   'PROi'
     |         |         | TYPE   ZERO
     |         |
     |         | ADVMOD  [ PRED   'there' ]
     |         |
     |         | AUX     | FORM   'to'
     | TENSE     PAST
     | PUNCT   | FORM      '.'
ROOT |         | STMT-TYPE DECLARATIVE
```

Figure 5.44. The functional structure for ‘Sarah tried to go there.’

As for a transitive object-control verb, the external subject of its open complement is the same as its object. An example of transitive object-control verbs is shown below. In this sentence, the external subject of an open complement of the verb ‘persuaded’ is the direct object of ‘persuade.’

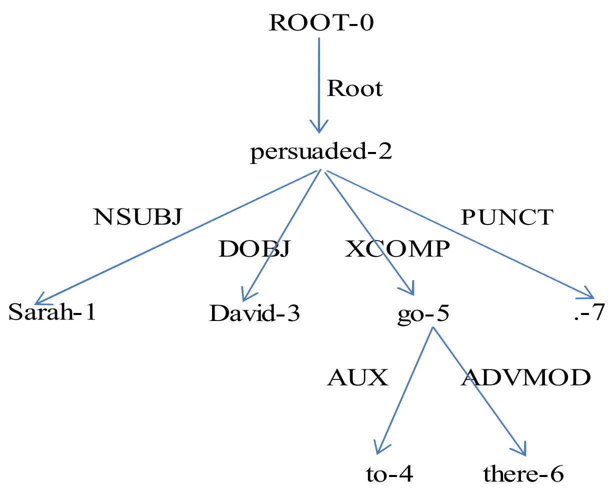


Figure 5.45. The typed-dependency tree for ‘Sarah persuaded David to go there.’

```
|          |PRED     'persuade<NSUBJ, DOBJ, XCOMP>'        | |
|          |NSUBJ   |PRED       'Sarah'        |            | |
|          |        |PERSON     3RD            |            | |
|          |        |NUMBER     SINGULAR       |            | |
|          |                                              | |
|          |DOBJ    |PRED       'David_i'      |            | |
|          |        |PERSON     3RD            |            | |
|          |        |NUMBER     SINGULAR       |            | |
|          |        |GENDER     MASCULINE      |            | |
|          |                                              | |
|          |XCOMP   |PRED       'go<NSUBJ>'      |          | |
|          |        |NSUBJ     |PRED   'PRO_i'|  |          | |
|          |        |          |TYPE   ZERO   |  |          | |
|          |        |                            |          | |
|          |        |ADVMOD    |PRED   'there'|  |          | |
|          |        |                            |          | |
|          |        |AUX       |FORM   'to'   |  |          | |
|          |TENSE    PAST                                   | |
|          |PUNCT   |FORM       '.'            |            | |
|ROOT      |        |STMT-TYPE  DECLARATIVE    |            | |
```

Figure 5.46. The functional structure for ‘Sarah persuaded David to go there.’

In the Stanford Parser output, however, the subject of the open complement of a transitive object-control verb depends on this open-complement verb with the dependency type “nsubj”, as shown in the typed-dependency tree below. This analysis allows the dependency tree to reflect the semantic relation between the one which is often considered to be the object of a transitive object-control verb (‘David’ in the example below) and the open complement verb (‘go’ in the example below).

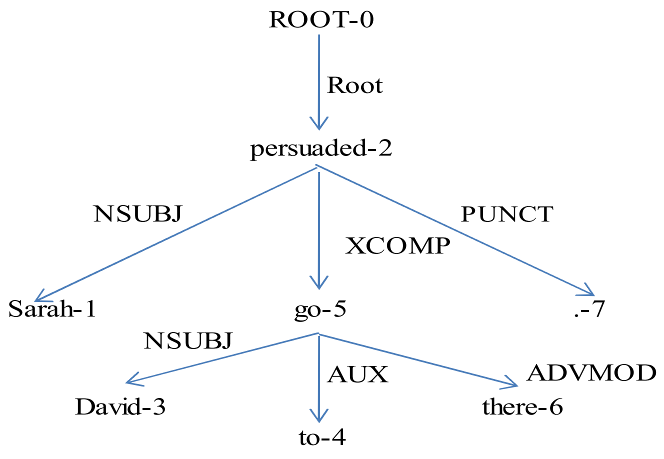


Figure 5.47. The typed-dependency tree for 'Sarah persuaded David to go there.' in Stanford Parser output

The equivalent functional structure for the tree in Figure 5.47 is shown below. Notice that there is no PRO, and that the lexical form of the verb 'persuaded' in this functional structure is not the same as the lexical form of the same verb in the functional structure in Figure 5.46, because this functional structure does not contain the direct object of the verb 'persuade.' For this functional structure to be complete, the verb 'persuade' should not require its direct object.

```
|      |PRED    'persuade<NSUBJ, XCOMP>'                              ||
|      |NSUBJ  |PRED     'Sarah'        |                            ||
|      |       |PERSON   3RD            |                            ||
|      |       |NUMBER   SINGULAR       |                            ||
|      |       |GENDER   FEMININE       |                            ||
|      |                                                             ||
|      |XCOMP  |PRED     'go<NSUBJ>'                       |         ||
|      |       |NSUBJ   |PRED     'David'      |           |         ||
|      |       |        |PERSON   3RD          |           |         ||
|      |       |        |NUMBER   SINGULAR     |           |         ||
|      |       |        |GENDER   MASCULINE    |           |         ||
|      |       |                                           |         ||
|      |       |ADVMOD  [PRED     'there'  ]               |         ||
|      |       |                                           |         ||
|      |       |AUX     |FORM     'to'     |               |         ||
|      |TENSE   PAST                                                 ||
|      |PUNCT  |FORM      '.'              |                         ||
|ROOT  |       |STMT-TYPE DECLARATIVE      |                         ||
```

Figure 5.48. The functional structure for ‘Sarah persuaded David to go there.’ which is equivalent to the typed-dependency tree in Figure 5.47

As for a transitive subject-control verb, the external subject of its open complement is the same as its subject. For example, the external subject of an external complement of the verb ‘promised’ is also the subject of ‘promised.’

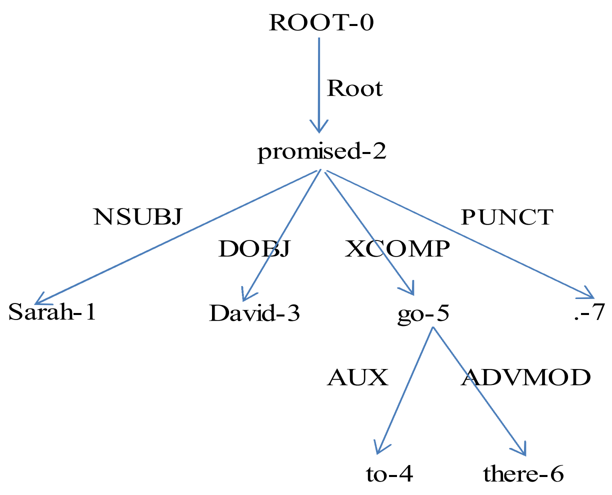


Figure 5.49. The typed-dependency tree for ‘Sarah promised David to go there.’

```
|      |PRED    'promise<NSUBJ, DOBJ, XCOMP>'          ||
|      |NSUBJ   |PRED     'Sarah_i'   |                ||
|      |        |PERSON   3RD         |                ||
|      |        |NUMBER   SINGULAR    |                ||
|      |        |GENDER   FEMININE    |                ||
|      |                                               ||
|      |DOBJ    |PRED     'David'     |                ||
|      |        |PERSON   3RD         |                ||
|      |        |NUMBER   SINGULAR    |                ||
|      |        |GENDER   MASCULINE   |                ||
|      |                                               ||
|      |XCOMP   |PRED     'go<NSUBJ>'     |            ||
|      |        |NSUBJ    |PRED   'PRO_i'||            ||
|      |        |         |TYPE   ZERO  ||            ||
|      |        |                         |            ||
|      |        |ADVMOD  {|PRED   'there'|}            ||
|      |        |                         |            ||
|      |        |AUX      |FORM   'to'  ||            ||
|      |TENSE   PAST                                   ||
|      |PUNCT   |FORM       '.'           |            ||
|ROOT  |        |STMT-TYPE  DECLARATIVE   |            ||
```

Figure 5.50. The functional structure for ‘Sarah promised David to go there.’

However, the Stanford Parser yields an incorrect parse for a transitive subject-control verb, because the subject of the complement is not open. In the example below, the subject of the verb ‘go’ is ‘David,’ which does not reflect the actual meaning of the sentence.

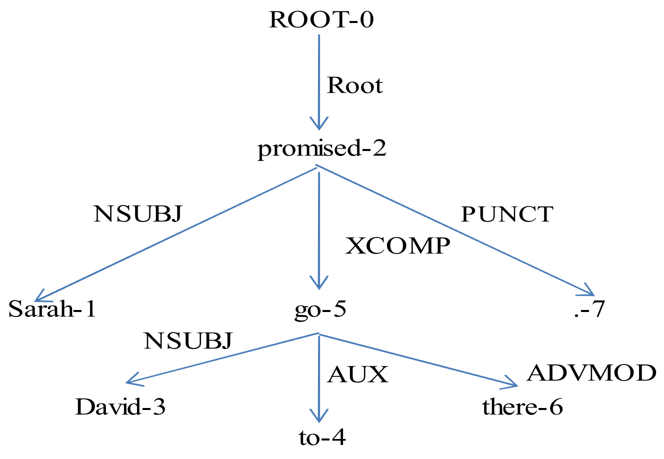


Figure 5.51. The typed-dependency tree for ‘Sarah promised David to go there’ in the Stanford Parser output

```
        PRED    'promise<NSUBJ, XCOMP>'
        NSUBJ   PRED     'Sarah'
                PERSON   3RD
                NUMBER   SINGULAR
                GENDER   FEMININE

        XCOMP   PRED     'go<NSUBJ>'
                NSUBJ    PRED      'David'
                         PERSON    3RD
                         NUMBER    SINGULAR
                         GENDER    MASCULINE

                ADVMOD   PRED      'there'

                AUX      FORM      'to'
        TENSE   PAST
        PUNCT   FORM      '.'
ROOT            STMT-TYPE DECLARATIVE
```

Figure 5.52. The functional structure for 'Sarah promised David to go there.' which is equivalent to the typed-dependency tree in Figure 5.51

These examples show that the Stanford Parser does not differentiate transitive subject-control verbs from transitive object-control verbs, and it yields incorrect output for transitive subject-control verbs. This analysis is a plausible one, because the majority of transitive control verbs are object-control, while there are only two transitive subject-control verbs: *promise* and *vow* (Davies & Dubinsky 2008, p.351).

As previously mentioned in the subsection for "aux", the Stanford Parser analyzes the auxiliaries that include to-infinitives (e.g., 'have to,' 'be to,' 'ought to') as the root of the sentence, and the 'to' to be an auxiliary depending on the verb after the auxiliary, which depends on the root with the type "xcomp." Figure 5.53 is the typed-dependency tree for 'Sarah has to read this book.,' and Figure 5.54 is the functional structure for the same sentence. Notice that the preposition 'to' is an auxiliary whose lexical information is its FORM attribute. Also notice that the MODALITY attribute has the value DEONTIC-NECESSITY.

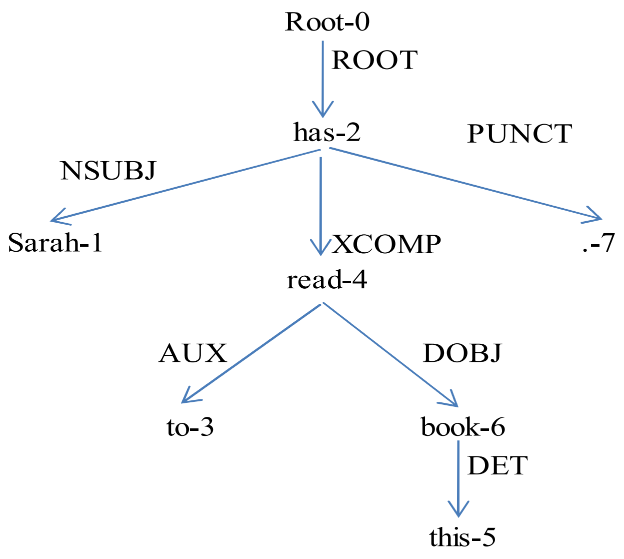


Figure 5.53. The typed-dependency tree for ‘Sarah has to read this book.’ in the Stanford Parser output.

```
|        |PRED      'have<NSUBJ,XCOMP>'                                          ||
|        |NSUBJ    |PRED 'Sarah_i'        |                                    ||
|        |         |PERSON    3RD         |                                    ||
|        |         |NUMBER    SINGULAR    |                                    ||
|        |         |GENDER    FEMININE    |                                    ||
|        |                                                                     ||
|        |XCOMP    |PRED 'read<NSUBJ,DOBJ>'                                |   ||
|        |         |NSUBJ   |PRED          'PRO_i'|                         |   ||
|        |         |        |FORM          ZERO   |                         |   ||
|        |         |                                                       |   ||
|        |         |DOBJ    |PRED          'book'                     |    |   ||
|        |         |        |PERSON        3RD                        |    |   ||
|        |         |        |NUMBER        SINGULAR                   |    |   ||
|        |         |        |GENDER        NEUTRAL                    |    |   ||
|        |         |        |DET          |FORM     'this'       |   |    |   ||
|        |         |        |             |TYPE     DEFINITE     |   |    |   ||
|        |         |                                                       |   ||
|        |         |AUX     |FORM          'to' |                          |   ||
|        |                                                                     ||
|        |TENSE     PRESENT                                                    ||
|        |MODALITY  DEONTIC-NECESSITY                                          ||
|        |                                                                     ||
|        |PUCNT    |FORM       '.'         |                                   ||
|   ROOT |         |STMT-TYPE  DECLARATIVE |                                   ||
```

Figure 5.54. The functional structure for ‘Sarah has to read this book.’

Figure 5.55 is the typed-dependency tree for ‘Sarah is to read this book.,’ and Figure 5.56 is the functional-structure representation for the same sentence. Notice that the TENSE attribute has the value FUTURE[30], and the MODALITY attribute has the value DEONTIC-NECESSITY; this information is carried by the chunk ‘is to *infinitive verb*.’

[30] In this dissertation, it is assumed that English has the future tense (see section 5.3.2).

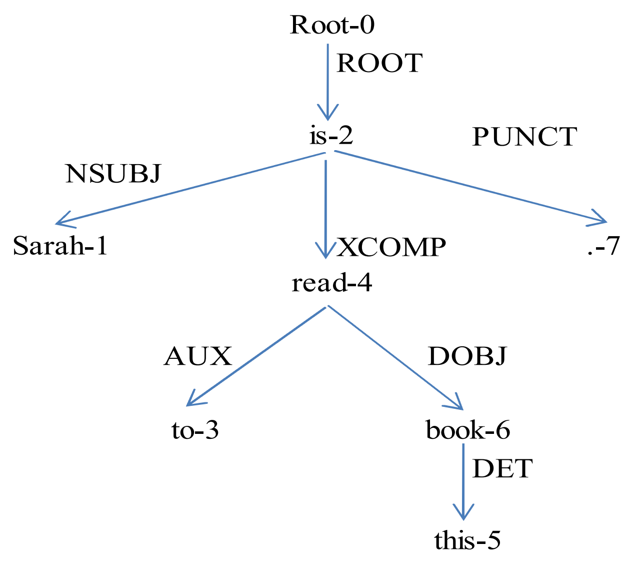


Figure 5.55. The typed-dependency tree for ‘Sarah is to read this book.’ in the Stanford Parser output.

```
     | PRED     'be<NSUBJ,XCOMP>'                                        ||
     | NSUBJ   |PRED 'Sarah_i'          |                                ||
     |         |PERSON  3RD             |                                ||
     |         |NUMBER SINGULAR         |                                ||
     |         |GENDER FEMININE         |                                ||
     |                                                                   ||
     | XCOMP   |PRED 'read<NSUBJ,DOBJ>'                          |       ||
     |         |NSUBJ   |PRED        'PRO_i'|                    |       ||
     |         |        |FORM        ZERO   |                    |       ||
     |         |                                                 |       ||
     |         |DOBJ    |PRED        'book'               |      |       ||
     |         |        |PERSON      3RD                  |      |       ||
     |         |        |NUMBER      SINGULAR             |      |       ||
     |         |        |GENDER      NEUTRAL              |      |       ||
     |         |        |DET        |FORM   'this'    |   |      |       ||
     |         |        |           |TYPE   DEFINITE  |   |      |       ||
     |         |                                                 |       ||
     |         |AUX     |FORM        'to' |                      |       ||
     |                                                                   ||
     | TENSE    FUTURE                                                   ||
     | MODALITY DEONTIC-NECESSITY                                        ||
     |                                                                   ||
     | PUCNT   |FORM     '.'            |                                ||
ROOT |         |STMT-TYPE DECLARATIVE   |                                ||
```

Figure 5.56. The functional structure for ‘Sarah is to read this book.’

Figure 5.57 is the typed-dependency tree for ‘Sarah ought to read this book.,’ and Figure 5.58 is the functional-structure representation for the same sentence.

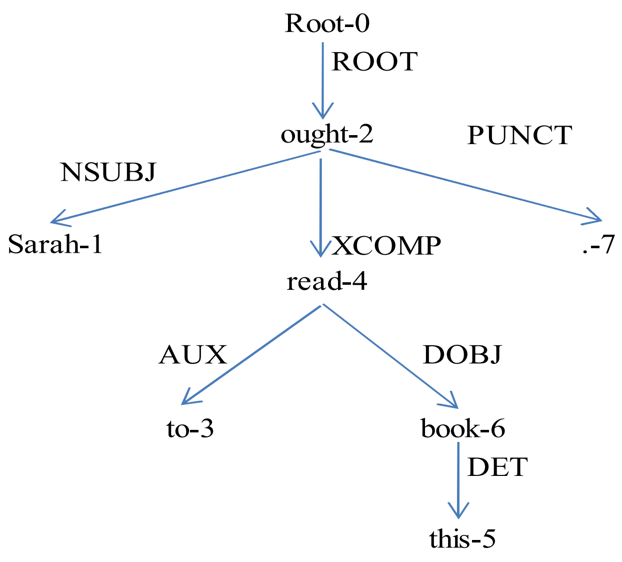


Figure 5.57. The typed-dependency tree for ‘Sarah ought to read this book.’

```
    |PRED     'ought<NSUBJ,XCOMP>'                                        ||
    |NSUBJ   |PRED 'Sarah_i'          |                                   ||
    |        |PERSON 3RD              |                                   ||
    |        |NUMBER SINGULAR         |                                   ||
    |        |GENDER FEMININE         |                                   ||
    |                                                                     ||
    |XCOMP   |PRED 'read<NSUBJ,DOBJ>'                              |      ||
    |        |NSUBJ  |PRED      'PRO_i'|                           |      ||
    |        |       |FORM      ZERO   |                           |      ||
    |        |                                                     |      ||
    |        |DOBJ   |PRED      'book'                      |      |      ||
    |        |       |PERSON    3RD                         |      |      ||
    |        |       |NUMBER    SINGULAR                    |      |      ||
    |        |       |GENDER    NEUTRAL                     |      |      ||
    |        |       |DET      |FORM  'this'       |        |      |      ||
    |        |       |         |TYPE  DEFINITE     |        |      |      ||
    |        |                                                     |      ||
    |        |AUX    |FORM      'to' |                             |      ||
    |                                                                     ||
    |TENSE    PRESENT                                                     ||
    |MODALITY DEONTIC-NECESSITY                                           ||
    |                                                                     ||
    |PUCNT   |FORM      '.'          |                                    ||
ROOT|        |STMT-TYPE DECLARATIVE  |                                    ||
```

Figure 5.58. The functional structure for ‘Sarah ought to read this book.’

The dependency type “xcomp” does not completely follow Mel’čuk’s original Criterion A for SSyntRel. The linear order of the words in this dependency type is such that the main verb precedes the verb in the subject-external clause; however, the verb in the main clause and the verb in the subject-external clause do not form a prosodic unit (e.g., ‘…persuaded… go’). On the other hand, the verb in the main clause and the verb in the subject-external clause do form a semantic unit. Therefore, this type follows the revised Criterion A discussed in Section 2.4.3.1.

This dependency type also follows Mel’čuk’s Criterion B. The main verb determines the passive valence of the phrase. For example, a verb that takes an external-subject clause can be dependent on another verb, such as ‘John believes Sarah persuaded David to go there.’ Therefore, both the main verb and the verb in the external-subject clause follow Mel’čuk’s Criterion B1.

In addition, this dependency type follows the revised Criterion C proposed in Section 2.4.3.3 because it implies a certain kind of semantic relationship between verbs, and this semantic relationship cannot be expressed by any other dependency type.

Pcomp – clausal complement of a preposition

The dependency type "pcomp" is a subtype of "comp" and represents cases where the head of the dependency is a preposition and the dependent is a clausal complement. Figures 5.59 through 5.62 present example sentences that contain the dependency type "pcomp" and their functional structures. In each of these examples, the prepositional object is a present participle of a verb whose subject is a zero pronoun.

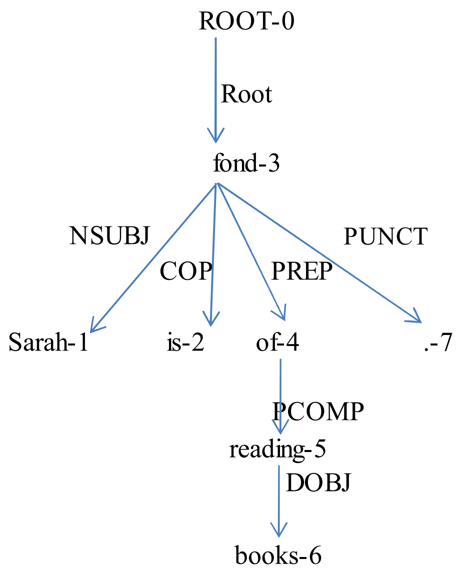


Figure 5.59. The typed-dependency tree for 'Sarah is fond of reading books.'

The typed-dependency tree above is equivalent to the functional structure below. Notice that the same index 'i' assigned to 'Sarah' and to 'PRO' indicates that this 'PRO' refers to 'Sarah.'

```
|         |PRED    'fond<NSUBJ, PREP>'                                                   ||
|         |NSUBJ  |PRED      'Sarah_i'  |                                                ||
|         |       |PERSON    3RD        |                                                ||
|         |       |NUMBER    SINGULAR   |                                                ||
|         |       |GENDER    FEMININE   |                                                ||
|         |                                                                              ||
|         |COP    |FORM      'is'       |                                                ||
|         |                                                                              ||
|         |PREP   |PRED      'of<PCOMP>'                                        |        ||
|         |       |PCOMP    |PRED        'read<SUBJ, OBJ>'               |      |        ||
|         |       |         |SUBJ       |PRED      'PRO_i'  |            |      |        ||
|         |       |         |           |TYPE      ZERO     |            |      |        ||
|         |       |         |                                            |      |        ||
|         |       |         |DOBJ       |PRED      'books'  |            |      |        ||
|         |       |         |           |PERSON    3RD      |            |      |        ||
|         |       |         |           |NUMBER    PLURAL   |            |      |        ||
|         |       |         |           |GENDER    NEUTRAL  |            |      |        ||
|         |TENSE   PRESENT                                                               ||
|         |PUNCT  |FORM      '.'         |                                               ||
|ROOT     |       |STMT-TYPE DECLARATIVE |                                               ||
```

Figure 5.60. The functional structure for 'Sarah is fond of reading books.'

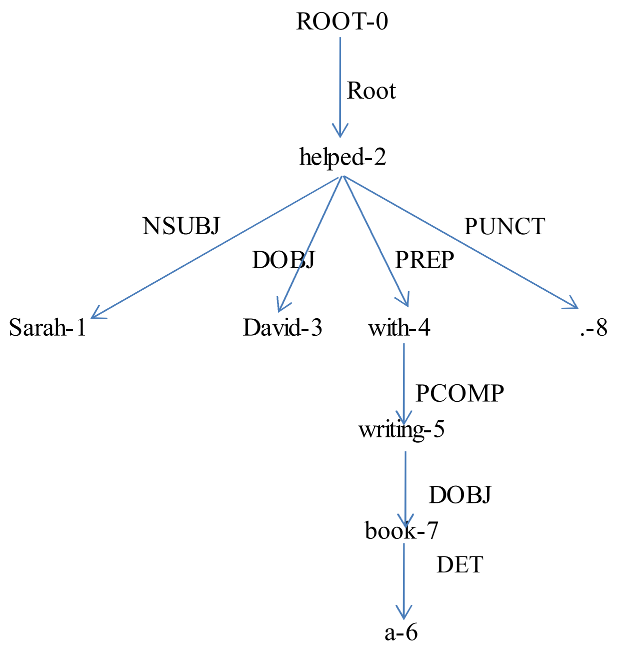


Figure 5.61. The typed-dependency tree for 'Sarah helped David with writing a book.'

The typed-dependency tree above is equivalent to the functional structure below.

```
       PRED    'help<NSUBJ, DOBJ>'
       NSUBJ   PRED      'Sarah'
               PERSON    3RD
               NUMBER    SINGULAR
               GENDER    FEMININE

       DOBJ    PRED      'David'
               PERSON    3RD
               NUMBER    SINGULAR
               GENDER    MASCULINE

       PREP    PRED      'with<PCOMP>'
               PCOMP     PRED       'write<NSUBJ,DOBJ>'
                         NSUBJ      PRED    'PRO'
                                    TYPE    ZERO

                         DOBJ       PRED    'book'
                                    PERSON  3RD
                                    GENDER  NEUTRAL
                                    DET     FORM    'a'
                                            TYPE    INDEFINITE
                                            NUMBER  SINGULAR
       TENSE   PAST
       PUNCT   FORM      '.'
ROOT           STMT-TYPE DECLARATIVE
```

Figure 5.62. The functional structure for ‘Sarah helped David with writing a book.’

When the main predicate of a prepositional complement is an adjective, this adjective depends on the preposition, as shown below.

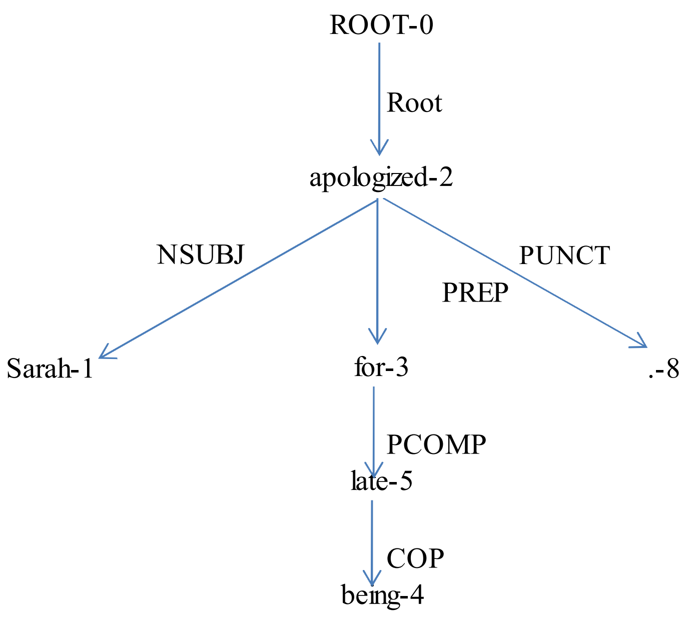


Figure 5.63. The typed-dependency tree for 'Sarah apologized for being late.'

```
|         |PRED     'apologize <NSUBJ, PREP>'                                   ||
|         |NSUBJ   |PRED       'Sarah'       |                                  ||
|         |        |PERSON     3RD           |                                  ||
|         |        |NUMBER     SINGULAR      |                                  ||
|         |        |GENDER     FEMININE      |                                  ||
|         |                                                                     ||
|         |PREP    |PRED       'for<PCOMP>'                             |       ||
|         |        |PCOMP     |PRED         'late<NSUBJ>'         |     |       ||
|         |        |          |NSUBJ       |PRED   'PRO'   |      |     |       ||
|         |        |          |            |TYPE   ZERO    |      |     |       ||
|         |        |          |                                   |     |       ||
|         |        |          |COP         |FORM   'being' |      |     |       ||
|         |TENSE    PAST                                                        ||
|         |PUNCT   |FORM       '.'           |                                  ||
|ROOT     |        |STMT-TYPE  DECLARATIVE   |                                  ||
```

Figure 5.64. The functional structure for 'Sarah apologized for being late.'

The dependency type "pcomp" partly follows Mel'čuk's Criteria for SSyntRel. The preposition and the present-participial verb form a prosodic unit (e.g., 'of reading'), and the linear order of the words in this dependency type is such that the preposition precedes the present-participle verb. However, when the main predicate is an adjective and the copula is in the present participle form, as in ‘Sarah apologized for being late’ in the example above, the preposition and the adjective do not form a prosodic unit. However, this type follows the revised Criterion A discussed in Section 2.4.3.1, because the preposition and the adjective in this dependency type represent a semantic unit. In other words, they constitute a fragment functional structure (see Section 3.3).

The passive valence of the phrase ‘of reading’ in the example above is determined by the preposition ‘of,’ and according to Mel’čuk’s Criterion B1, the preposition is the head and the present-participial verb is the dependent.

This dependency type also implies a unique semantic relationship between the governor and its dependent, and the prototypical dependent of this type is a present-participle verb, or an adjective accompanied by a copula. Therefore, this dependency type follows the revised Criteria C1 and C2 proposed in Section 2.4.3.3.

Obj – object

This dependency type is a subtype of “comp” and describes cases where the dependency head is an object-taking element and the dependent is the object of the head. This dependency type is further subcategorized into three subtypes: “dobj,” “iobj,” and “pobj.”

These dependency types follow Mel’čuk’s criteria for SSyntRel. In all cases, the object-taking element and its dependent form a prosodic unit (e.g., ‘read books,’ ‘give him,’ and ‘about books’), and the linear order of the words in this dependency type is such that the verb precedes the noun. These types also follow the revised Criterion A (see Section 2.4.3.1) because they represent semantic units, even though they do not form prosodic units (e.g., ‘read’ and ‘books’ in ‘Sarah has read these old, expensive books’).

In addition, the object-taking element, not its dependent, determines the passive valence of the phrase. Therefore, the object-taking element and its dependent follow Mel’čuk’s Criterion B1. For example, for the dependency relation between ‘read’ and ‘book’ in the sentence ‘Sarah has read this book’ shows that ‘read’ depends on another element (“root”).

This dependency type also follows the revised Criterion C proposed in Section 2.4.3.3, because it implies a certain kind of semantic relationship between a noun and a verb, and this semantic relationship cannot be expressed by any other dependency type.

Dobj - direct object

This dependency type is a subtype of “obj” used for cases where the dependency head is an object-taking element and the dependent is the direct object of the head. Several cases of “dobj” have been shown in the example sentences so far. In the figure below, the noun ‘book’ is the direct object of the verb ‘written.’ In other words, the noun ‘book’ depends on the verb ‘written’ with the dependency type “dobj.”

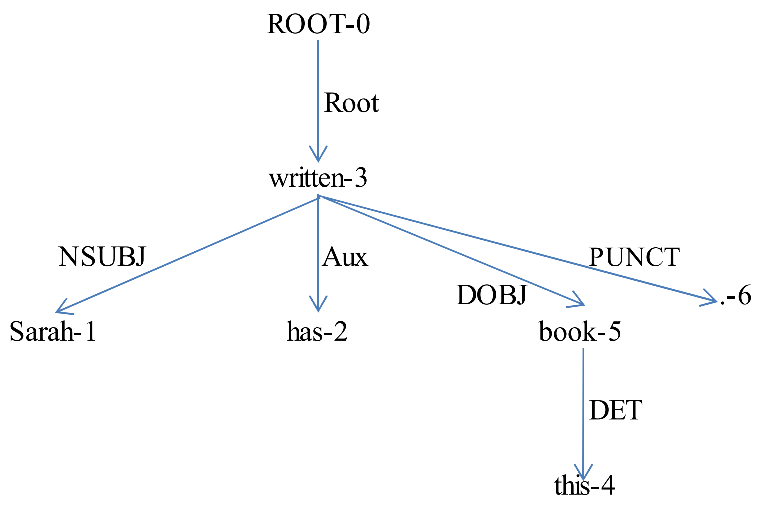


Figure 5.65. The typed-dependency tree for ‘Sarah has written this book.’

The typed-dependency tree above is equivalent to the functional structure below.

$$\left|\begin{array}{ll} \text{ROOT} & \left|\begin{array}{ll} \text{PRED} & \text{'write<NSUBJ,DOBJ>'} \\ \text{NSUBJ} & \left|\begin{array}{ll} \text{PRED} & \text{'Sarah'} \\ \text{PERSON} & \text{3RD} \\ \text{NUMBER} & \text{SINGULAR} \\ \text{GENDER} & \text{FEMININE} \end{array}\right| \\ \text{DOBJ} & \left|\begin{array}{ll} \text{PRED} & \text{'book'} \\ \text{PERSON} & \text{3RD} \\ \text{NUMBER} & \text{SINGULAR} \\ \text{GENDER} & \text{NEUTRAL} \\ \text{DET} & \left|\begin{array}{ll} \text{FORM} & \text{'this'} \\ \text{TYPE} & \text{DEFINITE} \end{array}\right| \end{array}\right| \\ \text{AUX} & \left\{ \left|\begin{array}{ll} \text{FORM} & \text{'has'} \\ \text{TENSE} & \text{PRESENT} \\ \text{ASPECT} & \text{PERFECT} \end{array}\right| \right\} \\ \text{PUCNT} & \left|\begin{array}{ll} \text{FORM} & \text{'.'} \\ \text{STMT-TYPE} & \text{DECLARATIVE} \end{array}\right| \end{array}\right| \end{array}\right\|$$

Figure 5.66. The functional structure for ‘Sarah has written this book.’

Iobj - indirect object

This dependency type is a subtype of “obj” and describes cases where the dependency head is an object-taking element and the dependent is the indirect object of the head. Figures 5.67 through 5.70 present the typed-dependency trees for example sentences that contain the dependency type “iobj” along with their functional structures.

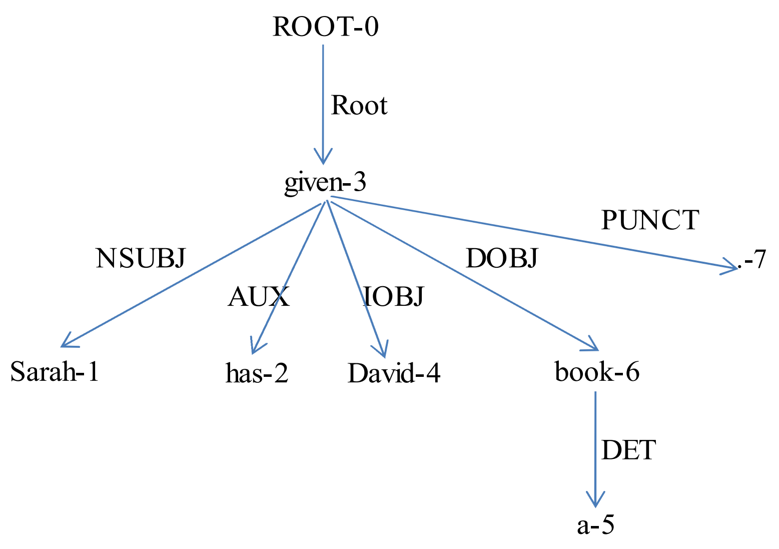


Figure 5.67. The typed-dependency tree for ‘Sarah has given David a book.’

The typed-dependency tree above is equivalent to the functional structure below.

```
|          |PRED     'give<NSUBJ,IOBJ,DOBJ>'                  ||
|          |NSUBJ    |PRED 'Sarah'                   |        ||
|          |         |PERSON  3RD                    |        ||
|          |         |NUMBER SINGULAR                |        ||
|          |         |GENDER  FEMININE               |        ||
|          |                                                  ||
|          |IOBJ     |PRED       'David'             |        ||
|          |         |PERSON  3RD                    |        ||
|          |         |NUMBER SINGULAR                |        ||
|          |         |GENDER  MASCULINE              |        ||
|          |                                                  ||
|          |DOBJ     |PRED 'book'                     |       ||
|          |         |PERSON  3RD                     |       ||
|          |         |NUMBER SINGULAR                 |       ||
|          |         |GENDER  NEUTRAL                 |       ||
|          |         |DET        |FORM 'a'           ||       ||
|          |         |           |TYPE INDEFINITE    ||       ||
|          |                                                  ||
|          |AUX     {|FORM      'has'                |}       ||
|          |         |TENSE     PRESENT              |        ||
|          |         |ASPECT  PERFECT                |        ||
|          |                                                  ||
|          |PUCNT    |FORM      '.'                  |        ||
|ROOT      |         |STMT-TYPE     DECLARATIVE      |        ||
```

Figure 5.68. The functional structure for ‘Sarah has given David a book.’

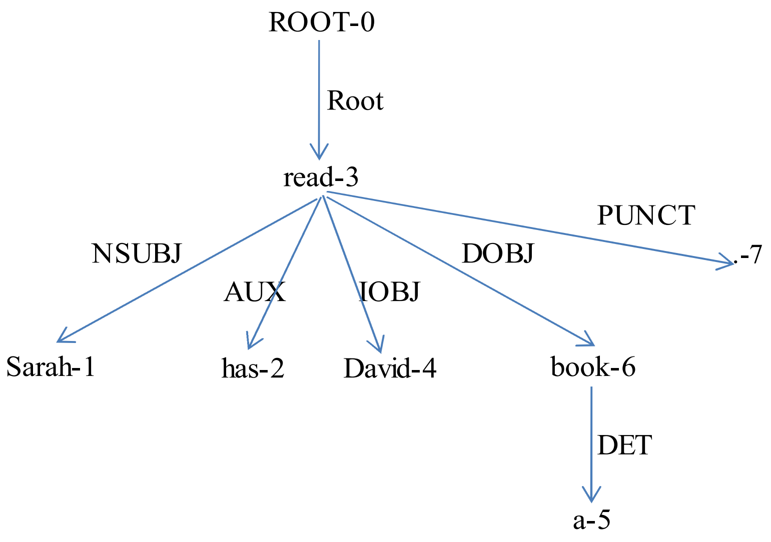


Figure 5.69. The typed-dependency tree for ‘Sarah has read David a book.’

The typed-dependency tree above is equivalent to the functional structure below.

```
     PRED    'read<NSUBJ,IOBJ,DOBJ>'
     NSUBJ   PRED 'Sarah'
             PERSON  3RD
             NUMBER  SINGULAR
             GENDER  FEMININE

     IOBJ    PRED    'David'
             PERSON  3RD
             NUMBER  SINGULAR
             GENDER  MASCULINE

     DOBJ    PRED 'book'
             PERSON  3RD
             NUMBER  SINGULAR
             GENDER  NEUTRAL
             DET     FORM 'a'
                     TYPE INDEFINITE

     AUX     FORM    'has'
             TENSE   PRESENT
             ASPECT  PERFECT

     PUCNT   FORM    '.'
ROOT         STMT-TYPE DECLARATIVE
```

Figure 5.70. The functional structure for ‘Sarah has read David a book.’

Pobj - object of preposition

This dependency type is a subtype of “obj” and describes cases where the dependency head is a preposition and the dependent is the object of the preposition. Figure 5.71 presents an example sentence with this dependency type and Figure 5.72 presents its equivalent functional structure.

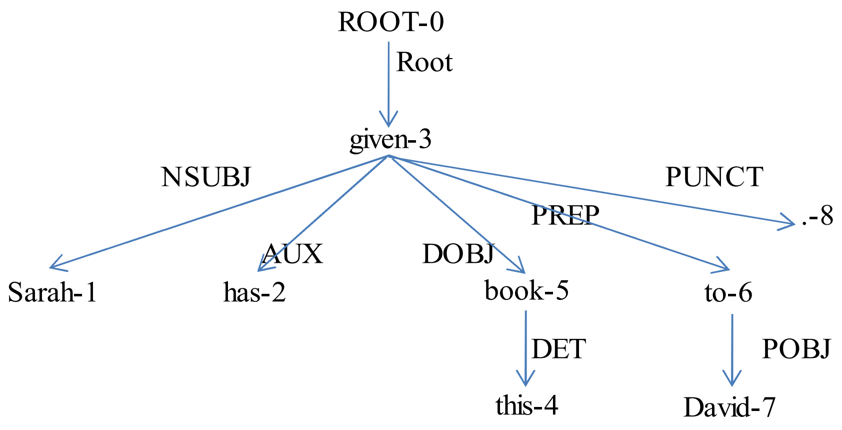


Figure 5.71. The typed-dependency tree for ‘Sarah has given this book to David.’

```
      | PRED     'give<NSUBJ,DOBJ,PREP>'                                  ||
      | NSUBJ    |PRED 'Sarah'          |                                 ||
      |          |PERSON  3RD           |                                 ||
      |          |NUMBER SINGULAR       |                                 ||
      |          |GENDER  FEMININE      |                                 ||
      |                                                                   ||
      | DOBJ     |PRED 'book'                          |                  ||
      |          |PERSON  3RD                          |                  ||
      |          |NUMBER SINGULAR                      |                  ||
      |          |GENDER  NEUTRAL                      |                  ||
      |          |DET       |FORM 'this'            | |                  ||
      |          |          |TYPE DEMONSTRATIVE     | |                  ||
      |                                                                   ||
      | PREP     |PRED      'to<POBJ>'                          |         ||
      |          |POBJ      |PRED         'David'      |        |         ||
      |          |          |PERSON       3RD          |        |         ||
      |          |          |NUMBER       SINGULAR     |        |         ||
      |          |          |GENDER       MASCULINE    |        |         ||
      |                                                                   ||
      | AUX     {|FORM      'has'        |}                               ||
      |         {|ASPECT    PERFECT      |}                               ||
      |         {|TENSE     PRESENT      |}                               ||
      |                                                                   ||
      | PUCNT    |FORM      '.'                     |                     ||
ROOT  |          |STMT-TYPE DECLARATIVE             |                     ||
```

Figure 5.72. The functional structure for ‘Sarah has given this book to David.’

Objects in topicalized sentences and interrogatives

There are some instances of marked syntactic structures whereby a phrase that is not usually placed at the beginning of a sentence occurs in that position, such as interrogatives or topicalization (Prince 1981, 1998). These cases carry a particular discourse function, such as "topic."

Topicalization

As for topicalized direct objects, one possible analysis is that the topicalized phrase depends on the predicate as a direct object, and thus the dependency would not be labeled as "topic," as shown in the following typed-dependency tree. The functional structure for this sentence is identical with that for 'Sarah has written this book.' except for the presence of "PUNCT."

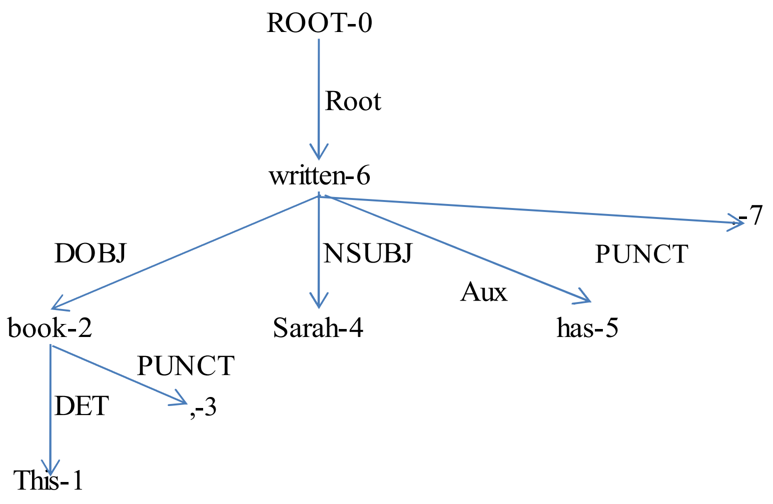


Figure 5.73. The typed-dependency tree for 'This book, Sarah has written.'

```
     | PRED    'write<NSUBJ,DOBJ>'
     | NSUBJ   | PRED 'Sarah'
     |         | PERSON  3RD
     |         | NUMBER SINGULAR
     |         | GENDER  FEMININE
     |
     | DOBJ    | PRED 'book'
     |         | PERSON  3RD
     |         | NUMBER SINGULAR
     |         | GENDER  NEUTRAL
     |         | DET       | FORM 'this'
     |         |           | TYPE DEFINITE
     |         |
     |         | PUNCT     | FORM ','
     |
     | AUX     | FORM     'has'
     |         | TENSE    PRESENT
     |         | ASPECT   PERFECT
     |
     | PUCNT   | FORM     '.'
ROOT |         | STMT-TYPE DECLARATIVE
```

Figure 5.74. The functional structure for ‘This book, Sarah has written.’

Another possible analysis is that the topicalized phrase depends on the predicate as a topic, and thus the dependency would be labeled as “topic,” as shown in the following typed-dependency tree.

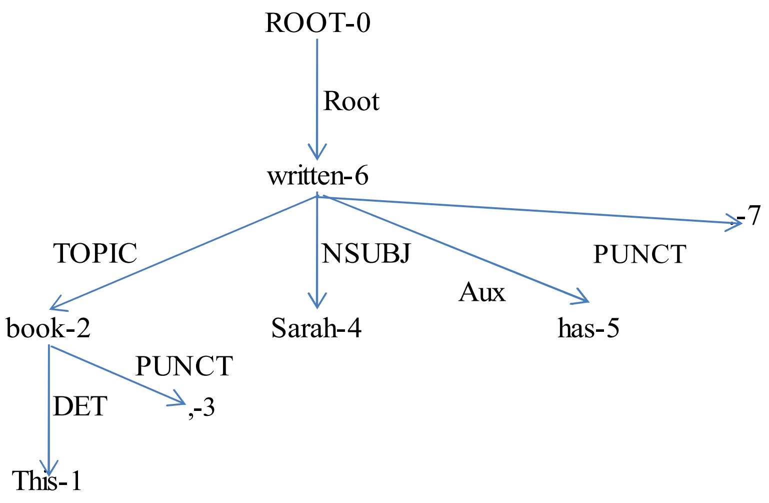


Figure 5.75. The typed-dependency tree for ‘This book, Sarah has written.’ with the dependency type “topic.”

In the functional structure for this typed-dependency tree, the DOBJ attribute has ‘PRO’ as its value, and this ‘PRO’ refers to ‘book,’ which is the PRED value of the TOPIC.

```
     | PRED    'write<NSUBJ,DOBJ>'                          | |
     | TOPIC  | PRED 'book_i'                          |    | |
     |        | PERSON  3RD                            |    | |
     |        | NUMBER SINGULAR                        |    | |
     |        | GENDER NEUTRAL                         |    | |
     |        | DET     |FORM 'this'     |             |    | |
     |        |         |TYPE DEFINITE   |             |    | |
     |        |                                        |    | |
     |        | PUNCT   |FORM ','   |                  |    | |
     |                                                      | |
     | NSUBJ  | PRED 'Sarah'                |               | |
     |        | PERSON  3RD                 |               | |
     |        | NUMBER SINGULAR             |               | |
     |        | GENDER FEMININE             |               | |
     |                                                      | |
     | DOBJ   | PRED     'PRO_i'            |               | |
     |        | TYPE     ZERO               |               | |
     |                                                      | |
     | AUX   {| FORM     'has'                  |}          | |
     |       {| TENSE    PRESENT                |}          | |
     |       {| ASPECT   PERFECT                |}          | |
     |                                                      | |
     | PUCNT  | FORM     '.'                    |           | |
ROOT |        | STMT-TYPE DECLARATIVE           |           | |
```

Figure 5.76. The functional structure for ‘This book, Sarah has written.’ with the dependency type “topic.”

The latter analysis is better than the former one, because it includes the discourse function “topic” in the typed-dependency tree and its functional structure. Unfortunately, Stanford Dependencies do not include the dependency type “topic;” therefore, we cannot obtain correct parses for sentences with a topicalized object. In addition, the Stanford Parser does not parse sentences correctly with a topicalized object. In the output typed-dependency tree, the word ‘book’ is correctly parsed to depend on ‘written,’ but with an incorrect type “nsubj”, and the word ‘Sarah’ is incorrectly parsed to depend on ‘book’ with an incorrect type “appos.”

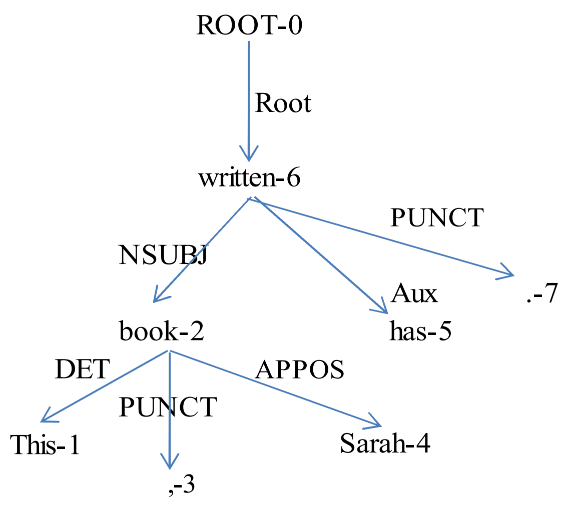


Figure 5.77. The incorrect typed-dependency tree for ‘This book, Sarah has written’ in the Stanford Parser output

The Stanford Parser has to be tuned up in order to yield a correct parse for sentences that contain topicalized phrases. We cannot deal with this problem at present, so we leave this topic for further research.

Interrogatives

Interrogative direct objects, which are also placed at the beginning of a sentence, are correctly parsed by the Stanford Parser.

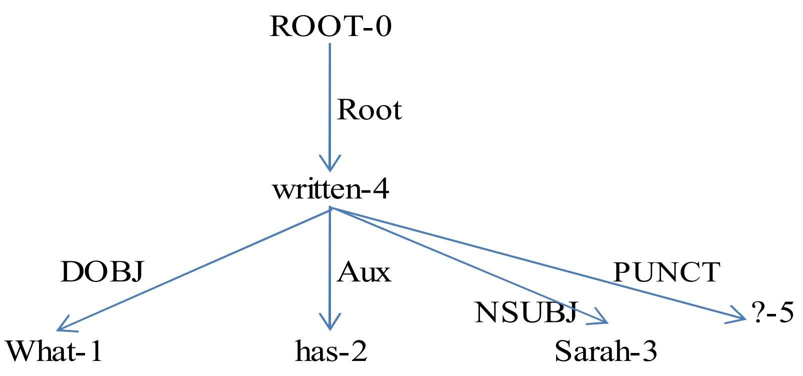


Figure 5.78. The typed-dependency tree for ‘What has Sarah written?’ in the Stanford Parser output

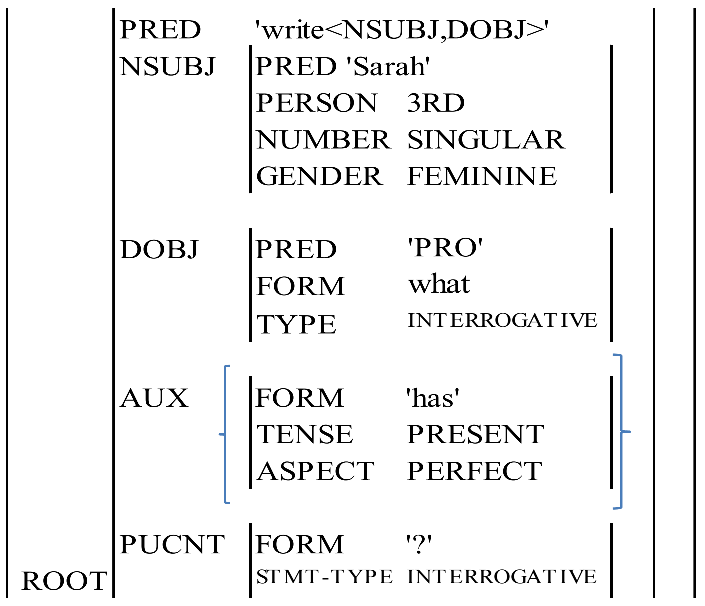


Figure 5.79. The functional structure for ‘What has Sarah written?’[31]

[31] This study does not distinguish the functional structures for interrogative sentences from those for echo questions such as ‘Sarah has written what?’ In both cases, the interrogative pronoun has one grammatical function (e.g., OBJ in Figure 5.79), and the difference in word order results in the difference in the tree, but not in functional structure. This is due to the insight that one functional structure can correspond to more than one constituent structure (or typed-dependency tree).

Interrogative indirect objects, on the other hand, are not correctly parsed by the Stanford Parser. In the example below, the word 'Who' is correctly parsed to depend on 'given,' but the dependency type is "dep", which means that the parser cannot give a correct type for the dependency. The correct dependency type is "iobj."

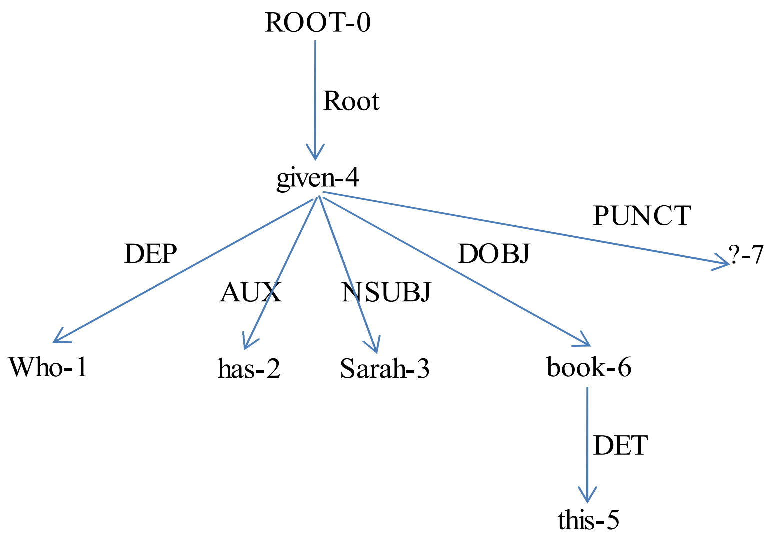


Figure 5.80. The incorrect typed-dependency tree for 'Who has Sarah given this book?' in the Stanford Parser output.

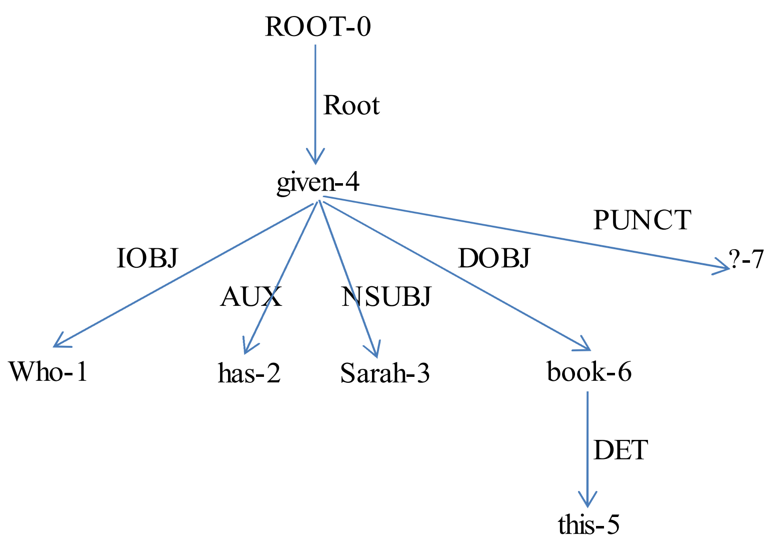


Figure 5.81. The correct typed-dependency tree for ‘Who has Sarah given this book?’

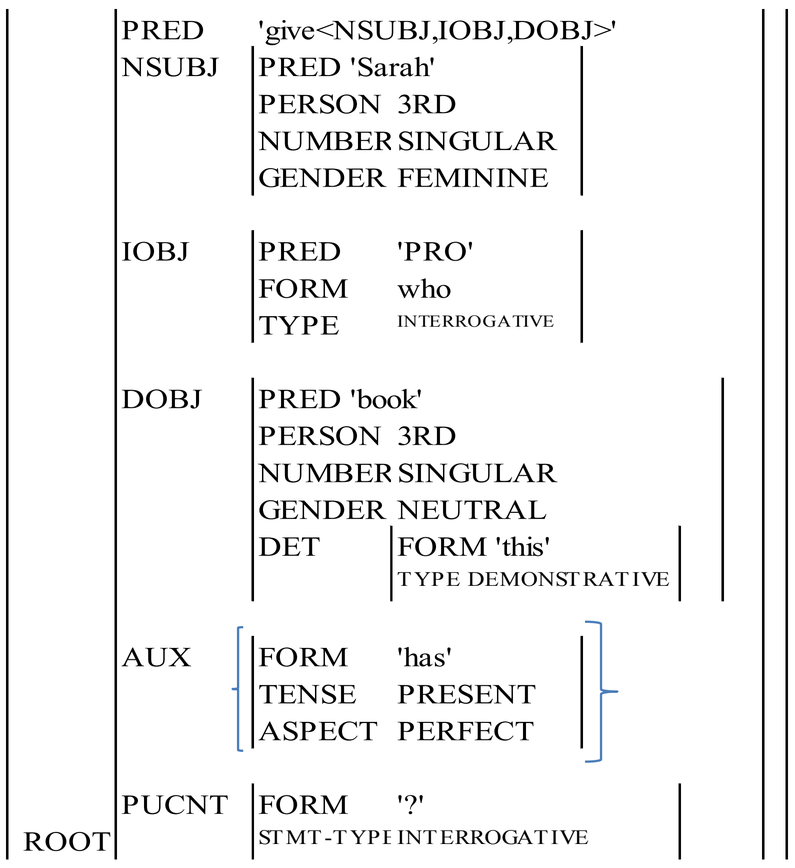


Figure 5.82. The functional structure for ‘Who has Sarah given this book?’

Interrogative prepositional objects are also not correctly parsed by the Stanford Parser. In the example below, the word 'What' is incorrectly parsed to depend on 'given,' and the dependency type is "dep." The correct parse would be that 'What' depend on 'on' with the type "pobj."

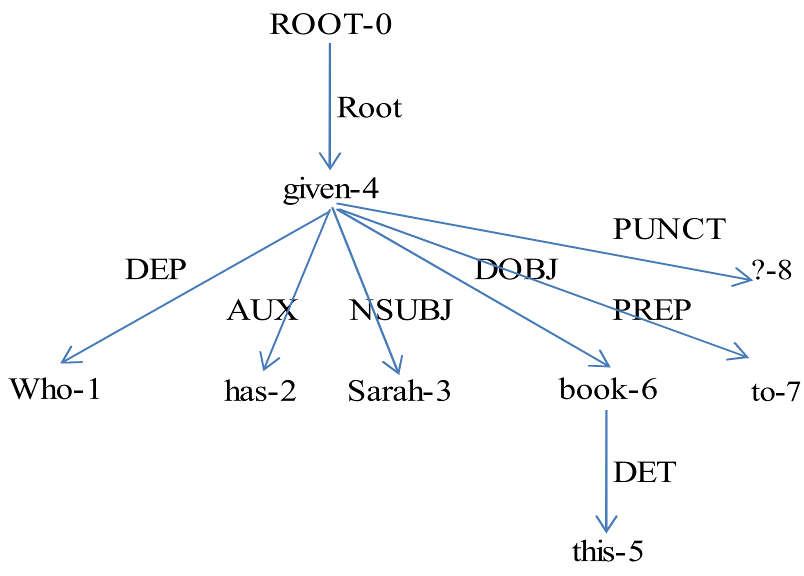


Figure 5.83. The typed-dependency tree for 'Who has Sarah given this book to?' in the Stanford Parser output.

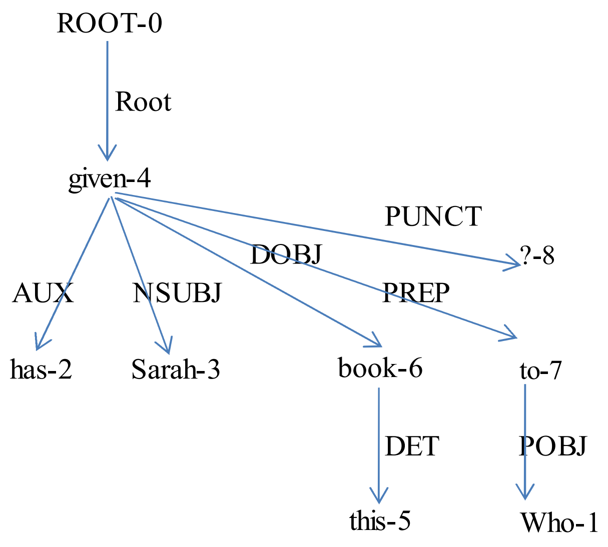


Figure 5.84. The correct typed-dependency tree for ‘Who has Sarah given this book to?’

```
|     |PRED    'give<NSUBJ,DOBJ,PREP>'                         ||
|     |NSUBJ  |PRED 'Sarah'           |                        ||
|     |       |PERSON    3RD          |                        ||
|     |       |NUMBER    SINGULAR     |                        ||
|     |       |GENDER    FEMININE     |                        ||
|     |                                                        ||
|     |DOBJ   |PRED 'book'                      |              ||
|     |       |PERSON    3RD                   |              ||
|     |       |NUMBER    SINGULAR              |              ||
|     |       |GENDER    NEUTRAL               |              ||
|     |       |DET      |FORM 'this'          ||              ||
|     |       |         |TYPE DEMONSTRATIVE   ||              ||
|     |                                                        ||
|     |PREP   |PRED      'to<POBJ>'                   |       ||
|     |       |POBJ     |PRED       'PRO'          ||       ||
|     |       |         |FORM       who            ||       ||
|     |       |         |TYPE       INTERROGATIVE  ||       ||
|     |                                                        ||
|     |AUX   [|FORM      'has'        |]                      ||
|     |       |ASPECT    PERFECT      |                       ||
|     |       |TENSE     PRESENT      |]                      ||
|     |                                                        ||
|     |PUCNT  |FORM      '?'                 |                 ||
| ROOT|       |STMT-TYPE INTERROGATIVE       |                 ||
```

Figure 5.85. The functional structure for ‘Who has Sarah given this book to?’

Subj – subject

This dependency type is a subtype of “arg” and describes cases where the dependency head is a subject-taking element and the dependent is the subject of the head. This dependency type is further subcategorized into four subtypes: “nsubj,” “nsubjpass,” “csubj,” and “csubjpass.”

There are some instances in which these dependency types do not follow Mel’čuk’s Criteria A for SSyntRel (see Section 2.4.3.1). The linear order of the words in this dependency type is such that the dependent precedes the head.

The subject-taking element and its dependent follow Mel'čuk's Criterion B1 because the subject-taking element, not its dependent, determines the passive valence of the phrase. For example, the dependency relation between 'read' and 'Sarah' in the sentence 'Sarah has read this book' shows that 'read' depends on another element (i.e., the "root").

This dependency type also follows the revised Criterion C proposed in Section 2.4.3.3, because it implies a certain kind of semantic relationship between a noun and a verb, and this semantic relationship cannot be expressed by any other dependency type.

Nsubj - nominal subject

This dependency type is a subtype of "subj" and describes cases where the head is a subject-taking element and the dependent is the nominal subject of the head. Cases of "nsubj" have been shown in the example sentences so far. In the figure below, the noun 'Sarah' is the nominal subject of the verb 'write.' In other words, the noun 'Sarah' depends on the verb 'write' with the dependency type "nsubj."

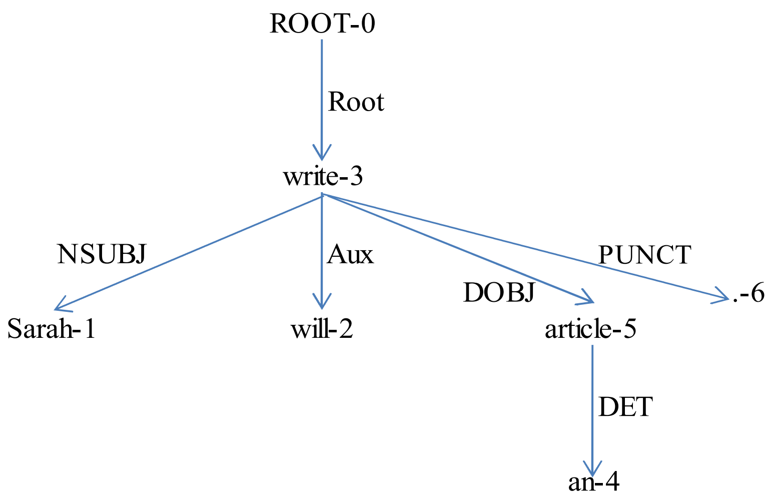


Figure 5.86. The typed-dependency tree for 'Sarah will write an article.'

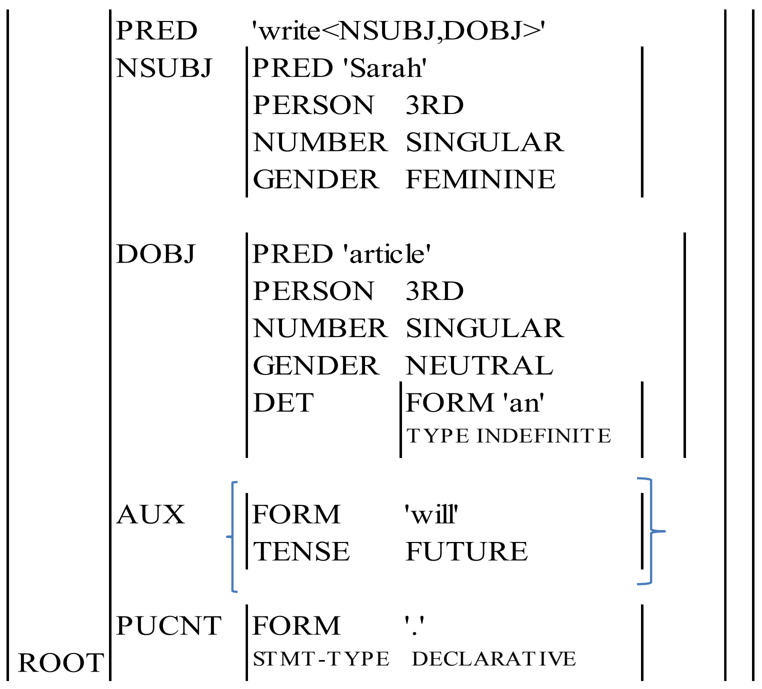


Figure 5.87. The functional structure for ‘Sarah will write an article.’

In addition to this, the subject of an external open complement of a transitive control verb is also typed as “nsubj,” as shown in the section on “xcomp” (the typed-dependency trees and functional structures for sentences that contain “xcomp” have already been shown in the section on “xcomp” above).

Csubj - clausal subject

This dependency type is a subtype of “subj” and represents cases where the dependency head is a subject-taking element and the dependent is the head of a clausal subject. The figure below presents the typed-dependency tree for an example sentence. In the figure below, the head of the clausal subject ‘said’ depends on the verb ‘makes’ with the dependency type “csubj.”

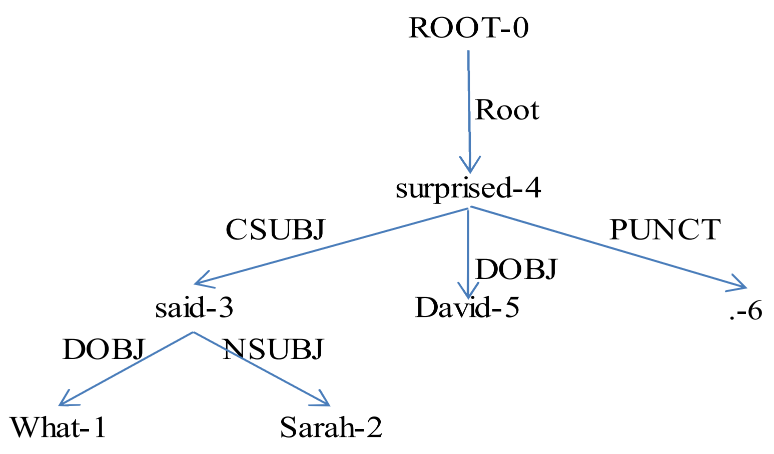


Figure 5.88. The typed-dependency tree for ‘What Sarah said surprised David.’

The typed-dependency tree above is equivalent to the functional structure below. Notice that the zero pronoun, which is the DOBJ of ‘said,’ refers to nothing in this structure. This zero pronoun refers to something *inter-clausally*, i.e., it refers to something in the context.

```
     | PRED    'surprise<CSUBJ, DOBJ>'
     | CSUBJ  | PRED      'say<NSUBJ,DOBJ>'
     |        | NSUBJ    | PRED        Sarah     |
     |        |          | PERSON      3RD       |
     |        |          | NUMBER      SINGULAR  |
     |        |          | GENDER      FEMININE  |
     |        |
     |        | DOBJ     | PRED        'PRO'     |
     |        |          | FORM        what      |
     |        |          | TYPE        RELATIVE  |
     |        | TENSE     PAST
     |
     | DOBJ   | PRED      David      |
     |        | PERSON    3RD        |
     |        | NUMBER    SINGULAR   |
     |        | GENDER    MASCULINE  |
     | TENSE   PAST
     | PUNCT  | FORM      '.'        |
ROOT |        | STMT-TYPE DECLARATIVE|
```

Figure 5.89. The functional structure for ‘What Sarah said surprised David.’

This dependency type is used for to-infinitive clauses or present-participle clauses.

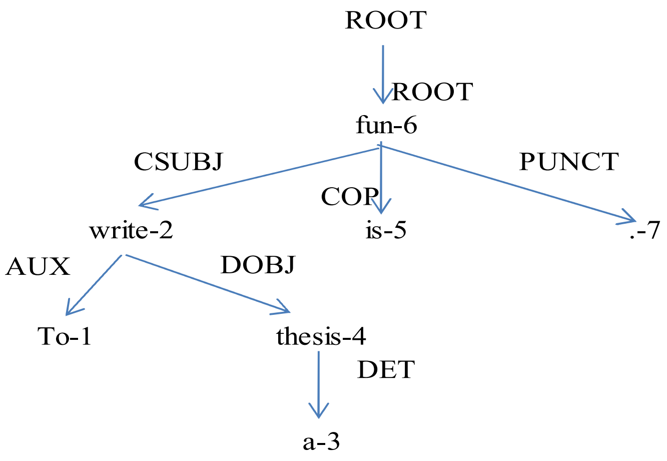


Figure 5.90. The typed-dependency tree for ‘To write a thesis is fun.’

The typed-dependency tree above is equivalent to the functional structure below. The zero pronoun, which is the NSUBJ of ‘write,’ refers to nothing within this structure; rather, it refers to some people who can be identified according to the context in which this sentence is used.

```
|         |PRED     'fun<CSUBJ>'                                        ||
|         |CSUBJ   |PRED      'write<NSUBJ,DOBJ>'                |      ||
|         |        |NSUBJ    |PRED       'PRO'          |        |      ||
|         |        |         |TYPE       ZERO           |        |      ||
|         |        |                                             |      ||
|         |        |DOBJ     |PRED       'thesis'          |     |      ||
|         |        |         |PERSON     3RD               |     |      ||
|         |        |         |NUMBER     SINGULAR          |     |      ||
|         |        |         |GENDER     NEUTRAL           |     |      ||
|         |        |         |DET       |FORM 'a'        ||     |      ||
|         |        |         |          |TYPE INDEFINITE ||     |      ||
|         |        |                                             |      ||
|         |        |AUX      |FORM   to |                        |      ||
|         |                                                             ||
|         |COP     |FORM      'is'      |                               ||
|         |        |TENSE     PRESENT   |                               ||
|         |                                                             ||
|         |PUNCT   |FORM      '.'       |                               ||
|ROOT     |        |STMT-TYPE DECLARATIVE|                              ||
```

Figure 5.91. The functional structure for ‘To write a thesis is fun.’

The dependency type “csubj” is also used for a participle functioning as an argument of a verb, as shown in the typed-dependency tree blow.

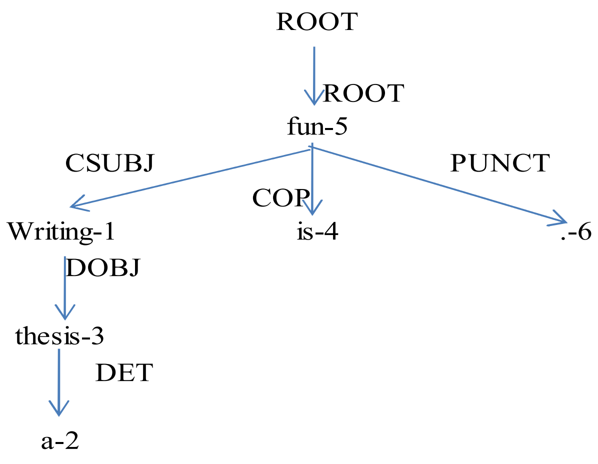


Figure 5.92. The typed-dependency tree for ‘Writing a thesis is fun.’

The typed-dependency tree above is equivalent to the functional structure below.   The zero pronoun, which is the NSUBJ of ‘writing,’ refers to people in general.

```
      |PRED    'fun<CSUBJ>'
      |CSUBJ  |PRED     'write<NSUBJ,DOBJ>'
      |       |NSUBJ   |PRED      'PRO'          |
      |       |        |TYPE      ZERO           |
      |       |
      |       |DOBJ    |PRED      'thesis'         |
      |       |        |PERSON    3RD              |
      |       |        |NUMBER    SINGULAR         |
      |       |        |GENDER    NEUTRAL          |
      |       |        |DET      |FORM 'a'        ||
      |       |        |         |TYPE INDEFINITE ||
      |       |
      |
      |COP    |FORM     'is'      |
      |       |TENSE    PRESENT   |
      |
      |PUNCT  |FORM     '.'          |
ROOT  |       |STMT-TYPE DECLARATIVE |
```

Figure 5.93. The functional structure for ‘Writing a thesis is fun.’

When the head of a present-participle clause is an intransitive verb without adverbials, or a transitive verb without its object, the Stanford Parser parses the head as a noun and the dependency type “csubj” is not used. For example, in the figure below, the word ‘writing’ has no direct object; hence it is analyzed as a noun that depends on the word ‘fun’ with the dependency type “nsubj.”

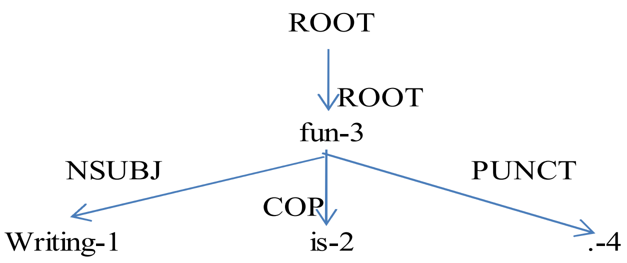


Figure 5.94. The typed-dependency tree for ‘Writing is fun.’ in the Stanford Parser output

$$\left[\text{ROOT}\ \left[\begin{array}{ll}
\text{PRED} & \text{'fun<NSUBJ>'} \\
\text{NSUBJ} & \left[\begin{array}{ll} \text{PRED} & \text{'writing'} \\ \text{PERSON} & \text{3RD} \\ \text{NUMBER} & \text{SINGULAR} \\ \text{GENDER} & \text{NEUTRAL} \end{array}\right] \\
\text{COP} & \left[\begin{array}{ll} \text{FORM} & \text{'is'} \\ \text{TENSE} & \text{PRESENT} \end{array}\right] \\
\text{PUNCT} & \left[\begin{array}{ll} \text{FORM} & \text{'.'} \\ \text{STMT-TYPE} & \text{DECLARATIVE} \end{array}\right]
\end{array}\right]\right]$$

Figure 5.95. The functional structure for ‘Writing is fun.’

In the figure below, the verb ‘Running’ is modified by a prepositional phrase, and it is analyzed as a verb that depends on ‘fun’ with the dependency type “csubj.”

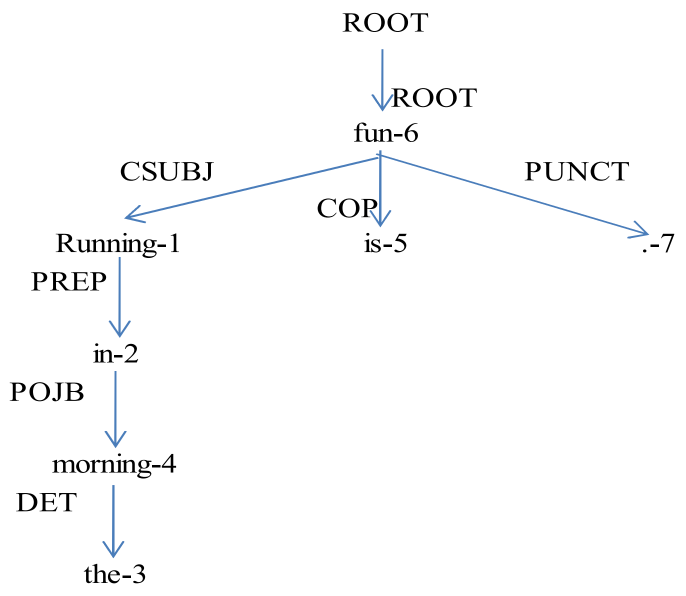


Figure 5.96. The typed-dependency tree for ‘Running in the morning is fun.’

```
     | PRED    'fun<CSUBJ>'
     | CSUBJ  | PRED    'running<NSUBJ,PREP>'
     |        | NSUBJ  | PRED     'PRO'
     |        |        | TYPE     ZERO
     |        |
     |        | PREP   | PRED     'in<POBJ>'
     |        |        | POBJ    | PRED      'morning'
     |        |        |         | PERSON    3RD
     |        |        |         | NUMBER    SINGULAR
     |        |        |         | GENDER    NEUTRAL
     |        |        |         | DET      | FORM 'the'
     |        |        |         |          | TYPE DEFINITE
     |
     | COP    | FORM    'is'
     |        | TENSE   PRESENT
     |
     | PUNCT  | FORM    '.'
ROOT |        | STMT-TYPE  DECLARATIVE
```

Figure 5.97. The functional structure for ‘Running in the morning is fun.’

When, on the other hand, there is no element modifying an intransitive present participle that depends on a verbal predicate and the present participle precedes the verbal predicate, the Stanford Parser analyses this participle as a nominal subject; hence, the dependency type is “nsubj.”

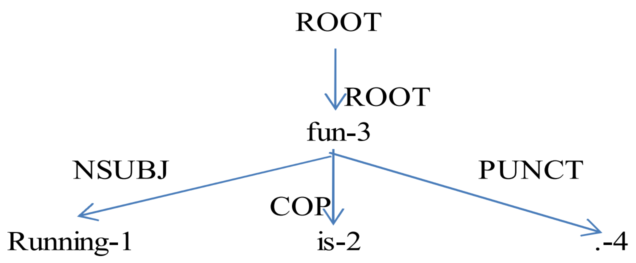


Figure 5.98. The typed-dependency tree for ‘Running is fun.’

```
|     |PRED    'fun<NSUBJ>'                    ||
|     |NSUBJ  |PRED       'running'    |       ||
|     |       |PERSON     3RD          |       ||
|     |       |NUMBER     SINGULAR     |       ||
|     |       |GENDER     NEUTRAL      |       ||
|     |                                        ||
|     |COP    |FORM       'is'         |       ||
|     |       |TENSE      PRESENT      |       ||
|     |                                        ||
|     |PUNCT  |FORM       '.'          |       ||
|ROOT |       |STMT-TYPE  DECLARATIVE  |       ||
```

Figure 5.99. The functional structure for ‘Running is fun.’

It must be pointed out that the Stanford Parser’s parsing policy on present-participle subjects shown above is not linguistically well-motivated at present. A present-participle subject with no other words dependent on it (e.g., ‘Running’ in the example just above) can be considered to be a clausal head, and therefore it can be analysed to depend on the main predicate (‘fun’ in the same example) with the dependency type “csubj.” Actually, with respect to Criterion C1[32], there seems to be no reason to distinguish nominal subjects and clausal subjects, because the semantic relationships which are implied by both of these dependency subtypes seem to be the same, and therefore we may only need the dependency type ‘subj’ for both of the cases. However, this parsing policy (labeling all of them as “subj”) may oversimplify the difference between the semantics of nominal subjects and clausal subjects, even though it is yet unclear to us. In this study, the distinction between “nsubj” and “csubj” is retained when parsing the sentences in different corpora (discussed in Chapter 7), for the sake of avoiding oversimplification. This policy does not deny the fact that the distinction of nominal subjects

[32] Criterion C1 states that a particular dependency type can imply a certain kind of semantic relationship between two words, and this semantic relationship is different from what is implied by another dependency type. See Section 2.4.3.3.

and clausal subjects by the Stanford Parser needs linguistic motivation, which is one of the issues to be addressed in future research.

Nsubjpass - passive nominal subject

This dependency type is a subtype of "subj" whereby the dependency head is a subject-taking element, the dependent is the nominal subject of the head, and the sentence is produced in the passive voice. The figure below presents an example of this dependency type (for the functional structure for the same sentence, see Figure 5.14).

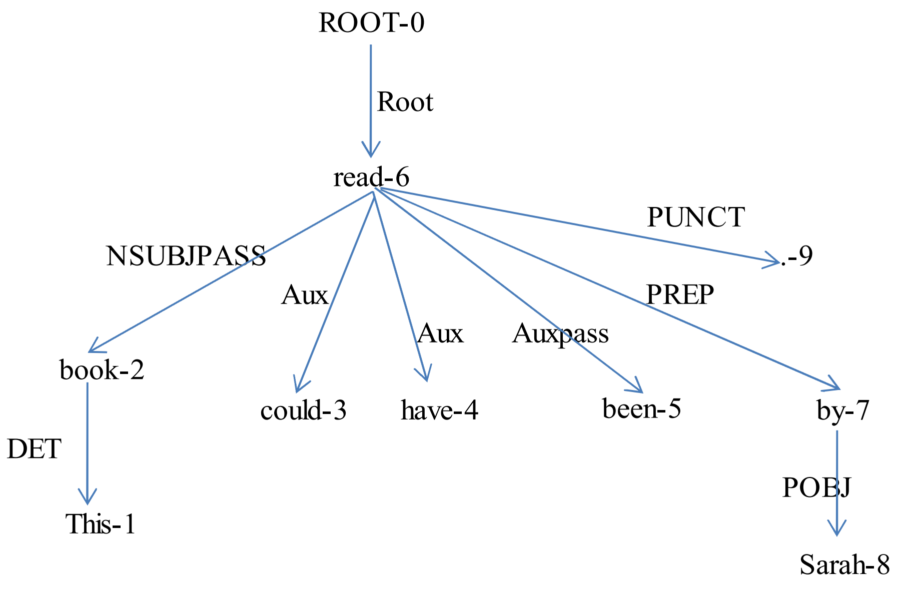


Figure 5.100. The typed-dependency tree for 'This could have been read by Sarah.'

De Marneffe & Manning (2012) do not specify the reason for distinguishing between the type "nsubj" and "nsubjpass." As is the distinction between "nsubj" and "csubj" mentioned in the previous subsection, we need to have a linguistic motivation to distinguish "nsubj" and "nsubjpass," with respect to Criterion C. We can infer that the different type names were intended to reflect the different meanings of active nominal subjects (i.e., the *agent* of the action) and passive nominal subjects (i.e., the *theme* of the action).

Csubjpass - passive clausal subject

This dependency type is a subtype of "subj" and describes cases where the dependency head is a subject-taking element, the dependent is the clausal subject of the head, and the sentence is produced in passive voice. The figure below presents an example of this dependency type. In the figure below, the verb 'lied' depends on the verb 'suspected' with the dependency type "csubjpass."

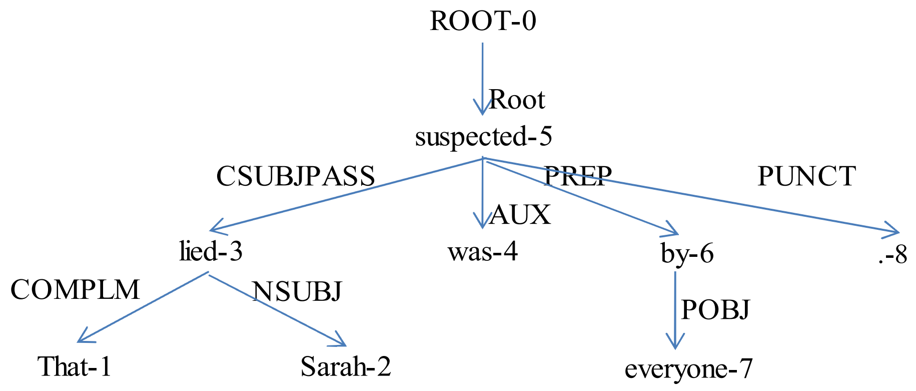


Figure 5.101. The typed-dependency tree for 'That Sarah lied was suspected by everyone.'

```
       PRED        'suspect<CSUBJPASS>'
       CSUBJPASS   PRED        'lie<NSUBJ>'
                   NSUBJ       PRED        'Sarah'
                               PERSON      3RD
                               NUMBER      SINGULAR
                               GENDER      FEMININE

                   COMPLM      FORM        'that'
       AUXPASS     FORM        'was'
                   TENSE       PAST
                   VOICE       PASSIVE
       PREP        PRED        'by<POBJ>'
                   POBJ        PRED        'PRO'
                               FORM        'everyone'
                               TYPE        PERSONAL
       PUNCT       FORM        '.'
ROOT               STMT-TYPE   DECLARATIVE
```

Figure 5.102. The functional structure for 'That Sarah lied was suspected by everyone.'

Cc - coordination

Coordination is "the relation between an element of a conjunct and the coordinating conjunction word of the conjunct" (de Marneffe & Manning 2012, p.4). Coordinating conjunction words include 'and' and 'or.' An example of "cc" is shown in the example presented below for "conj."

Conj - conjuncts

This dependency type 'is the relation between two elements connected by a coordinating conjunction' (de Marneffe & Manning 2012, p.4). In Stanford Parser output, conjuncts are treated *asymmetrically*; in other words, one conjunct depends on the other. The figure below presents an example of this type. In the phrase 'this book and that book,' the second 'book' (indicated as ‘book-7’ in the typed-dependency tree below) depends on the first ‘book’ (indicated as ‘book-5’ in the typed-dependency tree below).

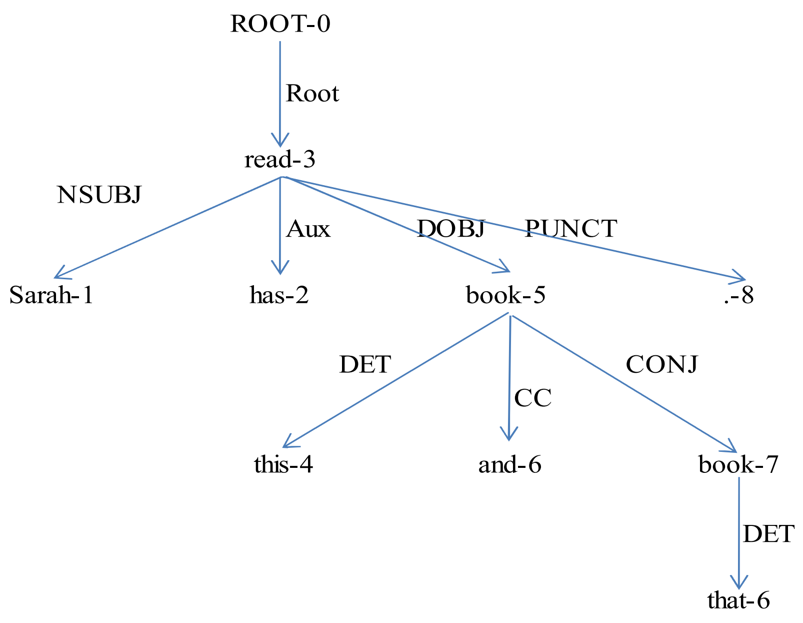


Figure 5.103. The asymmetric typed-dependency tree for ‘Sarah has read this book and that book.’

The correct typed-dependency tree, however, has a *symmetrical* structure in which both conjuncts depend on the same head with the same dependency type. As a result, the typed dependency “conj” should be replaced by other typed dependencies. For example, in the above tree, “conj” should be replaced by “dobj,” as shown in the figure below.

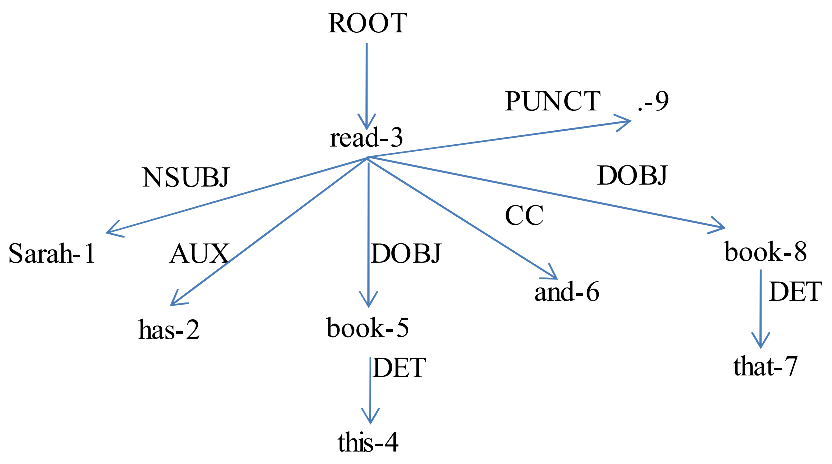


Figure 5.104. The symmetric typed-dependency tree for 'Sarah has read this book and that book.'

The typed-dependency tree above is equivalent to the functional structure below. The presence of 'and' ensures that the two direct objects do not result in a violation of the coherence constraint (see Section 3.2.2). The curled bracket around the local functional structures for 'this book' and 'that book' indicates that both of these local functional structures are the values of the one attribute "dobj."

```
      PRED    'read<NSUBJ,DOBJ>'
      NSUBJ   PRED      'Sarah'
              PERSON    3RD
              NUMBER    SINGULAR
              GENDER    FEMININE

      DOBJ    PRED      'book'
              PERSON    3RD
              NUMBER    SINGULAR
              GENDER    NEUTRAL
              DET       FORM      'this'
                        TYPE      DEFINITE
              CC        'and'
              PRED      'book'
              PERSON    3RD
              NUMBER    SINGULAR
              GENDER    NEUTRAL
              DET       FORM      'that'
                        TYPE      DEFINITE

      AUX     FORM      'has'
              TENSE     PRESENT
              ASPECT    PERFECT

      PUNCT   FORM      '.'
ROOT          STMT-TYPE DECLARATIVE
```

Figure 5.105. The functional structure for ‘Sarah has read this book and that book.’

If one of the conjuncts serves as the root of the sentence, so does the other conjunct, and in that case, both depend on the abstract element “ROOT,” as shown in Figures 5.106 and 5.107.

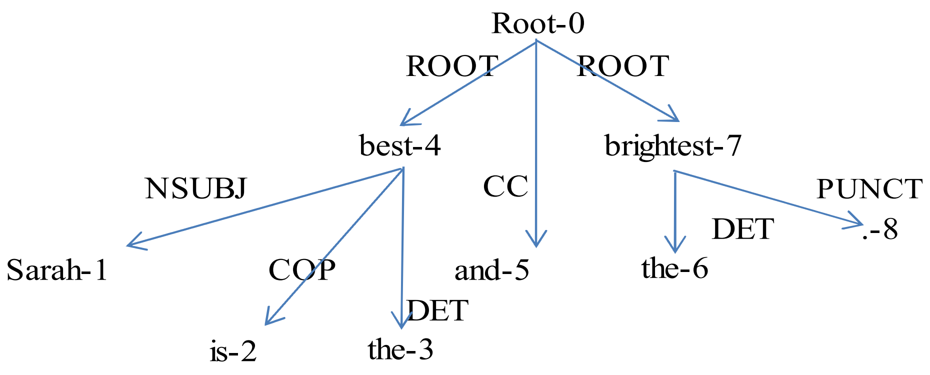


Figure 5.106. The typed-dependency tree for ‘Sarah is the best and the brightest.’

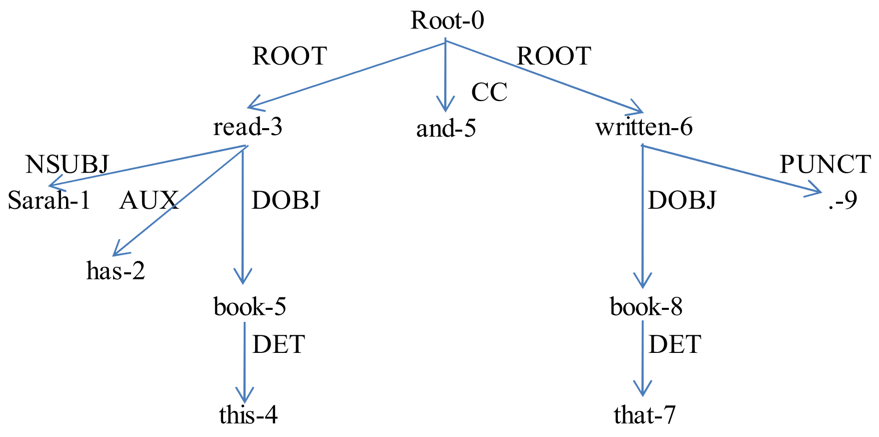


Figure 5.107. The typed-dependency tree for ‘Sarah has read this book and written this book.’

This study does not follow de Marneffe and Manning’s (2011) treatment of coordination in which conjuncts are aligned *asymmetrically* whereby the first conjunct is the head and the second conjunct is its dependent, and this second conjunct is another head and the third one is its dependent, and so on. This study does not use the option in the Stanford Parser for output in which the first two conjuncts are *propagated* so that they are aligned symmetrically to depend on one single head (see Section 5.4). This methodological decision was made because this option does not allow the third conjunct and later conjuncts to be propagated; therefore the output does not preserve tree structure (de Marneffe & Manning 2012).

In this study, the asymmetrical conjunct alignment in the Stanford Parser output is automatically fixed to have symmetrical conjunct alignment. Along with this adjustment, the typed dependency “conj” is replaced by another typed dependency according to the syntactic environment of the “conj” in the original parsed output. For example, when the first conjunct in the original Stanford Parser output is a direct object of a verb, then all the following conjuncts that depend on the first are adjusted to depend on the verb as direct objects. However, not all "conj" are replaced in this way, because the dependency between conjuncts and a preconjunct must be indicated. This issue will be discussed in more detail later in Section 5.3.4.

Expl - expletives

An existential 'there' depends on the head of a clause. The figure below is the typed-dependency tree for an example sentence in which an existential 'there' depends on 'are' with the type "expl." The noun 'books' depends on 'are' with the type "nsubj."

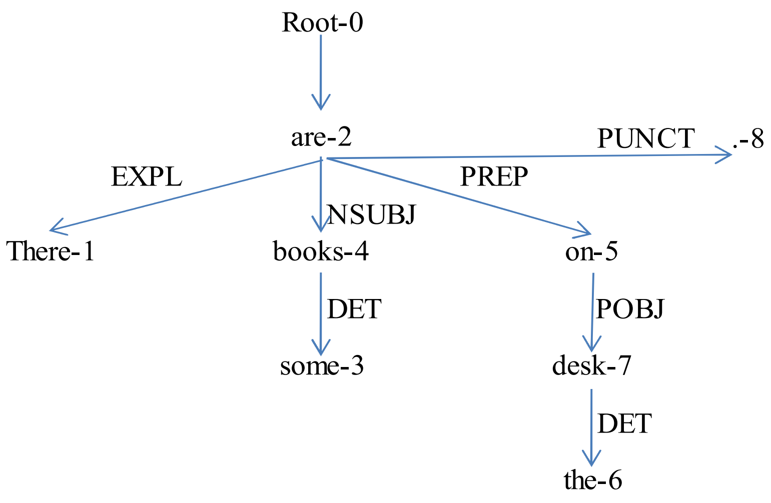


Figure 5.108. The typed-dependency tree for 'There are some books on the desk.'

```
|         |PRED     'be<NSUBJ,PREP>                                        ||
|         |                                                                ||
|         |EXPL    |FORM      'there'       |                              ||
|         |                                                                ||
|         |NSUBJ   |PRED      'books'                       |              ||
|         |        |PERSON    3RD                           |              ||
|         |        |NUMBER    PLURAL                        |              ||
|         |        |GENDER    NEUTRAL                       |              ||
|         |        |DET      |FORM       'some'        |    |              ||
|         |        |         |TYPE       INDEFINITE    |    |              ||
|         |                                                                ||
|         |PREP    |PRED      'on<POBJ>'                             |     ||
|         |        |POBJ     |PRED       'desk'                  |   |     ||
|         |        |         |PERSON     3RD                     |   |     ||
|         |        |         |NUMBER     SINGULAR                |   |     ||
|         |        |         |GENDER     NEUTRAL                 |   |     ||
|         |        |         |DET       |FORM      'the'     |   |   |     ||
|         |        |         |          |TYPE      DEFINITE  |   |   |     ||
|         |TENSE    PRESENT                                                ||
|         |PUNCT   |FORM       '.'          |                              ||
|ROOT     |        |STMT-TYPE  DECLARATIVE  |                              ||
```

Figure 5.109. The functional structure for ‘There are some books on the desk.’

When an existential ‘there’ depends on a verb ‘seem,’ and ‘seem’ is followed by ‘to be *noun*’ construction, then the noun depends on ‘seem’ with the type “xcomp.” The subject of this open complement is absent in the sentence.

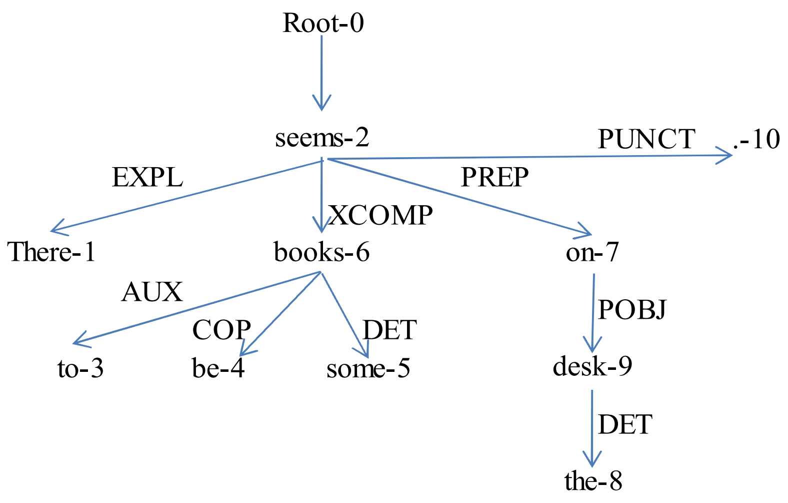


Figure 5.110. The typed-dependency tree for ‘There seems to be some books on the desk.’

```
|       |PRED    'seem<NSUBJ,PREP>'                                      ||
|       |                                                                 ||
|       |EXPL   |FORM      'there'     |                                  ||
|       |                                                                 ||
|       |XCOMP  |PRED      'books'                    |                   ||
|       |       |PERSON    3RD                        |                   ||
|       |       |NUMBER    PLURAL                     |                   ||
|       |       |GENDER    NEUTRAL                    |                   ||
|       |       |DET      |FORM       'some'      |   |                   ||
|       |       |         |TYPE       INDEFINITE  |   |                   ||
|       |       |                                     |                   ||
|       |       |AUX      |FORM 'to'   |              |                   ||
|       |       |                                     |                   ||
|       |       |COP      |FORM 'be'   |              |                   ||
|       |                                                                 ||
|       |PREP   |PRED      'on<POBJ>'                               |     ||
|       |       |POBJ     |PRED       'desk'                    |   |     ||
|       |       |         |PERSON     3RD                       |   |     ||
|       |       |         |NUMBER     SINGULAR                  |   |     ||
|       |       |         |GENDER     NEUTRAL                   |   |     ||
|       |       |         |DET       |FORM      'the'        |  |   |     ||
|       |       |         |          |TYPE      DEFINITE     |  |   |     ||
|       |TENSE   PRESENT                                                  ||
|       |PUNCT  |FORM      '.'         |                                  ||
|ROOT   |       |STMT-TYPE DECLARATIVE |                                  ||
```

Figure 5.111. the functional structure for ‘There seems to be some books on the table.’

The dependency type “expl” follows Mel’čuk’s criteria for SSyntRel. First, the head of a clause and existential ‘there’ form a prosodic unit (e.g., ‘there are’), and the expletive precedes the copula.

Second, the passive valence of the phrase ‘there are’ in the example above is determined by the proposition ‘are.’ According to Mel’čuk’s Criterion B1, the copula is the head and the existential ‘there’ is the dependent.

Third, this dependency type implies a unique semantic relationship between the governor and its dependent, and the prototypical dependent of this type is an existential ‘there.’ Therefore, this dependency type follows the revised Criteria C1 and C2 proposed in Section 2.4.3.3.

5.3.4 Mod - modifiers

The dependency type “mod” describes cases in which the dependent modifies the head. This dependency type has 25 subcategories, which are discussed in turn below.

Neg – negation modifier

The dependency type “neg” is a subtype of “mod” and represents cases where the dependent is a negation modifier, as shown in Figure 5.112 and Figure 5.113.

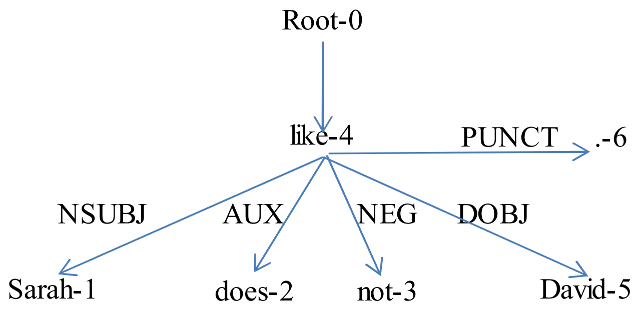


Figure 5.112. The typed-dependency tree for ‘Sarah does not like David.’

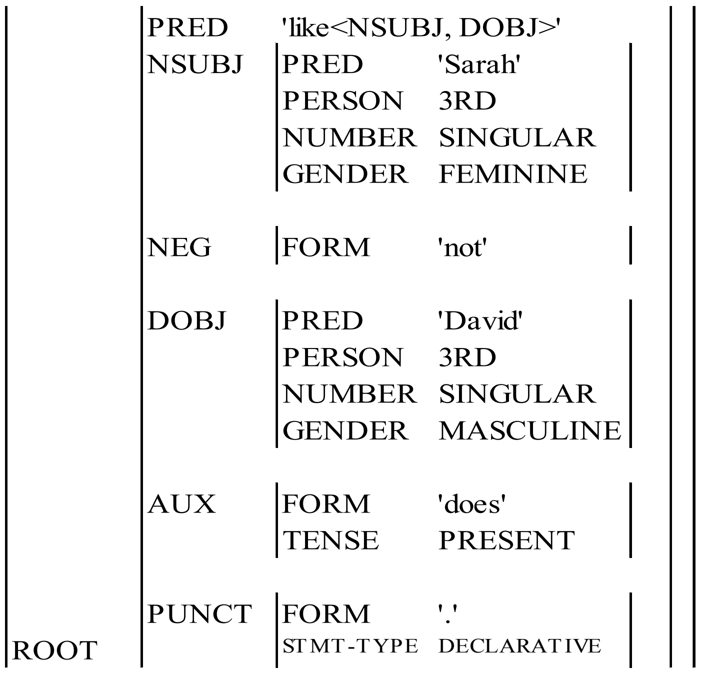


Figure 5.113. The functional structure for ‘Sarah does not like David.’

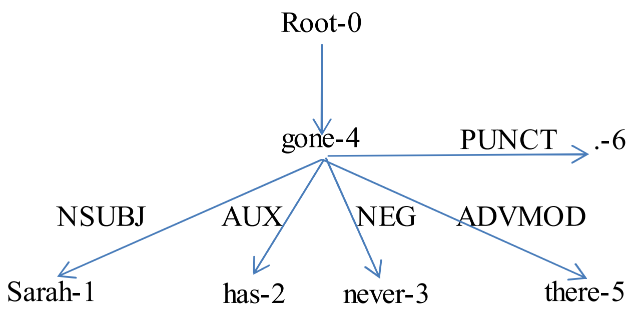


Figure 5.114. The typed-dependency tree for ‘Sarah has never gone there.’

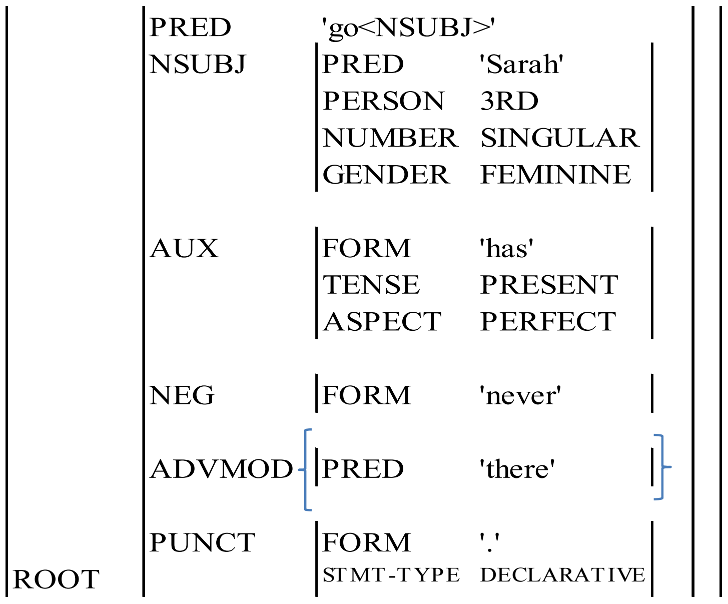

PRED 'go<NSUBJ>'
NSUBJ PRED 'Sarah' PERSON 3RD NUMBER SINGULAR GENDER FEMININE
AUX FORM 'has' TENSE PRESENT ASPECT PERFECT
NEG FORM 'never'
ADVMOD PRED 'there'
PUNCT FORM '.' STMT-TYPE DECLARATIVE
ROOT

Figure 5.115. The functional structure for ‘Sarah has never gone there.”

Dependencies between predicates and negative pronouns are not typed as “neg,” because negative pronouns are not negative modifiers, but rather arguments of the predicates.

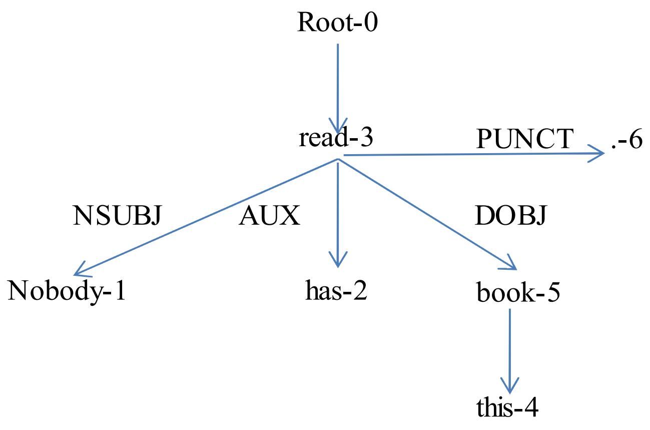


Figure 5.116. The typed-dependency tree for ‘Nobody has read this book.’

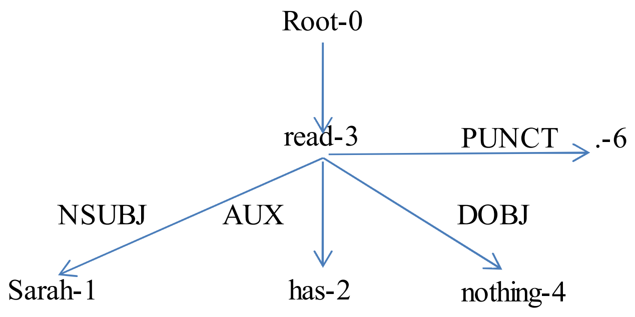


Figure 5.117. The typed-dependency tree for ‘Sarah has read nothing.’

There are some instances in which the dependency type “neg” does not follow Mel’čuk’s original Criterion A for SSyntRel. The linear order of the words in this dependency type is such that the negative modifier precedes the verb or adjective; however, the head of a clause and the negation modifier do not constitute a prosodic unit (e.g., ‘not like’ in the example above).

However, this type follows the revised Criterion A, proposed in Section 2.4.3.1, because the verb in the main clause and the subordinate verb form a semantic unit in which the semantic meaning of the verb is negated by the negative modifier.

This dependency type follows Mel’čuk’s Criterion B1, because the passive valence of the phrase is determined by the verb. For example, the passive valence of the phrase ‘not like’ is determined by the verb ‘like.’ According to Mel’čuk’s Criterion B1, the verb is the head and the negative modifier is the dependent.

This dependency type also follows the revised Criteria C1 and C2, proposed in Section 2.4.3.3, because it implies a unique semantic relationship between the governor and its dependent (i.e., the dependent negates the content of its governor), and the prototypical dependent of this dependency type is a negative modifier.

Det - determiner

The dependency type "det" is a subtype of "mod" and represents cases where the head is a noun and the dependent is a determiner. There are four types of determiners in English: definite/indefinite articles, demonstratives, interrogatives, and relatives.

English definite articles agree with the nouns they modify in terms of number (see Section 3.2.2 on the consistency constraint on functional structures), as shown in the following example.

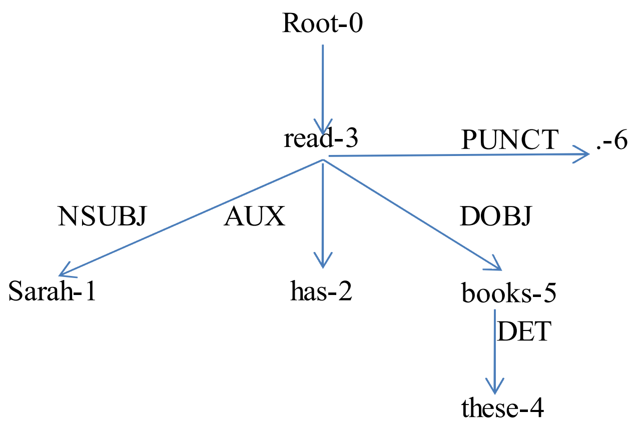


Figure 5.118. The typed-dependency tree for 'Sarah has read these books.'

```
|      |PRED    'read<NSUBJ, DOBJ>'                          ||
|      |NSUBJ  |PRED     'Sarah'    |                        ||
|      |       |PERSON   3RD        |                        ||
|      |       |NUMBER   SINGULAR   |                        ||
|      |       |GENDER   FEMININE   |                        ||
|      |                                                     ||
|      |DOBJ   |PRED     'books'                       |     ||
|      |       |PERSON   3RD                           |     ||
|      |       |NUMBER   PLURAL                        |     ||
|      |       |GENDER   NEUTRAL                       |     ||
|      |       |DET     |FORM       'these'      |    |     ||
|      |       |        |TYPE       DEFINITE     |    |     ||
|      |                                                     ||
|      |AUX    |FORM     'has'      |                        ||
|      |       |TENSE    PRESENT    |                        ||
|      |       |ASPECT   PERFECT    |                        ||
|      |                                                     ||
|      |PUNCT  |FORM      '.'        |                       ||
|ROOT  |       |STMT-TYPE DECLARATIVE|                       ||
```

Figure 5.119. The functional structure for ‘Sarah has read these books.’

In addition, the Stanford Parser analyzes the interrogative adjective ‘which’ as a determiner, as shown in the tree below.

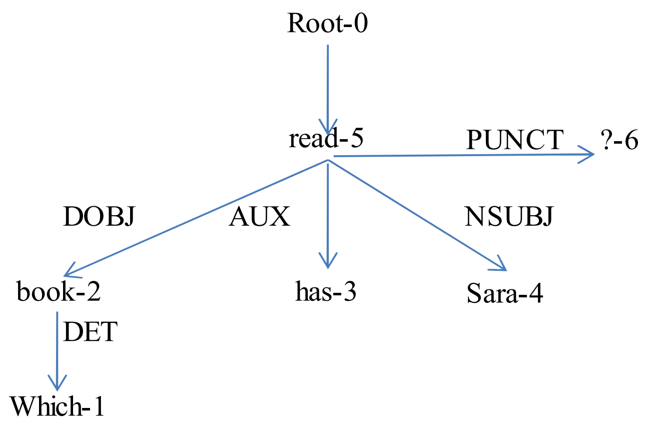


Figure 5.120. The typed-dependency tree for ‘Which book has Sarah read?’

```
        PRED    'read<NSUBJ, DOBJ>'
        NSUBJ   PRED     'Sarah'
                PERSON   3RD
                NUMBER   SINGULAR
                GENDER   FEMININE

        DOBJ    PRED     'book'
                PERSON   3RD
                NUMBER   SINGULAR
                GENDER   NEUTRAL
                DET      FORM       'which'
                         TYPE       INTERROGATIVE

        AUX     FORM     'has'
                TENSE    PRESENT
                ASPECT   PERFECT

        PUNCT   FORM     '?'
ROOT            STMT-TYPE  INTERROGATIVE
```

Figure 5.121. The functional structure for ‘Which book has Sarah read?’

The Stanford parser also analyzes compound relative pronouns as determiners. The dependency type “rcmod” in the tree below will be explained later in this section.

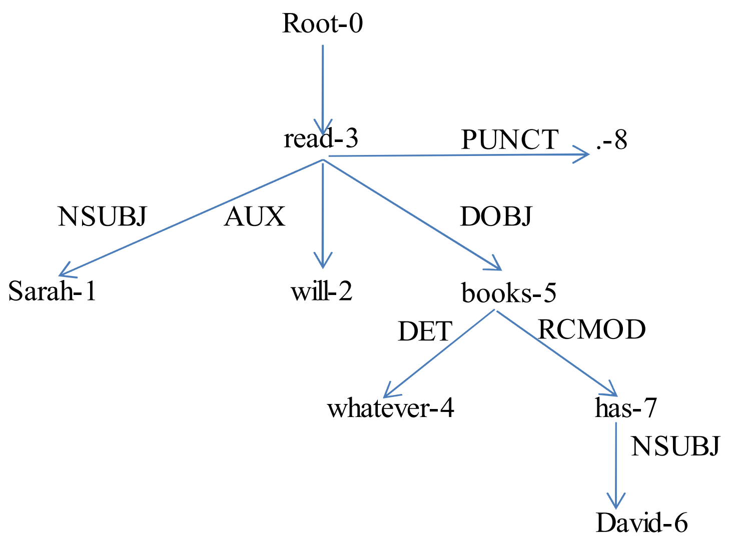


Figure 5.122. The typed-dependency tree for ‘Sarah will read whatever books David has.’

```
|      |PRED    'read<NSUBJ, DOBJ>'                                        ||
|      |NSUBJ  |PRED      'Sarah'   |                                      ||
|      |       |PERSON    3RD       |                                      ||
|      |       |NUMBER    SINGULAR  |                                      ||
|      |       |GENDER    FEMININE  |                                      ||
|      |                                                                   ||
|      |DOBJ   |PRED      'books_i'                                 |      ||
|      |       |PERSON    3RD                                       |      ||
|      |       |NUMBER    PLURAL                                    |      ||
|      |       |GENDER    NEUTRAL                                   |      ||
|      |       |DET      |FORM      'whatever'|                     |      ||
|      |       |         |TYPE      RELATIVE  |                     |      ||
|      |       |                                                    |      ||
|      |       |RCMOD    |PRED      'have<NSUBJ, DOBJ>'          |  |      ||
|      |       |         |NSUBJ    |PRED      'David'     |      |  |      ||
|      |       |         |         |PERSON    3RD         |      |  |      ||
|      |       |         |         |NUMBER    SINGULAR    |      |  |      ||
|      |       |         |         |GENDER    NEUTRAL     |      |  |      ||
|      |       |         |                                       |  |      ||
|      |       |         |DOBJ     |PRED      'PRO_i'     |      |  |      ||
|      |       |         |         |TYPE      ZERO        |      |  |      ||
|      |       |         |                                       |  |      ||
|      |       |         |TENSE     PRESENT                      |  |      ||
|      |                                                                   ||
|      |AUX    |FORM      'will'    |                                      ||
|      |       |TENSE     FUTURE    |                                      ||
|      |                                                                   ||
|      |PUNCT  |FORM      '.'       |                                      ||
|ROOT  |       |STMT-TYPE DECLARATIVE|                                     ||
```

Figure 5.123. The functional structure for ‘Sarah will read whatever books David has.’

This dependency type does not always follow Mel’čuk’s original Criterion A for SSyntRel because a noun and its determiner do not always constitute a prosodic unit (e.g., ‘the latest book’). However, this dependency type has a fixed linear order where the dependent always precedes the head and they form a semantic unit. In this way, this type follows the revised

Criterion A discussed in Section 2.4.3.1.

This dependency type clearly follows Mel'čuk's Criterion B1 because the noun determines the passive valence of the phrase. For example, the dependency relation between 'book' and 'this' in the sentence 'Sarah has read this book' shows that 'this' depends on the noun 'book' because this 'book' is a dependent on another word, namely 'read.'

In addition, this dependency type follows the revised Criterion C because it implies a certain kind of semantic relationship between the governor and the dependent, i.e., the dependent adds definite or indefinite information to the semantic meaning of the governor. This semantic relationship cannot be expressed by any other dependency type, and the prototypical dependent of this dependency type is a determiner. Therefore, this dependency type follows the revised Criteria C1 and C2 proposed in Section 2.4.3.3.

Prep - prepositional modifier

The dependency type "prep" is a subtype of "mod" and represents cases where the head is a noun or a verb and the dependent is a preposition. The fact that the head of this dependency can be either a noun or a verb results in ambiguous syntactic analyses for the same sentence. Let us consider the sentence 'Sarah has read the book in the room.' One analysis of this sentence is such that the prepositional phrase depends on the noun phrase preceding it; the typed-dependency tree as a result of this analysis is shown in the figure below.

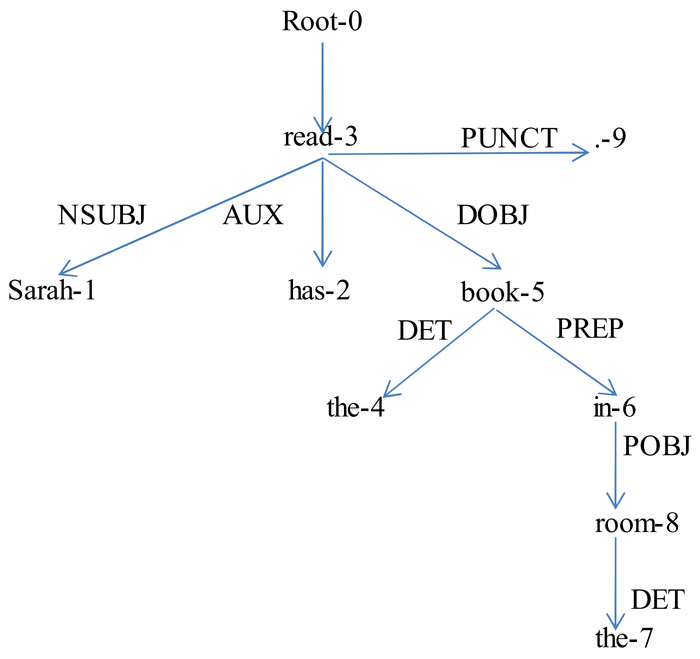


Figure 5.124. A typed-dependency tree for ‘Sarah has read the book in the room.’ (according to the default Stanford Parser output)

The typed-dependency tree above is equivalent to the functional structure below.

```
     | PRED    'read<NSUBJ, DOBJ>'                                          ||
     | NSUBJ  | PRED     'Sarah'     |                                      ||
     |        | PERSON   3RD         |                                      ||
     |        | NUMBER   SINGULAR    |                                      ||
     |        | GENDER   FEMININE    |                                      ||
     |                                                                      ||
     | DOBJ   | PRED     'book'                                        |    ||
     |        | PERSON   3RD                                           |    ||
     |        | NUMBER   SINGULAR                                      |    ||
     |        | GENDER   NEUTRAL                                       |    ||
     |        | DET     | FORM     'the'      |                        |    ||
     |        |         | TYPE     DEFINITE   |                        |    ||
     |        |                                                        |    ||
     |        | PREP    | PRED     'in<POBJ>'                       |  |    ||
     |        |         | POBJ    | PRED     'room'              |  |  |    ||
     |        |         |         | PERSON   3RD                 |  |  |    ||
     |        |         |         | NUMBER   SINGULAR            |  |  |    ||
     |        |         |         | GENDER   NEUTRAL             |  |  |    ||
     |        |         |         | DET     | FORM  'the'      |  |  |    ||
     |        |         |         |         | TYPE  DEFINITE | |  |  |    ||
     |                                                                      ||
     | AUX    | FORM     'has'       |                                      ||
     |        | TENSE    PRESENT     |                                      ||
     |        | ASPECT   PERFECT     |                                      ||
     |                                                                      ||
     | PUNCT  | FORM      '.'         |                                     ||
ROOT |        | STMT-TYPE DECLARATIVE |                                     ||
```

Figure 5.125. The functional structure for 'Sarah has read the book in the room.' equivalent to the typed-dependency tree in Figure 5.124

Another analysis would be that the prepositional phrase depends on the main verb of the sentence; the typed-dependency tree as a result of this analysis is shown in the figure below.

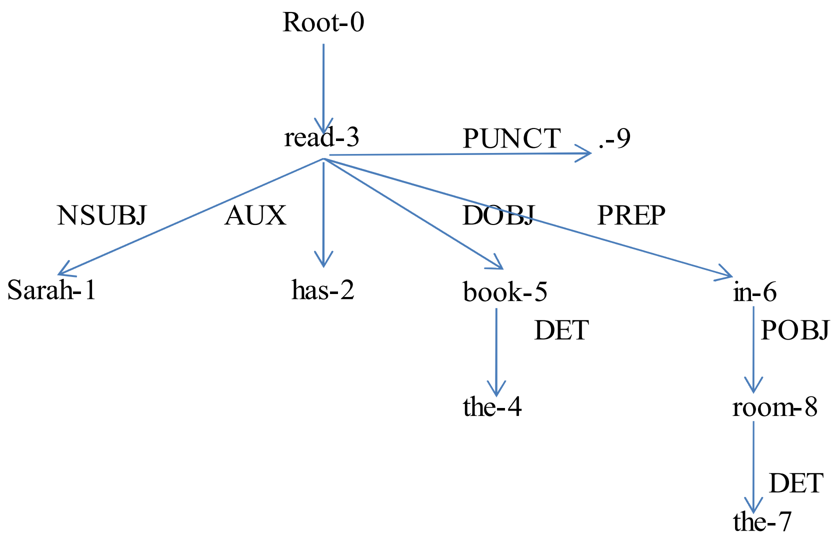


Figure 5.126. Another typed-dependency tree for ‘Sarah has read the book in the room.’

The typed-dependency tree above is equivalent to the functional structure below.

```
        PRED    'read<NSUBJ, DOBJ>'
        NSUBJ   PRED      'Sarah'
                PERSON    3RD
                NUMBER    SINGULAR
                GENDER    FEMININE

        DOBJ    PRED      'book'
                PERSON    3RD
                NUMBER    SINGULAR
                GENDER    NEUTRAL
                DET       FORM       'the'
                          TYPE       DEFINITE

        PREP    PRED      'in<POBJ>'
                POBJ      PRED       'room'
                          PERSON     3RD
                          NUMBER     SINGULAR
                          GENDER     NEUTRAL
                          DET        FORM     'the'
                                     TYPE     DEFINITE

        AUX     FORM      'has'
                TENSE     PRESENT
                ASPECT    PERFECT

        PUNCT   FORM      '.'
ROOT            STMT-TYPE DECLARATIVE
```

Figure 5.127. The functional structure for 'Sarah has read the book in the room.' equivalent to the typed-dependency tree in Figure 5.126

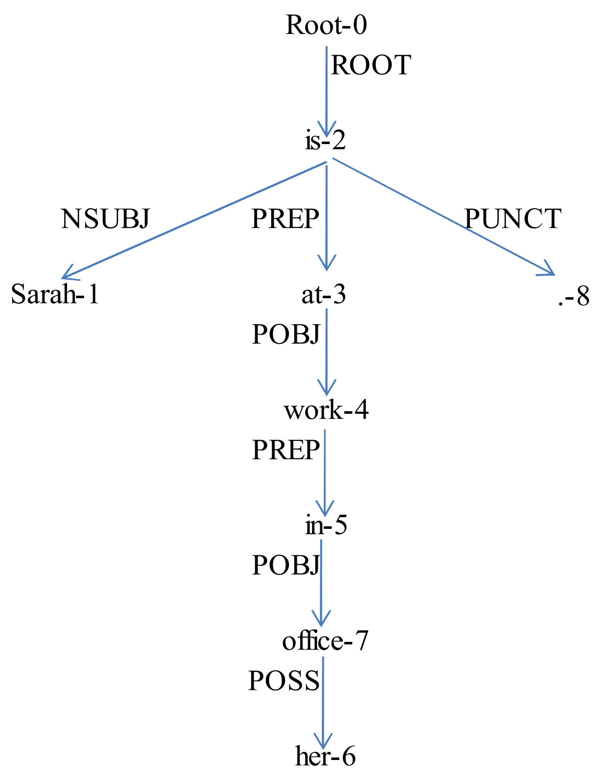


Figure 5.128. The typed-dependency tree for 'Sarah is at work in her office.' (according to the default Stanford Parser output)

```
      | PRED    'be<NSUBJ, PREP>'
      | NSUBJ  | PRED      'Sarah'    |
      |        | PERSON    3RD        |
      |        | NUMBER    SINGULAR   |
      |        | GENDER    FEMININE   |
      |
      | PREP   | PRED      'at<POBJ>'
      |        | POBJ     | PRED      'work'
      |        |          | PERSON    3RD
      |        |          | NUMBER    SINGULAR
      |        |          | GENDER    NEUTRAL
      |        |          | PREP     | PRED     'in<POBJ>'
      |        |          |          | POBJ    | PRED     'office'
      |        |          |          |         | PERSON   3RD
      |        |          |          |         | NUMBER   SINGULAR
      |        |          |          |         | GENDER   NEUTRAL
      |        |          |          |         | POSS    | FORM     'her'     |
      |        |          |          |         |         | PERSON   3RD       |
      |        |          |          |         |         | NUMBER   SINGULAR  |
      |        |          |          |         |         | GENDER   FEMININE  |
      | TENSE   PRESENT
      | PUNCT  | FORM      '.'         |
|ROOT |        | STMT-TYPE DECLARATIVE |
```

Figure 5.129. The functional structure for ‘Sarah is at work in her office.’ equivalent to the typed-dependency tree in Figure 5.128

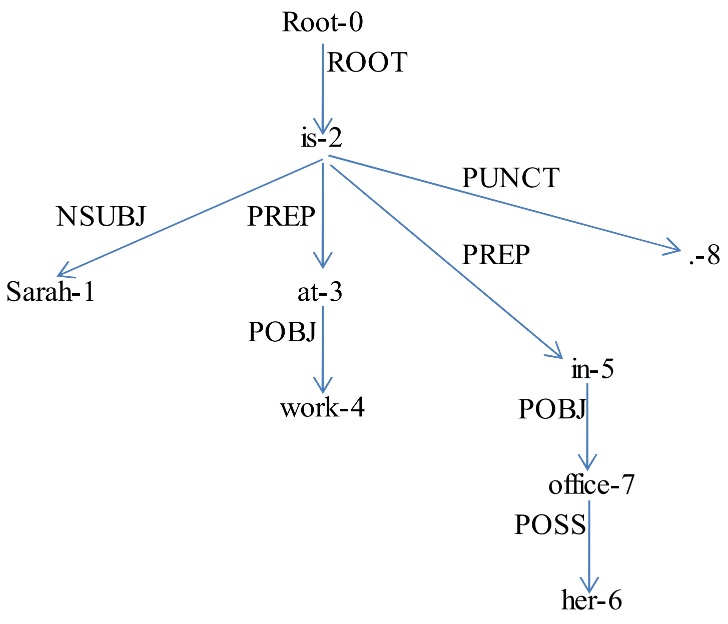


Figure 5.130. Another typed-dependency tree for ‘Sarah is at work in her office.’

```
        PRED    'be<NSUBJ, PREP>'
        NSUBJ   PRED      'Sarah'
                PERSON    3RD
                NUMBER    SINGULAR
                GENDER    FEMININE

        PREP    PRED      'at<POBJ>'
                POBJ      PRED      'work'
                          PERSON    3RD
                          NUMBER    SINGULAR
                          GENDER    NEUTRAL

        PREP    PRED      'in<POBJ>'
                POBJ      PRED      'office'
                          PERSON    3RD
                          NUMBER    SINGULAR
                          GENDER    NEUTRAL
                          POSS      FORM      'her'
                                    PERSON    3RD
                                    NUMBER    SINGULAR
                                    GENDER    FEMININE
        TENSE   PRESENT
        PUNCT   FORM      '.'
ROOT            STMT-TYPE DECLARATIVE
```

Figure 5.131. The functional structure for ‘Sarah is at work in her office.’ equivalent to the typed-dependency tree in Figure 5.130

The Stanford Parser yields only one parsed output for each input sentence by default, and one option (-printPCFGkBest *n*) allows it to yield n-best parses. However, the Stanford Parser does not determine which parse is the best fit for the context of a given sentence. Manually determining which parse is best will be a difficult task for researchers with large-scale corpora. In addition, it is impossible to determine which is the best parse if there is no context for an input sentence. Therefore, when sentences are parsed by the Stanford Parser in this study, the default setting is used in order to obtain only one parse for each sentence.

The dependency type “prep” does not follow Mel’čuk’s Criterion A for SSyntRel because a preposition and the word on which it depends do not constitute a prosodic unit (e.g., ‘book of’ or ‘know about’), and they are not aligned in a fixed word order in sentences whose heads are verbs. The prepositional phrase dependent on a verb can precede or follow the verb (e.g., ‘Sarah teaches English on Tuesday’ or ‘On Tuesday, Sarah teaches English’). However, this type forms a semantic unit, and therefore follows the revised Criterion A proposed in Section 2.4.3.1.

This dependency type also follows Mel’čuk’s Criterion B. In a dependency between a noun and a preposition, the noun determines the passive valence of the phrase. For example, the dependency relation between ‘book’ and ‘of’ in the sentence ‘Sarah has read the latest book of linguistics carefully’ shows that the preposition ‘of’ depends on the noun ‘book,’ which can be subordinated to the verb ‘read.’ The same logic applies for dependencies between verbs and prepositions.

In addition, this dependency type follows the revised Criterion C proposed in Section 2.4.3.3, because it implies a certain kind of semantic relationship between a preposition and the word it depends on, and this semantic relationship cannot be expressed by any other dependency type. Moreover, the prototypical dependent of this type is a preposition.

Prepc – prepositional clausal modifier

The dependency type “prepc” is a subtype of “mod” and represents cases where the head is an adjective, a noun or a verb and the dependent is a clause introduced by a preposition. This dependency type is used in the “collapsed” output style (See Section 5.4). Each instance of “prepc” is subtyped by the name of the preposition.

The figures below are examples of a typed-dependency tree and its corresponding functional structure with the dependency type “prepc” where the head is a verb and its tail is another verb. In the example below, the head of this type in this tree is a verb ‘read,’ and its tail is ‘using.’

The subject of ‘using’ is a zero pronoun that refers to the subject of ‘read,’ represented as a PRO in the functional structure.

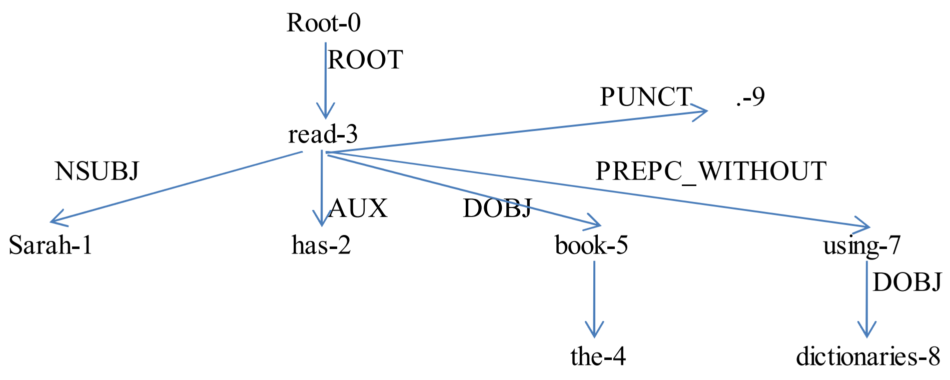


Figure 5.132. The typed-dependency tree for ‘Sarah has read the book without using dictionaries.’

```
       PRED      'read<NSUBJ, DOBJ>'
       NSUBJ     PRED     'Sarah'
                 PERSON   3RD
                 NUMBER   SINGULAR
                 GENDER   FEMININE

       DOBJ      PRED     'book'
                 PERSON   3RD
                 NUMBER   SINGULAR
                 GENDER   NEUTRAL
                 DET      FORM        this
                          TYPE        DEFINITE

       PREPC_    PRED     'use<NSUBJ,DOBJ>'
       WITHOUT   NSUBJ    PRED 'PRO'
                          TYPE ZERO

                 DOBJ     PRED        'dictionaries'
                          PERSON      3RD
                          NUMBER      PLURAL
                          GENDER      NEUTRAL

       AUX       FORM     'has'
                 TENSE    PRESENT
                 ASPECT   PERFECT

       PUNCT     FORM     '.'
ROOT             STMT-TYPE DECLARATIVE
```

Figure 5.133. The functional structure for ‘Sarah has read this book without using dictionaries.’

The figures below are examples of a typed-dependency tree and its corresponding functional structure with the dependency type “prepc” where the head is a noun and its tail is a verb. In the example below, the head of this type in this tree is a noun ‘difficulty’ and its tail is ‘understanding.’[33]

[33] As for the sentence “Sarah had no difficulty understanding the book.”, the word “understanding” depends on the noun “difficulty” with the dependency type “partmod” which will be defined later in this section.

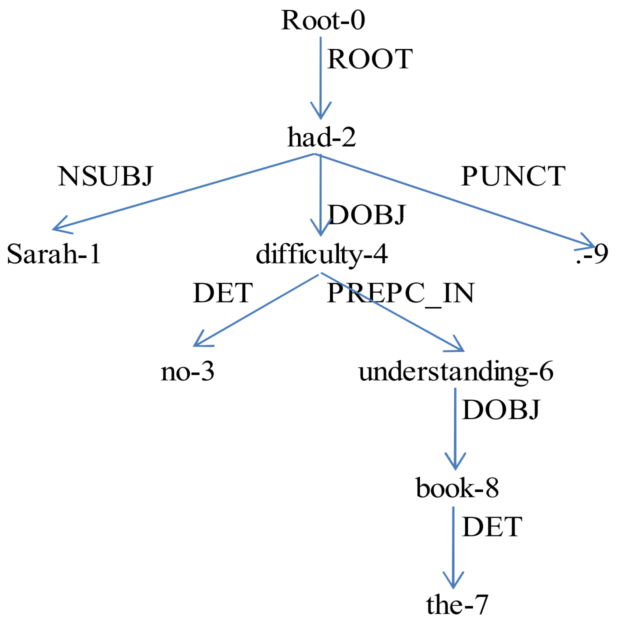


Figure 5.134. The typed-dependency tree for ‘Sarah had no difficulty in understanding the book.’

```
     PRED      'have<NSUBJ, DOBJ>'
     NSUBJ     PRED      'Sarah'
               PERSON    3RD
               NUMBER    SINGULAR
               GENDER    FEMININE

     DOBJ      PRED      'difficulty'
               PERSON    3RD
               NUMBER    SINGULAR
               GENDER    NEUTRAL
               DET       FORM        no
                         TYPE        NEGATIVE

               PREPC_    PRED        'understand<NSUBJ,DOBJ>'
               IN        NSUBJ       PRED 'PRO'
                                     TYPE ZERO

                         DOBJ        PRED       'book'
                                     PERSON     3RD
                                     NUMBER     SINGULAR
                                     GENDER     FEMININE
                                     DET        FORM     the
                                                TYPE     DEFINITE
     TENSE     PAST
     PUNCT     FORM      '.'
ROOT           STMT-TYPE DECLARATIVE
```

Figure 5.135. The functional structure for ‘Sarah had no difficulty in understanding the book.’

The next figures below are an example of a typed-dependency tree and its corresponding functional structure with the dependency type “prepc” where the head is an adjective and its tail is a verb. In the example below, the head is ‘responsible’ and its tail is ‘writing.’

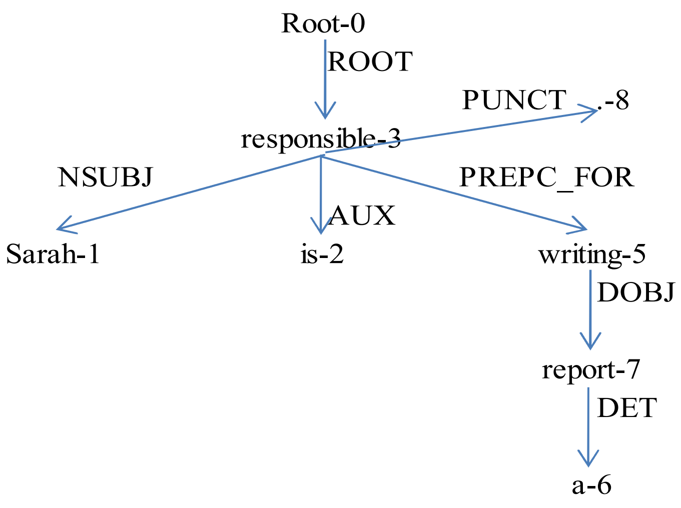
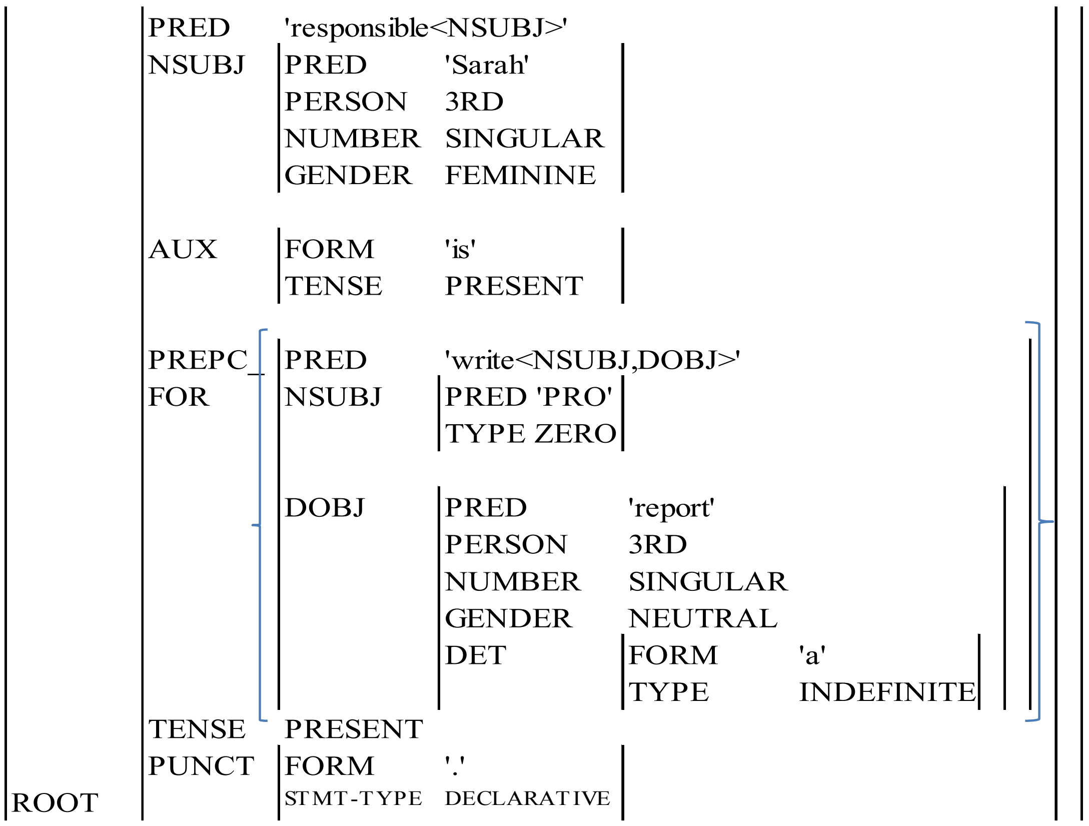


Figure 5.136. The typed-dependency tree for ‘Sarah is responsible for writing a report.’



```
       PRED      'responsible<NSUBJ>'
       NSUBJ     PRED      'Sarah'
                 PERSON    3RD
                 NUMBER    SINGULAR
                 GENDER    FEMININE

       AUX       FORM      'is'
                 TENSE     PRESENT

       PREPC_    PRED      'write<NSUBJ,DOBJ>'
       FOR       NSUBJ     PRED 'PRO'
                           TYPE ZERO

                 DOBJ      PRED        'report'
                           PERSON      3RD
                           NUMBER      SINGULAR
                           GENDER      NEUTRAL
                           DET         FORM      'a'
                                       TYPE      INDEFINITE
       TENSE     PRESENT
       PUNCT     FORM      '.'
ROOT             STMT-TYPE DECLARATIVE
```

Figure 5.137. The functional structure for ‘Sarah is responsible for writing a report.’

The dependency type “prepc” does not follow Mel’čuk’s Criterion A for SSyntRel because the head and its tail do not form a prosodic unit (e.g., read … using, difficulty … understanding, interested … studying). Moreover, they do not necessarily have a fixed order in sentences; for example, it is acceptable to say 'Without using a dictionary, Sarah has read the book.,' but not 'In understanding the book, Sarah had no difficulty.' nor 'In studying linguistics, Sarah is interested.' However, this type forms a semantic unit, thus following the revised Criterion A proposed in Section 2.4.3.1.

This dependency type follows Mel'čuk's Criterion B because the head determines the passive valence of the phrase. For example, for the dependency relation between 'read' and 'using' in the sentence 'Sarah has read the book without using a dictionary,' 'using' depends on the verb 'read,' which has the ability to be subordinated to the root of the sentence.

Additionally, this dependency type follows the revised Criterion C proposed in Section 2.4.3.3, because it implies a certain kind of semantic relationship between the head and its tail, and this semantic relationship cannot be expressed by any other dependency type. The prototypical dependent of this dependency type is a verb.

Amod - adjectival modifier

The dependency type "amod" is a subtype of "mod" and describes cases where the head is a noun and the dependent is an attributive adjective. In the figures below, the word 'interesting' depends on 'book' with the type 'amod.' In other words, the word 'interesting' modifies 'book' as an adjective.

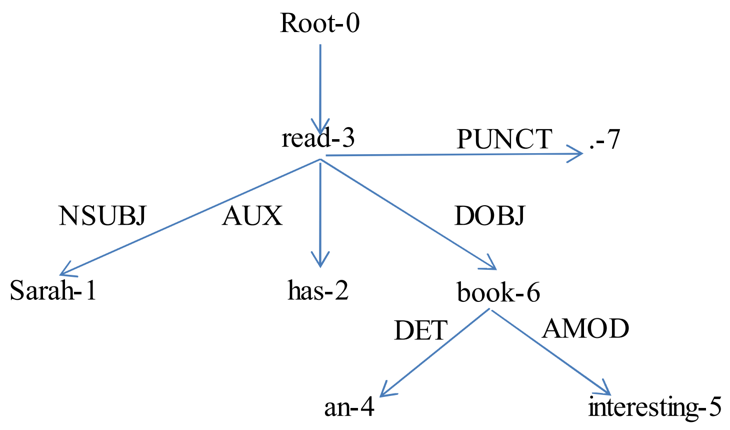


Figure 5.138. The typed-dependency tree for ‘Sarah has read an interesting book.’

```
|          |PRED    'read<NSUBJ, DOBJ>'                           ||
|          |NSUBJ  |PRED      'Sarah'    |                         ||
|          |       |PERSON    3RD        |                         ||
|          |       |NUMBER    SINGULAR   |                         ||
|          |       |GENDER    FEMININE   |                         ||
|          |                                                       ||
|          |DOBJ   |PRED      'book'                         |     ||
|          |       |PERSON    3RD                            |     ||
|          |       |NUMBER    SINGULAR                       |     ||
|          |       |GENDER    NEUTRAL                        |     ||
|          |       |DET      |FORM     'an'          |       |     ||
|          |       |         |TYPE     INDEFINITE    |       |     ||
|          |       |                                         |     ||
|          |       |AMOD     |PRED     'interesting' |       |     ||
|          |                                                       ||
|          |AUX    |FORM      'has'      |                         ||
|          |       |TENSE     PRESENT    |                         ||
|          |       |ASPECT    PERFECT    |                         ||
|          |                                                       ||
|          |PUNCT  |FORM      '.'        |                         ||
|ROOT      |       |STMT-TYPE DECLARATIVE                          ||
```

Figure 5.139. The functional structure for ‘Sarah has read an interesting book.’

Attributive adjectives can follow ‘something,’ ‘somebody,’ or ‘someone.’ The word ‘interesting’ depends on the word ‘something’ with the dependency type “amod.”

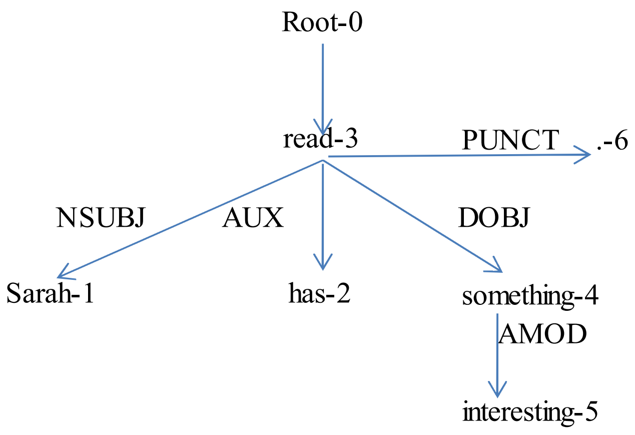


Figure 5.140. The typed-dependency tree for ‘Sarah has read something interesting.’

When an attributive adjective modifies ‘something,’ ‘someone’ or ‘somebody’ that is the object of a predicate, the Stanford Parser incorrectly analyzes the adjective as an open complement that takes ‘something’ (in this example) as its nominal subject.

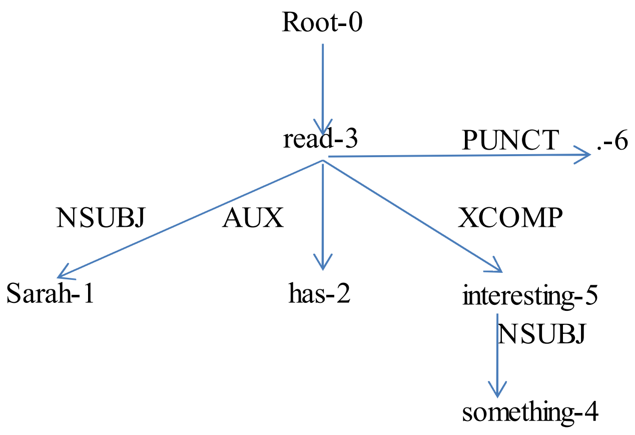


Figure 5.141. The incorrect typed-dependency tree for ‘Sarah has read something interesting’ in the Stanford Parser output

The dependency type “amod” follows Mel’čuk’s Criterion A for SSyntRel because a noun and its adjective modifier form a prosodic unit (e.g., 'latest book'), and they have a fixed order in sentences (attributive adjectives precede the nouns on which they depend, and they follow words such as 'something,' 'someone,' or 'somebody'). Moreover, this type forms a semantic unit, thus following the revised Criterion A proposed in Section 2.4.3.1.

This dependency type also follows Mel'čuk's Criterion B because the noun determines the passive valence of the phrase. For example, for the dependency relation between 'book' and 'latest' in the sentence 'Sarah has read the latest book of linguistics carefully,' it is 'latest' that depends on the noun 'book,' which has the ability to be subordinated to the verb 'read.'

Additionally, this dependency type follows the revised Criterion C proposed in Section 2.4.3.3, because it implies a certain kind of semantic relationship between a noun and an adjective, and this semantic relationship cannot be expressed by any other dependency type. The prototypical dependent of this dependency type is an adjective.

Advmod - adverbial modifier

The dependency type "advmod" is a subtype of "mod" and represents cases where the dependent is an adverb.

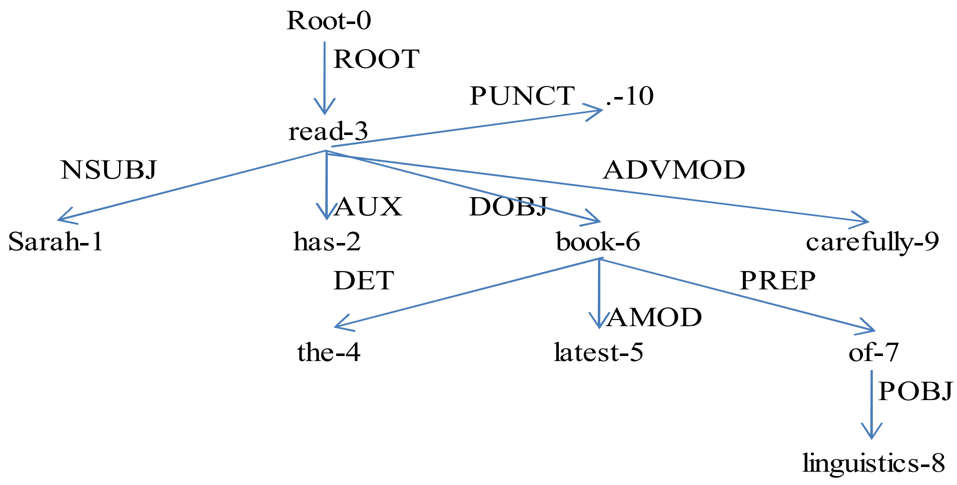


Figure 5.142. The typed-dependency tree for ‘Sarah has read the latest book of linguistics carefully.’

```
      PRED     'read<NSUBJ, DOBJ>'
      NSUBJ    PRED     'Sarah'
               PERSON   3RD
               NUMBER   SINGULAR
               GENDER   FEMININE

      DOBJ     PRED     'book'
               PERSON   3RD
               NUMBER   SINGULAR
               GENDER   NEUTRAL
               DET      FORM       'the'
                        TYPE       DEFINITE

               AMOD   { PRED       'latest' }

               PREP     PRED       'of<POBJ>'
                        POBJ       PRED      'linguistics'
                                   PERSON    3RD
                                   NUMBER    SINGULAR
                                   GENDER    NEUTRAL
      ADVMOD { PRED     'carefully' }

      AUX      FORM     'has'
               TENSE    PRESENT
               ASPECT   PERFECT

      PUNCT    FORM     '.'
ROOT           STMT-TYPE DECLARATIVE
```

Figure 5.143. The functional structure for ‘Sarah has read the latest book of linguistics carefully.’

The difference between sentential adverbs and verbal adverbs are not represented in the output of the Stanford Parser. For example, in both Figure 5.144 and Figure 5.145, the adverb ‘Naturally’ depends on the verb ‘speaks.’

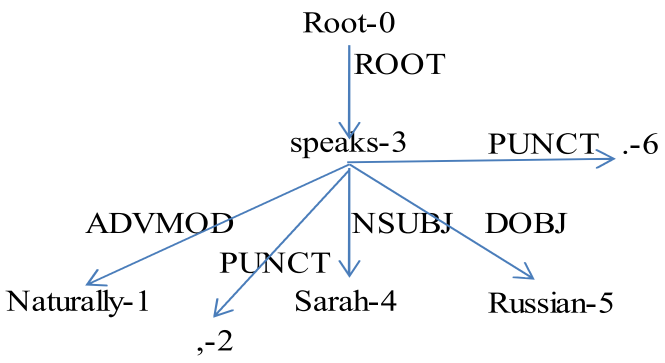


Figure 5.144. The typed-dependency tree for ‘Naturally, Sarah speaks Russian’ in the output of the Stanford Parser.

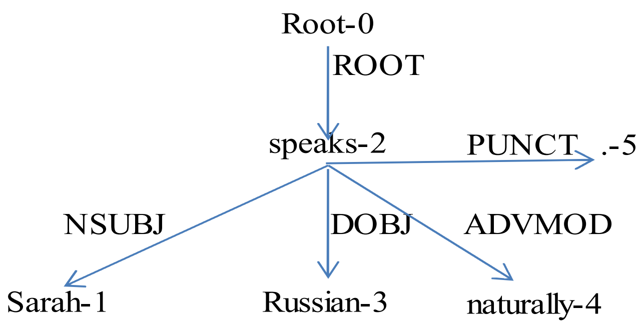


Figure 5.145. The typed-dependency tree for ‘Sarah speaks Russian naturally’ in the output of the Stanford Parser.

It is possible to consider sentential adverbials as being dependent on Root, so that it can modify the whole sentence. In the typed-dependency tree below, the adverb ‘Naturally’ depends on Root, modifying the whole sentence.

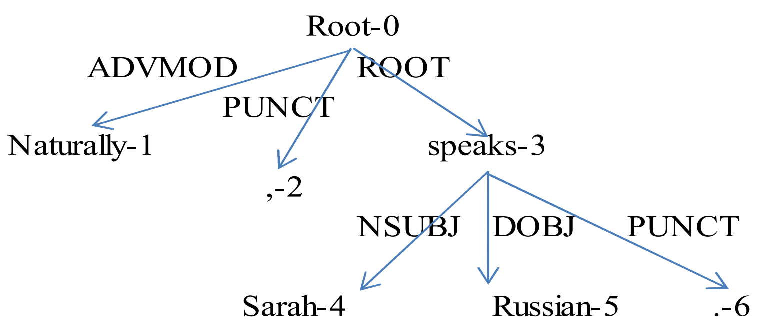


Figure 5.146. A possible typed-dependency tree for ‘Naturally, Sarah speaks Russian.’

```
|        | PRED     'speak<NSUBJ, DOBJ>'          | |
|        | NSUBJ   | PRED       'Sarah'       |   | |
|        |         | PERSON     3RD           |   | |
|        |         | NUMBER     SINGULAR      |   | |
|        |         | GENDER     FEMININE      |   | |
|        |                                        | |
|        | DOBJ    | PRED       'Russian'     |   | |
|        |         | PERSON     3RD           |   | |
|        |         | NUMBER     SINGULAR      |   | |
|        |         | GENDER     NEUTRAL       |   | |
|        |                                        | |
|        | TENSE    PRESENT                       | |
|        |                                        | |
|        | PUNCT   | FORM       '.'           |   | |
| ROOT   |         | STMT-TYPE  DECLARATIVE   |   | |
|                                                  |
| PUNCT  | FORM     ','          |                 |
| ADVMOD {| PRED    'naturally'  |}                |
```

Figure 5.147. The functional structure for ‘Naturally, Sarah speaks Russian.’

The dependency type “advmod” does not follow Mel’čuk’s Criterion A, because a verb and its adverb modifier do not always constitute a prosodic unit (e.g., ‘read the book carefully’), and the linear order of an adverb and a verb is not fixed. However, this dependency type follows the revised Criterion A because a verb and its adverb modifier form a semantic unit.

This dependency type follows Mel’čuk’s Criterion B because the verb determines the passive valence of the phrase. For example, the dependency relation between ‘read’ and ‘carefully’ in the sentence ‘Sarah has read the latest book of linguistics carefully,’ shows that ‘carefully’ depends on the verb ‘read,’ which can be subordinated to the root node.

The dependency type “advmod” also follows the revised Criterion C proposed in Section 2.4.3.3, because it implies a certain kind of semantic relationship between a verb and an adverb, and this semantic relationship cannot be expressed by any other dependency type. The prototypical dependent of this dependency type is an adverb.

Poss (possession modifier) and Possessive (possessive modifier)

The dependency type “poss” is a subtype of “mod” whereby the dependent is a possession modifier, and the dependency type “possessive” is also a subtype of “mod” and describes cases where the dependent is a possessive modifier ‘’s.’ The example sentences of “poss” and “possessive” are shown below.

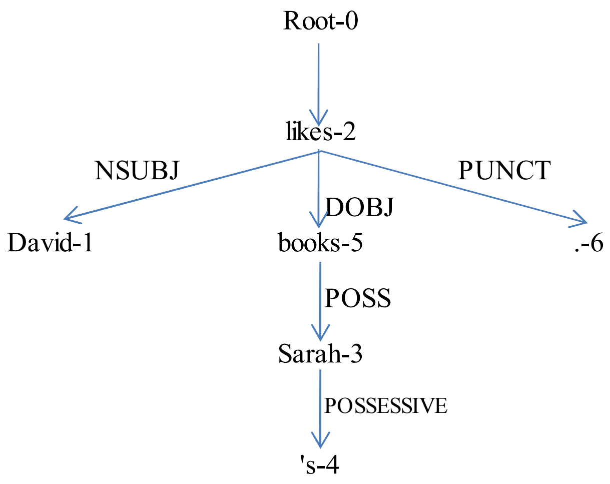


Figure 5.148. The typed-dependency tree for ‘David likes Sarah’s books.’

```
     | PRED    'like<NSUBJ,DOBJ>'                                  | |
     | NSUBJ  | PRED        'David'      |                         | |
     |        | PERSON      3RD          |                         | |
     |        | NUMBER      SINGULAR     |                         | |
     |        | GENDER      MASCULINE    |                         | |
     |                                                             | |
     | DOBJ   | PRED        'books'                              | | |
     |        | PERSON      3RD                                  | | |
     |        | NUMBER      PLURAL                               | | |
     |        | GENDER      NEUTRAL                              | | |
     |        | POSS       | PRED         'Sarah'            |   | | |
     |        |            | PERSON       3RD                |   | | |
     |        |            | NUMBER       SINGULAR           |   | | |
     |        |            | GENDER       FEMININE           |   | | |
     |        |            | POSSESSIVE  | FORM     's'  |   |   | | |
     | TENSE    PRESENT                                            | |
     | PUNCT  | FORM        '.'          |                         | |
|ROOT |        | STMT-TYPE   DECLARATIVE  |                         | |
```

Figure 5.149. The functional structure for ‘David likes Sarah’s books.’

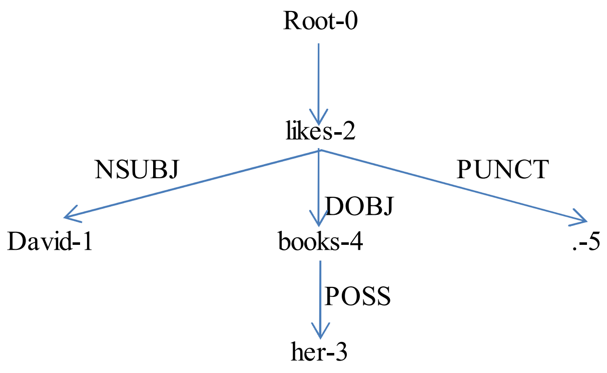


Figure 5.150. The typed-dependency tree for ‘David likes her books.’

```
|            |PRED     'like<NSUBJ,DOBJ>'                                ||
|            |NSUBJ   |PRED        'David'      |                        ||
|            |        |PERSON      3RD          |                        ||
|            |        |NUMBER      SINGULAR     |                        ||
|            |        |GENDER      MASCULINE    |                        ||
|            |                                                          ||
|            |DOBJ    |PRED        'books'                      |        ||
|            |        |PERSON      3RD                          |        ||
|            |        |NUMBER      PLURAL                       |        ||
|            |        |GENDER      NEUTRAL                      |        ||
|            |        |POSS       |PRED       'PRO'        |    |        ||
|            |        |           |PERSON     3RD          |    |        ||
|            |        |           |NUMBER     SINGULAR     |    |        ||
|            |        |           |GENDER     FEMININE     |    |        ||
|            |        |           |FORM       'her'        |    |        ||
|            |TENSE   PRESENT                                            ||
|            |PUNCT   |FORM        '.'          |                        ||
|ROOT        |        |STMT-TYPE   DECLARATIVE  |                        ||
```

Figure 5.151. The functional structure for ‘David likes her books.’

A noun with the possessive modifier ‘’s’ can function as a possessive pronoun. For example, ‘Sarah’s’ in the typed-dependency tree below refers to something Sarah possesses. The Stanford Parser yields the following typed-dependency tree in which the possessor is the direct object of the verb.

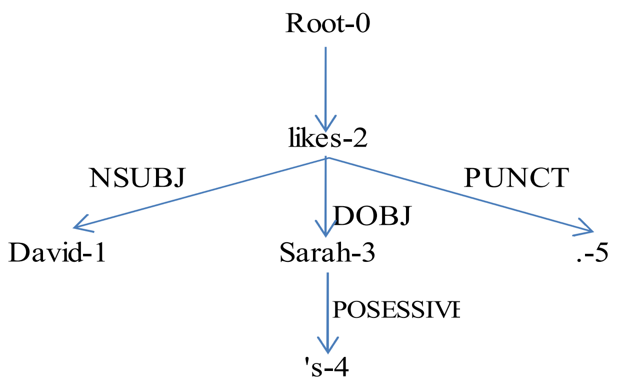


Figure 5.152. The typed-dependency tree for ‘David likes Sarah’s.’

The typed-dependency tree above is equivalent to the functional structure below.

```
|          |PRED     'like<NSUBJ,DOBJ>'                       | |
|          |NSUBJ    |PRED         'David'      |             | |
|          |         |PERSON       3RD          |             | |
|          |         |NUMBER       SINGULAR     |             | |
|          |         |GENDER       MASCULINE    |             | |
|          |                                                  | |
|          |DOBJ     |PRED         'Sarah'               |    | |
|          |         |PERSON       3RD                   |    | |
|          |         |NUMBER       SINGULAR              |    | |
|          |         |GENDER       FEMININE              |    | |
|          |         |POSSESSIVE  |FORM   's'  |       |    | |
|          |TENSE    PRESENT                              | |
|          |PUNCT    |FORM         '.'          |             | |
|ROOT      |         |STMT-TYPE    DECLARATIVE  |             | |
```

Figure 5.153. The functional structure for ‘David likes Sarah’s.’

Another possible functional structure is that the phrase ‘Sarah’s’ depends on a zero pronoun which refers to the thing Sarah possess.

```
|          |PRED     'like<NSUBJ,DOBJ>'                                        | |
|          |NSUBJ    |PRED         'David'      |                              | |
|          |         |PERSON       3RD          |                              | |
|          |         |NUMBER       SINGULAR     |                              | |
|          |         |GENDER       MASCULINE    |                              | |
|          |                                                                   | |
|          |DOBJ     |PRED         'PRO'                                   |   | |
|          |         |TYPE         ZERO                                    |   | |
|          |         |POSS         |PRED         'Sarah'             |     |   | |
|          |         |             |PERSON       3RD                 |     |   | |
|          |         |             |NUMBER       SINGULAR            |     |   | |
|          |         |             |GENDER       FEMININE            |     |   | |
|          |         |             |POSSESSIVE  |FORM    's'  |      |     |   | |
|          |TENSE    PRESENT                                                   | |
|          |PUNCT    |FORM         '.'          |                              | |
|ROOT      |         |STMT-TYPE    DECLARATIVE  |                              | |
```

Figure 5.154. Another functional structure for ‘David likes Sarah’s.’

Notice, however, that the functional structure above is not equivalent to the typed-dependency tree that the Stanford Parser yields, because the Stanford Parser output does not contain zero pronouns. Therefore, this study does not apply the zero-pronoun analysis for nouns with the possessive modifier '’s' functioning as a possessive pronoun, as shown in the functional structure above.

Possessive absolute pronouns are not categorized as dependents with the type “poss” or “possessive”, because they are directly dependent on a predicate as arguments, as shown below. The number and gender of the possessee (the thing that is possessed) are underspecified, hence not included in the functional structure.

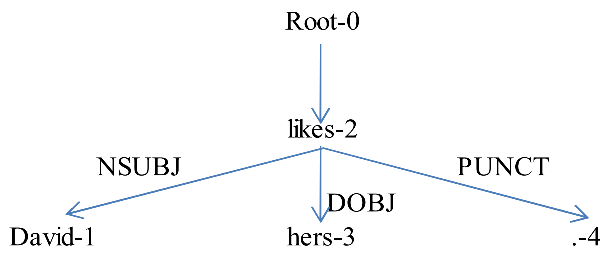


Figure 5.155. The typed-dependency tree for ‘David likes hers.’

```
|         |PRED     'like<NSUBJ,DOBJ>'                ||
|         |NSUBJ   |PRED        'David'      |        ||
|         |        |PERSON      3RD          |        ||
|         |        |NUMBER      SINGULAR     |        ||
|         |        |GENDER      MASCULINE    |        ||
|         |                                           ||
|         |DOBJ    |PRED        'PRO'        |        ||
|         |        |PERSON      3RD          |        ||
|         |        |FORM        'hers'       |        ||
|         |TENSE    PRESENT                           ||
|         |PUNCT   |FORM        '.'          |        ||
|ROOT     |        |STMT-TYPE   DECLARATIVE  |        ||
```

Figure 5.156. The functional structure for ‘David likes hers.’

The dependency type “poss” does not completely follow Mel’čuk’s Criterion A. The head and the dependent have a fixed linear order (the possessive modifier precedes the noun it modifies); however they do not constitute a prosodic unit (e.g., ‘Sarah book’ in the example sentence in Figure 5.148 and Figure 5.149). On the other hand, they do form a semantic unit, and thus follow the revised Criterion A proposed in Section 2.4.3.1.

This dependency type follows Mel’čuk’s criterion B. The noun, not its possession modifier, determines the passive valence of the phrase. For example, the dependency relation between ‘Sarah’ and ‘books’ in the sentence ‘David likes Sarah’s books’ shows that ‘Sarah’ depends on the noun ‘books,’ which can be subordinated to the verb ‘likes.’

The dependency type “poss” also follows the revised Criterion C proposed in Section 2.4.3.3, because it implies a certain kind of semantic relationship between nouns, i.e., possession. This semantic relationship cannot be expressed by any other dependency type, and the prototypical dependent of this dependency type is a noun with a possessive modifier.

The dependency type “possessive” follows Mel’čuk’s Criterion A because a noun and its possessive modifier constitute a prosodic unit (e.g., ‘Sarah’s’), and they have a fixed linear order (the noun precedes the possessive modifier).

In addition, this dependency type follows Mel’čuk’s Criterion B, because the noun determines the passive valence of the phrase. For example, the dependency relation between ‘Sarah’ and ‘’s’ in the sentence ‘David likes Sarah’s books’ illustrates that ‘’s’ depends on the noun ‘Sarah,’ which can be subordinated to the noun ‘books.’

The dependency type “possessive” also follows the Criterion C proposed in Section 2.4.3.3, because it implies a certain kind of semantic relationship between a noun and its possessive modifier, and this semantic relationship cannot be expressed by any other dependency type. Moreover, the prototypical dependent of this dependency type is the possessive modifier.

Infmod - infinitival modifier

The dependency type "infmod" is a subtype of "mod" whereby the head is a noun and the dependent is an infinitival modifier. The preposition 'to' is analyzed as an auxiliary, and the dependency between the head verb and the preposition 'to' is typed as "aux." An example of this type is presented in the typed-dependency tree below, in which the word 'say' modifies the word 'something' as an infinitival modifier. In other words, the word 'say' in this figure depends on the word 'something' with the dependency type "infmod."

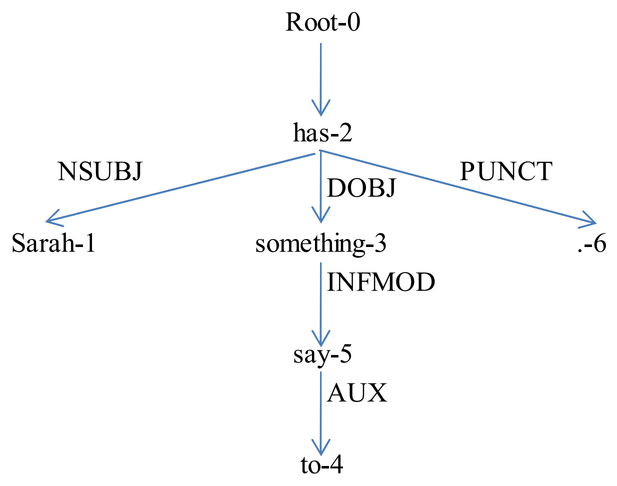


Figure 5.157. The typed-dependency tree for 'Sarah has something to say.'

The typed-dependency tree above is equivalent to the functional structure below. The same indices 'i' on 'Sarah' and on the zero pronoun as the NSUBJ of the verb 'say' indicate that this zero pronoun refers to 'Sarah.' The same indices 'j' on 'something' and on the zero pronoun as the DOBJ of the verb ‘say’ indicate that this zero pronoun refers to ‘something.’

```
      PRED     'have<NSUBJ,DOBJ>'
      NSUBJ    PRED 'Sarah$_i$'
               PERSON    3RD
               NUMBER    SINGULAR
               GENDER    FEMININE

      DOBJ     PRED      'PRO'
               PERSON    3RD
               NUMBER    SINGULAR
               GENDER    NEUTRAL
               FORM      'something$_j$'
               INFMOD    PRED       'say<NSUBJ,DOBJ>'
                         NSUBJ      PRED    'PRO$_i$'
                                    TYPE    ZERO

                         DOBJ       PRED    'PRO$_j$'
                                    TYPE    ZERO

                         AUX        FORM    'to'
      TENSE    PRESENT
      PUCNT    FORM      '.'
ROOT           STMT-TYPE DECLARATIVE
```

Figure 5.158. The functional structure for ‘Sarah has something to say.’

Infinitival modifiers can modify interrogative pronouns such as ‘what,’ ‘where’ to constitute phrases such as ‘what to do’ or ‘where to go.’ In these cases, the infinitival modifier depends on the interrogative pronoun with the dependency type “infmod”, as shown in Figure 5.159 and Figure 5.161. These typed-dependency trees are equivalent to the functional structures in Figure 5.160 and Figure 5.162, respectively.

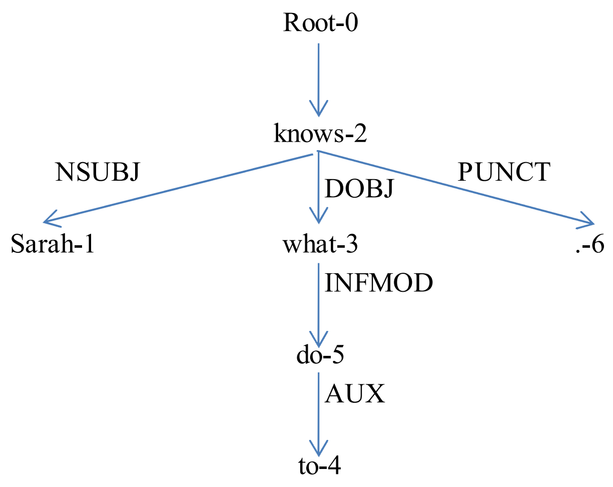


Figure 5.159. The typed-dependency tree for ‘Sarah knows what to do.’

```
      |PRED    'know<NSUBJ,DOBJ>'                                                  ||
      |NSUBJ  |PRED 'Sarah_i'          |                                           ||
      |       |PERSON    3RD           |                                           ||
      |       |NUMBER    SINGULAR      |                                           ||
      |       |GENDER    FEMININE      |                                           ||
      |                                                                            ||
      |DOBJ   |PRED      'PRO_j'                                                  |||
      |       |PERSON    3RD                                                      |||
      |       |NUMBER    SINGULAR                                                 |||
      |       |GENDER    NEUTRAL                                                  |||
      |       |FORM      'what'                                                  ]|||
      |       |INFMOD  [|PRED       'do<NSUBJ,DOBJ>'                            | |||
      |       |         |NSUBJ     |PRED     'PRO_i'  |                         | |||
      |       |         |          |TYPE     ZERO     |                         | |||
      |       |        {|                                                       |}|||
      |       |         |DOBJ      |PRED     'PRO_j'  |                         | |||
      |       |         |          |TYPE     ZERO     |                         | |||
      |       |         |                                                       | |||
      |       |        [|AUX       |FORM     'to'  |                            |]|||
      |TENSE   PRESENT                                                             ||
      |PUCNT  |FORM      '.'                  |                                    ||
ROOT  |       |STMT-TYPE  DECLARATIVE         |                                    ||
```

Figure 5.160. The functional structure for ‘Sarah knows what to do.’

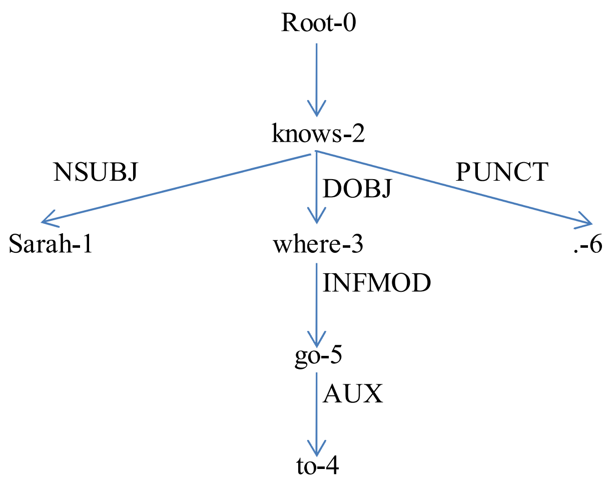


Figure 5.161. The typed-dependency tree for ‘Sarah knows where to go.’

```
     | PRED    'know<NSUBJ,DOBJ>'                                         | |
     | NSUBJ  | PRED 'Sarah_i'        |                                   | |
     |        | PERSON    3RD         |                                   | |
     |        | NUMBER    SINGULAR    |                                   | |
     |        | GENDER    FEMININE    |                                   | |
     |                                                                    | |
     | DOBJ   | PRED      'PRO_j'                                      |  | |
     |        | PERSON    3RD                                          |  | |
     |        | NUMBER    SINGULAR                                     |  | |
     |        | GENDER    NEUTRAL                                      |  | |
     |        | FORM      'where'                                      |  | |
     |        | INFMOD  [ | PRED      'go<NSUBJ,PREP>'           | ] |  | |
     |        |         [ | NSUBJ    | PRED     'PRO_i'  |       | ] |  | |
     |        |         [ |          | TYPE     ZERO     |       | ] |  | |
     |        |         [ |                                      | ] |  | |
     |        |         [ | PREP     | PRED     'PRO_j'  |       | ] |  | |
     |        |         [ |          | TYPE     ZERO     |       | ] |  | |
     |        |         [ |                                      | ] |  | |
     |        |         [ | AUX      | FORM     'to'  |          | ] |  | |
     | TENSE   PRESENT                                                    | |
     | PUCNT  | FORM      '.'                  |                          | |
ROOT |        | STMT-TYPE  DECLARATIVE         |                          | |
```

Figure 5.162. The functional structure for ‘Sarah knows where to go.’

The Stanford Parser does not parse infinitival modifiers correctly when they depend on interrogative pronouns. As the figure below shows, the infinitival modifier 'do' is incorrectly parsed to depend on 'know' with the dependency type "ccomp," and the interrogative pronoun 'what' is incorrectly parsed to depend on 'do' with an incorrect dependency type "nsubj."

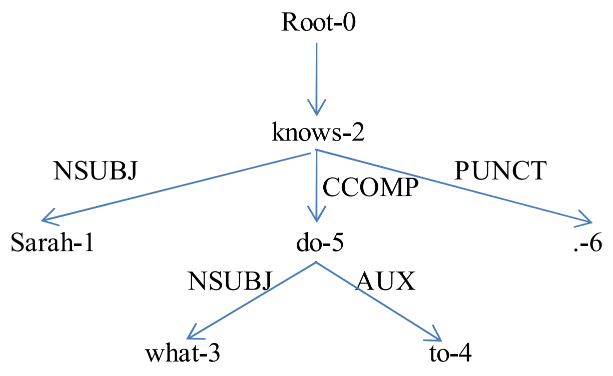


Figure 5.163. The incorrect typed-dependency tree for 'Sarah knows what to do' in the Stanford Parser output.

The figure below shows that the infinitival modifier 'go' is incorrectly parsed to depend on 'know' with the dependency type "xcomp," and the interrogative pronoun 'where' is incorrectly parsed to depend on 'go' with an incorrect dependency type "advmod."

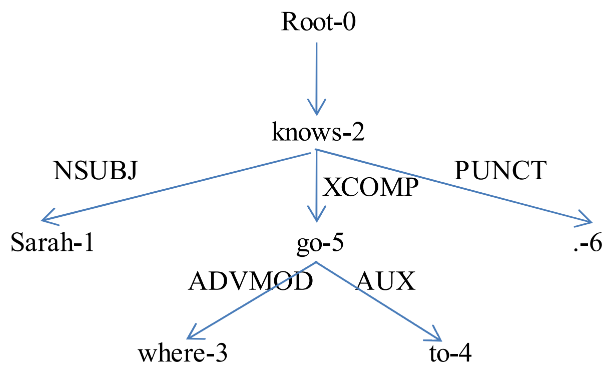


Figure 5.164. The incorrect typed-dependency tree for 'Sarah knows where to go' in the Stanford Parser output.

This dependency type does not completely follow Mel'čuk's Criterion A. The head and the dependent have a fixed linear order; however, they do not constitute a prosodic unit (e.g., 'something say' in the examples above). On the other hand, they do form a semantic unit and follow the revised Criterion A proposed in Section 2.4.3.1.

This dependency type follows Mel'čuk's Criterion B. The noun, not its infinitival modifier, determines the passive valence of the phrase. For example, the dependency relation between 'something' and 'say' in the sentence 'Sarah has something to say' shows that 'something' depends on the verb 'has,' which can be subordinated to the root.

The dependency type "infmod" also follows the revised Criterion C proposed in Section 2.4.3.3, because it implies a certain kind of semantic relationship between a noun and an infinitival modifier. This semantic relationship cannot be expressed by any other dependency type, and the prototypical dependent of this dependency type is an infinitival verb.

Partmod - participial modifier

The dependency type "partmod" is a subtype of "mod" whereby the head is a noun or a verb, and the dependent is a participial modifier. In the typed-dependency tree below, the past participle 'written' modifies 'essay' as a participial modifier. In other words, the word 'written' depends on 'essay' with the dependency type "partmod."

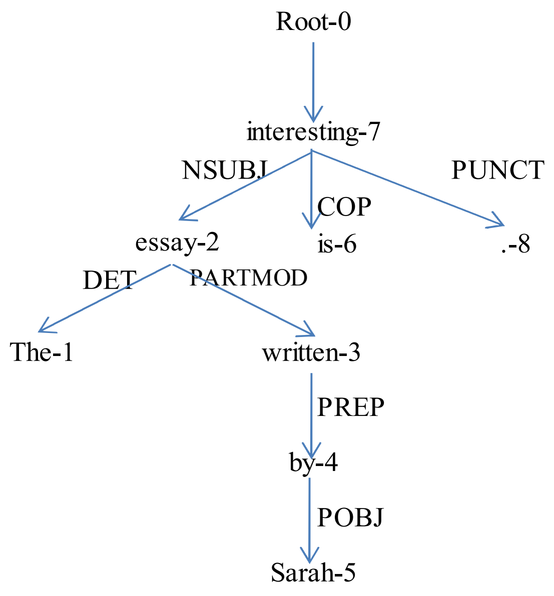


Figure 5.165. The typed-dependency tree for ‘The essay written by Sarah is interesting.’

The typed-dependency tree above is equivalent to the functional structure below. Notice that the same indices ‘i’ on ‘essay’ and on the zero pronoun as the NSUBJ of the past participle ‘written’ indicate that this zero pronoun refers to ‘essay;’ hence, this ‘essay’ is also the NSUBJ of ‘written.’

```
      PRED    'interesting<NSUBJ,>'
      NSUBJ   PRED      'essay_i'
              PERSON    3RD
              NUMBER    SINGULAR
              GENDER    NEUTRAL
              DET       FORM      'the'
                        TYPE      DEFINITE

              PARTMOD   PRED      'written<NSUBJ>'
                        NSUBJ     PRED     'PRO_i'
                                  TYPE     ZERO

                        PREP      PRED     'by<POBJ>'
                                  POBJ     PRED     'Sarah'
                                           PERSON   3RD
                                           NUMBER   SINGULAR
                                           GENDER   FEMININE
      COP     FORM      'is'
              TENSE     PRESENT

      PUCNT   FORM      '.'
ROOT          STMT-TYPE DECLARATIVE
```

Figure 5.166. The functional structure for 'The essay written by Sarah is interesting.'

In the typed-dependency tree below, the present participle 'writing' modifies the verb 'thinking' as a participial modifier. In other words, the word 'writing' depends on 'thinking' with the dependency type "partmod."

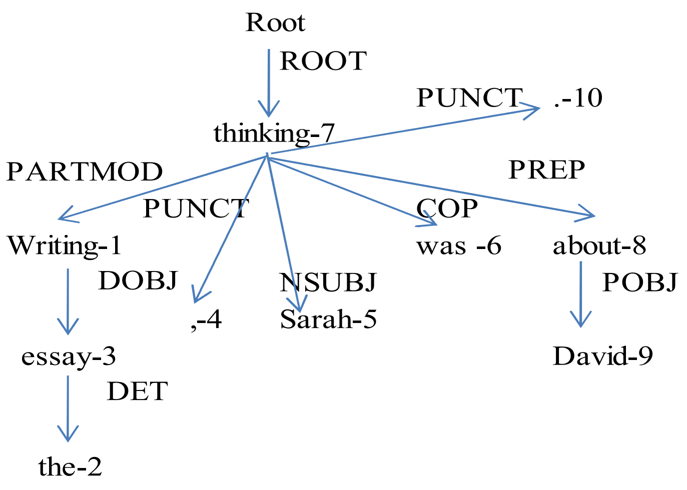


Figure 5.167. The typed-dependency tree for ‘Writing the essay, Sarah was thinking about David.’

This dependency type does not follow Mel’čuk’s Criterion A because a noun and its participial modifier do not constitute a prosodic unit (e.g., ‘Writing thinking’ in the example sentence above). However, they do form a semantic unit and therefore follow the revised Criterion A proposed in Section 2.4.3.1.

This dependency type follows Mel’čuk’s Criterion B because the noun, not its participial modifier, determines the passive valence of the phrase. For example, the dependency relation between ‘essay’ and ‘written’ in the sentence ‘Writing the essay, Sarah was thinking about David’ is such that ‘Writing’ depends on the verb ‘thinking,’ which can be subordinated to the root.

This dependency type also follows the revised Criterion C proposed in Section 2.4.3.3, because it implies a certain kind of semantic relationship between a noun and a participial modifier. This semantic relationship cannot be expressed by any other dependency type, and the prototypical dependent of this dependency type is a participle.

Advcl - adverbial clause modifier

Mark - markers (words introducing an adverbial clause)

The dependency type "advcl" is a subtype of "mod" whereby the head is a verb and the dependent is the main predicate of an adverbial clause that modifies the verb. Adverbial clauses are often introduced by a marker. The dependency type "mark" is another subtype of "mod" whereby the head is the main predicate of an adverbial clause and the dependent functions as a marker, such as 'before,' 'after,' 'because' or 'if.' Examples of both subtypes are shown in the figure below. The marker 'before' introduces the adverbial clause headed by 'visited.' This verb modifies the main verb 'read.' In other words, the word 'before' depends on 'visited' with the dependency type "mark," and 'visited' depends on 'read' with the dependency type "advcl."

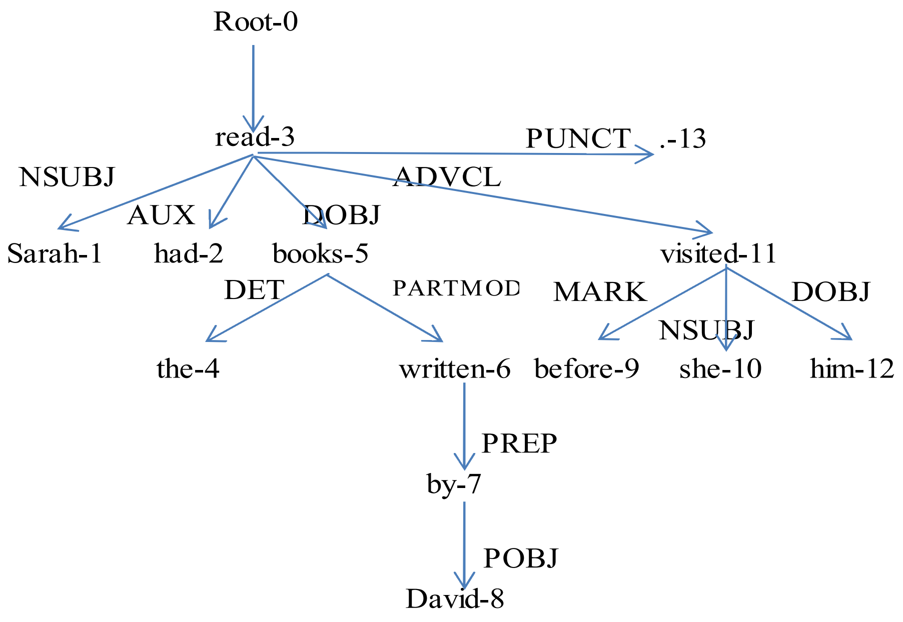


Figure 5.168. The typed-dependency tree for 'Sarah had read the books written by David before she visited him.'

In the figure below, the marker ‘because’ introduces the adverbial clause headed by ‘admires.’ This verb modifies the main verb ‘read.’ In other words, the word ‘because’ depends on ‘admires’ with the dependency type “mark,” and ‘admires’ depends on ‘read’ with the dependency type “advcl.”

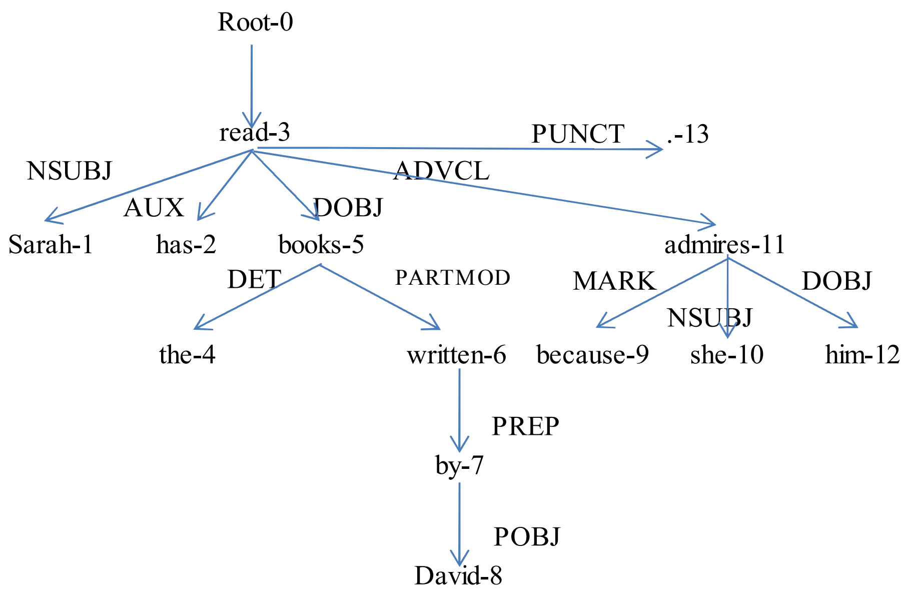


Figure 5.169. The typed-dependency tree for ‘Sarah has read the books written by David because she admires him.’

The Stanford Parser analyzes clauses introduced by the marker ‘if’ as adverbial clauses, and this can cause an incorrect parse. In the sentence ‘Sarah didn’t know if it was true,’ the word ‘true’ is parsed to be the main predicate of a clausal complement of the word ‘know.’ Therefore, ‘true’ depends on ‘know’ with the dependency type “ccomp,” and the word ‘if’ depends on ‘true’ with the dependency type “complm.”

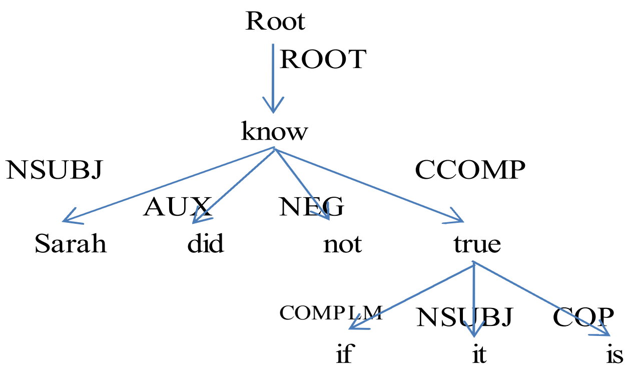


Figure 5.170. The typed-dependency tree for ‘Sarah didn’t know if it was true.’

In the Stanford Parser output for the same sentence, however, the word ‘true’ is parsed to be the main predicate of an adverbial complement of the word ‘know.’ Therefore, ‘true’ depends on ‘know’ with an incorrect dependency type “advcl,” and the word ‘if’ depends on ‘true’ with an incorrect dependency type “mark.”

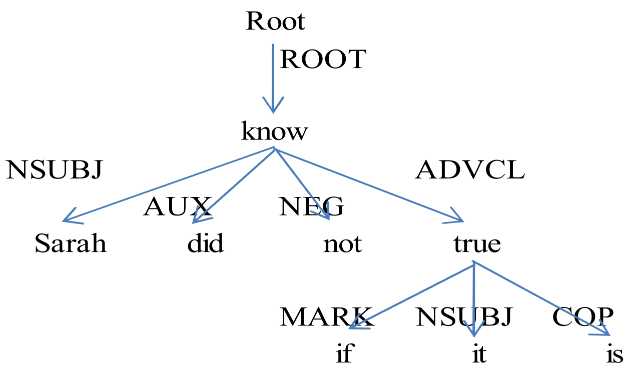


Figure 5.171. The typed-dependency tree for ‘Sarah didn’t know if it was true.’ in the Stanford Parser output.

This study does not address this type of incorrect parsed outputs, and accepts them as they are, because it is difficult to determine whether a given verb takes a clausal complement.

The dependency type “advcl” does not follow Mel’čuk’s original Criterion A because the verb in the main clause and the verb in the adverbial clause do not form a prosodic unit (e.g., ‘…read… visited’). Moreover, their linear order is not fixed. However, the main clause verb and the subordinate verb definitely form a semantic unit, thus following the revised Criterion A proposed in Section 2.4.3.1.

This dependency type follows Mel’čuk’s Criterion B because the main verb determines the passive valence of the phrase. For example, a verb taking a subordinate clause can be dependent on another verb, such as ‘John believes Sarah had read the books written by David before she visited him.’ Therefore, the main verb and the verb in the adverbial clause follow Mel’čuk’s Criterion B1.

This dependency type also follows the revised Criterion C proposed in Section 2.4.3.3, because it implies a certain kind of semantic relationship between the verbs, and this semantic relationship cannot be expressed by any other dependency type. In addition, the prototypical dependent of this dependency type is a verb.

Rcmod - relative clause modifier

The Stanford-Dependencies framework proposes three dependency types that are related to relative clauses: “rcmod,” “ref,” and “rel.” First, the dependency type “rcmod” is a subtype of “mod.” The head of this dependency type is a noun and the dependent is the head of a relative clause modifying the noun. In the figure below, the verb ‘wrote’ is the main predicate of a relative clause modifying the noun ‘book.’ In other words, the verb ‘wrote’ depends on the noun ‘book’ with the dependency type ‘rcmod.’

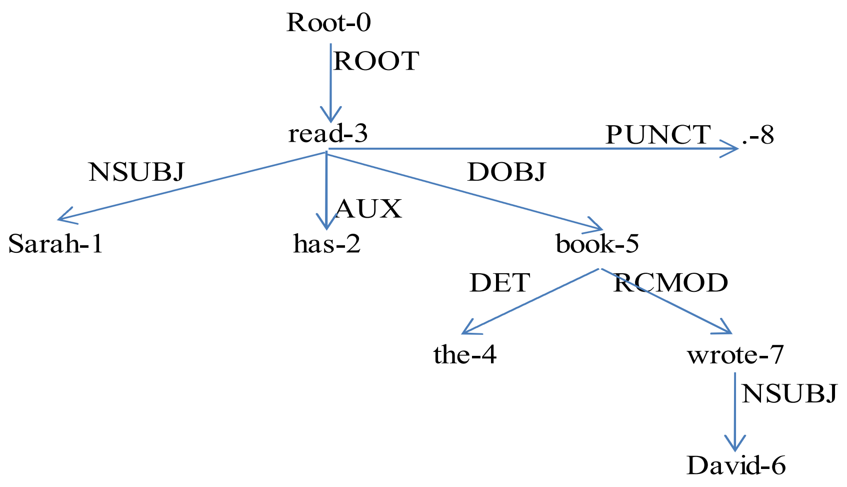


Figure 5.172. The typed-dependency tree for ‘Sarah has read the book David wrote.’

The typed-dependency tree above is equivalent to the functional structure below. Notice that the word ‘book’ is assigned the same index as the zero pronoun of the DOBJ of the verb ‘wrote,’ which indicates that this zero pronoun refers to ‘book.’

```
      |PRED    'read<NSUBJ, DOBJ>'
      |NSUBJ  |PRED      'Sarah'     |
      |       |PERSON    3RD         |
      |       |NUMBER    SINGULAR    |
      |       |GENDER    FEMININE    |
      |
      |DOBJ   |PRED      'book_i'
      |       |PERSON    3RD
      |       |NUMBER    SINGULAR
      |       |GENDER    NEUTRAL
      |       |DET      |FORM      'the'      |
      |       |         |TYPE      DEFINITE   |
      |       |
      |       |RCMOD    |PRED      'write<NSUBJ,DOBJ>'
      |       |         |NSUBJ    |PRED      'David'     |
      |       |         |         |PERSON    3RD         |
      |       |         |         |NUMBER    SINGULAR    |
      |       |         |         |GENDER    MASCULINE   |
      |       |         |
      |       |         |DOBJ     |PRED      'PRO_i'     |
      |       |         |         |TYPE      ZERO        |
      |       |         |TENSE     PAST
      |AUX    |FORM      'has'      |
      |       |TENSE     PRESENT    |
      |       |ASPECT    PERFECT    |
      |
      |PUNCT  |FORM      '.'          |
ROOT  |       |STMT-TYPE DECLARATIVE  |
```

Figure 5.173. The functional structure for ‘Sarah has read the book David wrote.’

In the figure below, the verb ‘gave’ is the main predicate of a relative clause modifying the noun ‘book.’ In other words, the verb ‘gave’ depends on the noun ‘book’ with the dependency type ‘rcmod.’

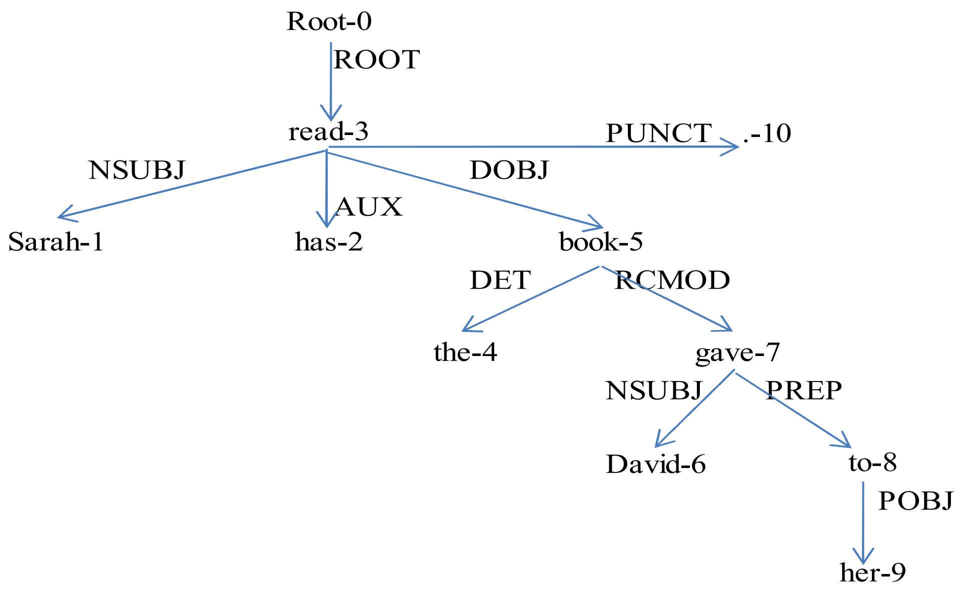


Figure 5.174. The typed-dependency tree for ‘Sarah has read the book David gave to her.’

```
      PRED    'read<NSUBJ, DOBJ>'
      NSUBJ   PRED       'Sarah_j'
              PERSON     3RD
              NUMBER     SINGULAR
              GENDER     FEMININE

      DOBJ    PRED       'book_i'
              PERSON     3RD
              NUMBER     SINGULAR
              GENDER     NEUTRAL
              DET        FORM       'the'
                         TYPE       DEFINITE

              RCMOD      PRED       'give<NSUBJ,DOBJ,PREP>'
                         NSUBJ      PRED      'David'
                                    PERSON    3RD
                                    NUMBER    SINGULAR
                                    GENDER    MASCULINE

                         DOBJ       PRED      'PRO_i'
                                    TYPE      ZERO

                         PREP       PRED      'to<POBJ>'
                                    POBJ      PRED       'PRO_j'
                                              PERSON     3RD
                                              NUMBER     SINGULAR
                                              GENDER     FEMININE
                                              FORM       'her'
                         TENSE      PAST

      AUX     FORM       'has'
              TENSE      PRESENT
              ASPECT     PERFECT

      PUNCT   FORM       '.'
ROOT          STMT-TYPE  DECLARATIVE
```

Figure 5.175. The functional structure for ‘Sarah has read the book David gave to her.’

When a relative pronoun introduces a relative clause, it functions as an argument or an adjunct of the main predicate of the relative clause. In the figure below, the relative pronoun ‘which’ introduces the relative clause, and it functions as the direct object of the ‘wrote.’ In other words, the relative pronoun ‘which’ depends on the verb ‘wrote’ with the dependency type “dobj.”

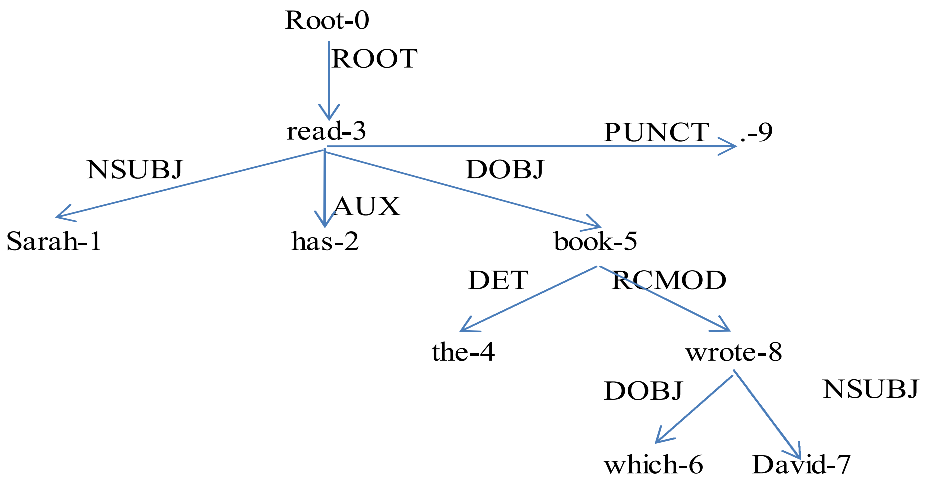


Figure 5.176. The typed-dependency tree for ‘Sarah has read the book which David wrote.’

The typed-dependency tree above is equivalent to the functional structure below. Notice that the indices ‘i’ on ‘which’ and ‘book’ indicate that the relative pronoun ‘which’ refers to ‘book.’

```
     | PRED    'read<NSUBJ, DOBJ>'
     | NSUBJ  | PRED      'Sarah'     |
     |        | PERSON    3RD         |
     |        | NUMBER    SINGULAR    |
     |        | GENDER    FEMININE    |
     |
     | DOBJ   | PRED      'book_i'
     |        | PERSON    3RD
     |        | NUMBER    SINGULAR
     |        | GENDER    NEUTRAL
     |        | DET      | FORM     'the'      |
     |        |          | TYPE     DEFINITE   |
     |        |
     |        | RCMOD   {| PRED     'write<NSUBJ,DOBJ>'
     |        |          | NSUBJ   | PRED     'David'     |
     |        |          |         | PERSON   3RD         |
     |        |          |         | NUMBER   SINGULAR    |
     |        |          |         | GENDER   MASCULINE   |
     |        |          |
     |        |          | DOBJ    | PRED     'PRO_i'     |
     |        |          |         | TYPE     RELATIVE    |
     |        |          |         | FORM     'which'     |
     |        |          | TENSE    PAST                  }
     | AUX    | FORM      'has'       |
     |        | TENSE     PRESENT     |
     |        | ASPECT    PERFECT     |
     |
     | PUNCT  | FORM      '.'         |
ROOT |        | STMT-TYPE DECLARATIVE |
```

Figure 5.177. The functional structure for ‘Sarah has read the book which David wrote.’

In the figure below, the relative pronoun ‘which’ introduces the relative clause, and it functions as the direct object of the ‘gave.’ In other words, the relative pronoun ‘which’ depends on the verb ‘gave’ with the dependency type “dobj.”

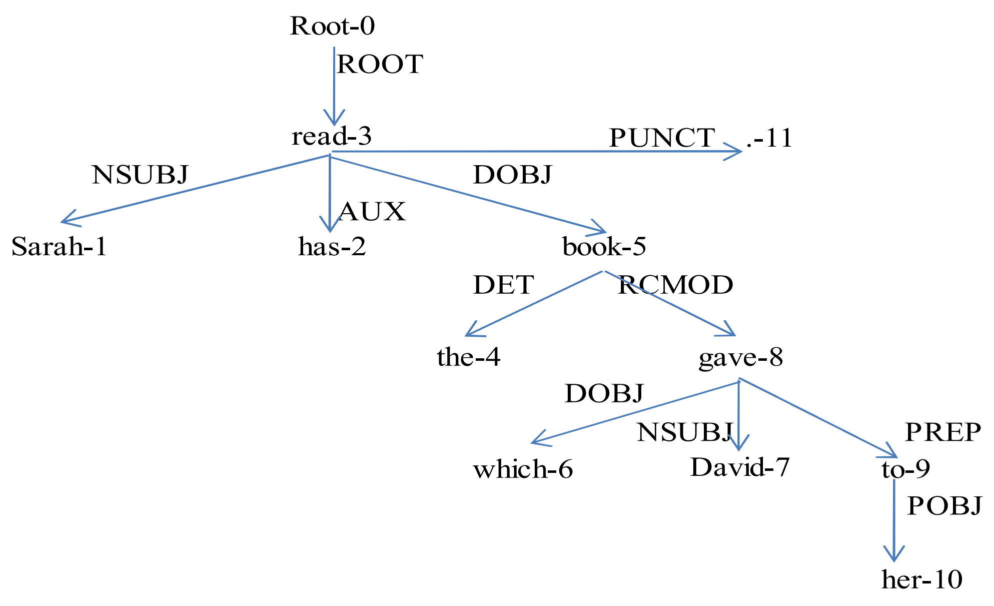


Figure 5.178. The typed-dependency tree for ‘Sarah has read the book which David gave to her.’

The typed-dependency tree above is equivalent to the functional structure below. Notice that the indices ‘i’ on ‘which’ and ‘book’ indicate that the relative pronoun ‘which’ refers to ‘book,’ and the indices ‘j’ on ‘Sarah’ and ‘her’ indicate that this ‘her’ refers to ‘Sarah.’

$$\begin{array}{ll}
\text{ROOT} & \left[\begin{array}{ll}
\text{PRED} & \text{'read<NSUBJ, DOBJ>'} \\
\text{NSUBJ} & \left[\begin{array}{ll}
\text{PRED} & \text{'Sarah}_j\text{'} \\
\text{PERSON} & \text{3RD} \\
\text{NUMBER} & \text{SINGULAR} \\
\text{GENDER} & \text{FEMININE}
\end{array}\right] \\
\text{DOBJ} & \left[\begin{array}{ll}
\text{PRED} & \text{'book}_i\text{'} \\
\text{PERSON} & \text{3RD} \\
\text{NUMBER} & \text{SINGULAR} \\
\text{GENDER} & \text{NEUTRAL} \\
\text{DET} & \left[\begin{array}{ll}
\text{FORM} & \text{'the'} \\
\text{TYPE} & \text{DEFINITE}
\end{array}\right] \\
\text{RCMOD} & \left[\begin{array}{ll}
\text{PRED} & \text{'give<NSUBJ,DOBJ,PREP>'} \\
\text{NSUBJ} & \left[\begin{array}{ll}
\text{PRED} & \text{'David'} \\
\text{PERSON} & \text{3RD} \\
\text{NUMBER} & \text{SINGULAR} \\
\text{GENDER} & \text{MASCULINE}
\end{array}\right] \\
\text{DOBJ} & \left[\begin{array}{ll}
\text{PRED} & \text{'PRO}_i\text{'} \\
\text{TYPE} & \text{RELATIVE} \\
\text{FORM} & \text{'which'}
\end{array}\right] \\
\text{PREP} & \left[\begin{array}{ll}
\text{PRED} & \text{'to<POBJ>'} \\
\text{POBJ} & \left[\begin{array}{ll}
\text{PRED} & \text{'PRO}_j\text{'} \\
\text{PERSON} & \text{3RD} \\
\text{NUMBER} & \text{SINGULAR} \\
\text{GENDER} & \text{FEMININE} \\
\text{FORM} & \text{'her'}
\end{array}\right]
\end{array}\right] \\
\text{TENSE} & \text{PAST}
\end{array}\right]
\end{array}\right] \\
\text{AUX} & \left[\begin{array}{ll}
\text{FORM} & \text{'has'} \\
\text{TENSE} & \text{PRESENT} \\
\text{ASPECT} & \text{PERFECT}
\end{array}\right] \\
\text{PUNCT} & \left[\begin{array}{ll}
\text{FORM} & \text{'.'} \\
\text{STMT-TYPE} & \text{DECLARATIVE}
\end{array}\right]
\end{array}\right]
\end{array}$$

Figure 5.179. The functional structure for ‘Sarah has read the book which David gave to her.’

In the figure below, the relative pronoun ‘where’ introduces the relative clause, and it functions as an adverbial modifier for the verb ‘works.’ In other words, the relative pronoun ‘where’ depends on the verb ‘works’ with the dependency type “advmod.”

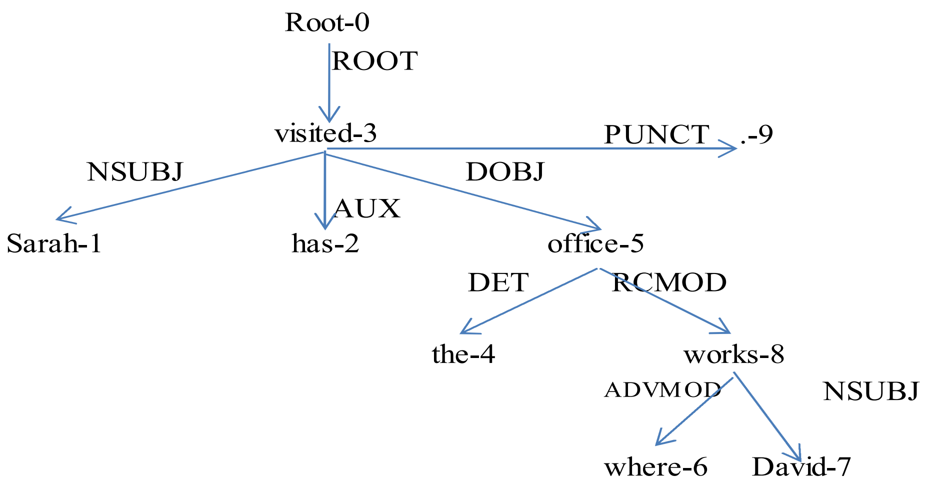


Figure 5.180. The typed-dependency tree for ‘Sarah has visited the office where David works.’

The typed-dependency tree above is equivalent to the functional structure below. The relative pronoun ‘where’ refers to ‘office.’

```
      PRED    'visit<NSUBJ, DOBJ>'
      NSUBJ   PRED        'Sarah'
              PERSON      3RD
              NUMBER      SINGULAR
              GENDER      FEMININE

      DOBJ    PRED        'office_i'
              PERSON      3RD
              NUMBER      SINGULAR
              GENDER      NEUTRAL
              DET         FORM        'the'
                          TYPE        DEFINITE

              RCMOD       PRED        'work<NSUBJ>'
                          NSUBJ       PRED      'David'
                                      PERSON    3RD
                                      NUMBER    SINGULAR
                                      GENDER    MASCULINE

                          ADVMOD      PRED      'PRO_i'
                                      TYPE      RELATIVE
                                      FORM      'where'
                          TENSE       PRESENT
      AUX     FORM        'has'
              TENSE       PRESENT
              ASPECT      PERFECT

      PUNCT   FORM        '.'
ROOT          STMT-TYPE   DECLARATIVE
```

Figure 5.181. The functional structure for ‘Sarah has visited the office where David works.’

This dependency type does not follow Mel’čuk’s Criterion A, because nouns and their relative-clause modifiers do not constitute prosodic units (e.g., ‘book wrote’ in the example sentence above). However, this dependency type forms a semantic unit whereby the noun is an argument or adjunct of the verb in the relative clause. In this way, this type follows the revised Criterion A proposed in Section 2.4.3.1.

This dependency type also follows Mel’čuk’s Criterion B. The noun, not its relative-clause modifier, determines the passive valence of the phrase. For example, the dependency relation between ‘book’ and ‘wrote’ in the sentence ‘Sarah has read the book David wrote’ shows that ‘wrote’ depends on the noun ‘book,’ which can be subordinated to the verb ‘read.’

In addition, this dependency type follows the revised Criterion C proposed in Section 2.4.3.3. This type implies a certain kind of semantic relationship between a noun and a relative-clause modifier. Moreover, this semantic relationship cannot be expressed by any other dependency type, and the prototypical dependent of this dependency type is a verb.

Ref – referent (word which a relative pronoun refers to)

The second dependency type related to relative clauses is “ref,” which is a subtype of “mod.” The head of this type of dependency is a noun and the dependent is a relative pronoun that refers to the noun. An example of the “ref” dependency type is presented in the figure below. The relative pronoun ‘which’ refers to the noun ‘book,’ and it also functions as the direct object of the verb ‘wrote.’ In other words, the relative pronoun ‘which’ depends on the noun ‘book’ with the dependency type “ref,” and it also depends on ‘wrote’ with the dependency type “dobj.”

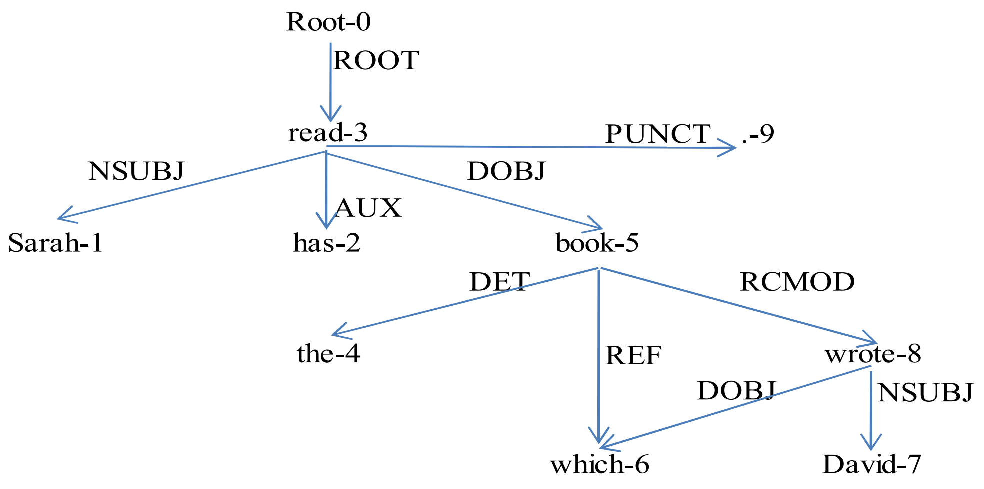


Figure 5.182. The dependency tree containing the dependency type “ref” for ‘Sarah has read the book which David wrote.’

This type is different from other dependency types in Stanford Dependencies in that it represents a semantic dependency whereby the semantic content of a relative pronoun is identified. This dependency type is not implemented in the default output style of the Stanford Parser (cf. Section 5.4.5).

Rel - relative (word introducing a relative clause modifier)

The third dependency type which is related to relative clauses is "rel," which is another subtype of "mod." The head of this type of dependency is the head of a relative clause, and the dependent is the relative pronoun that introduces the relative clause. De Marneffe and Manning (2012, p.10) state that this type characterizes the relative word that does not function as the subject of the relative clause. Consider again the sentence 'I saw the man whose wife you love.' According to this definition, the dependency between 'love' and 'wife' should be typed as "rel," as shown in the typed-dependency tree representation below.

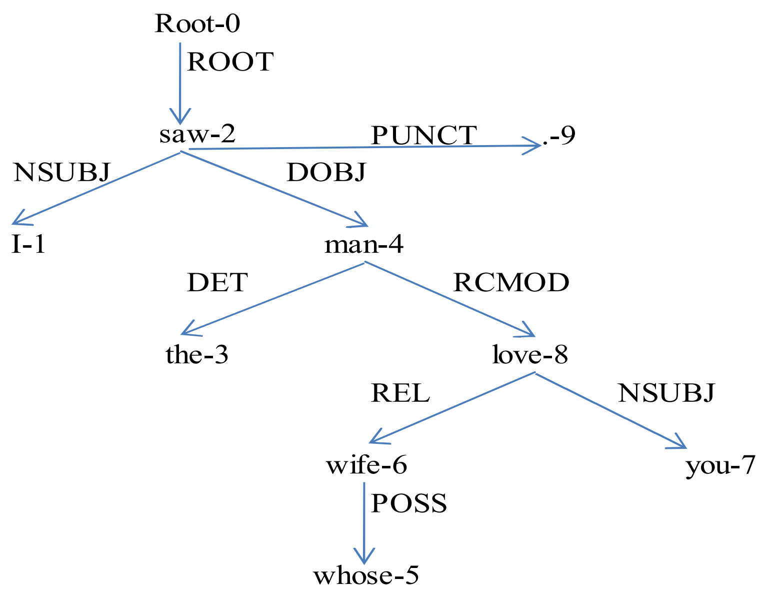


Figure 5.183. The dependency tree for ‘I saw the man whose wife you love.’

However, this dependency type is redundant and can be replaced by other dependency types (e.g., “nsubj,” “dobj,” “iobj”) that describe the relation between a noun and the head of a relative clause. In this study, the type “rel,” if it occurs in the parsed output, will be treated as a parse error, and will be replaced by an appropriate dependency type.

Purpcl - purpose clause modifier

The dependency type “purpcl” is another subtype of “mod” used to describe cases where the dependent is the head of a purpose clause introduced by ‘in order to.’ This dependency is not actually implemented in the Stanford Parser, and the dependency of a purpose clause is parsed as “xcomp,” as shown in the figure below.

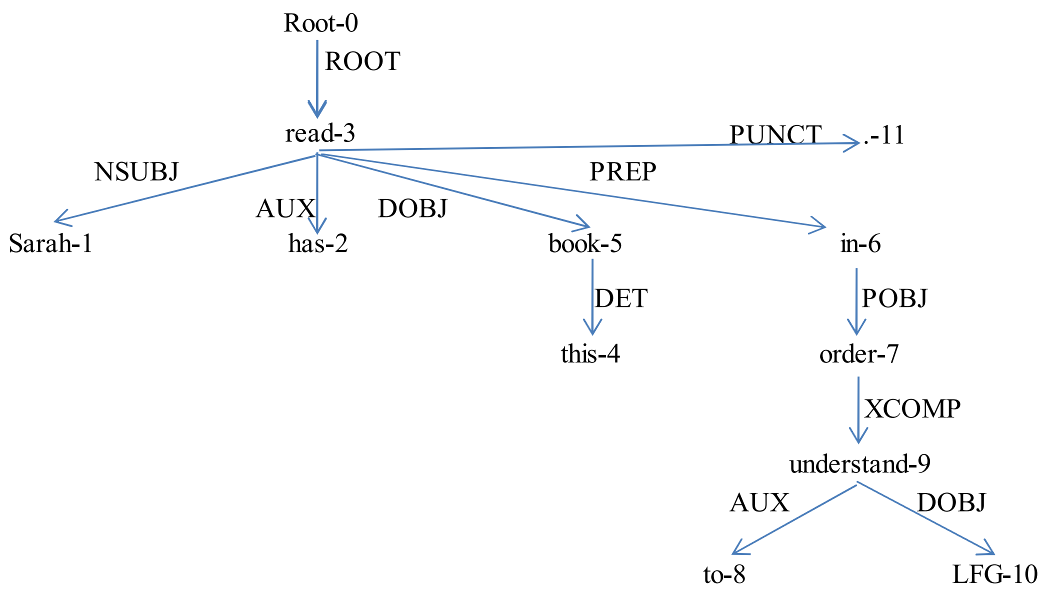


Figure 5.184. The typed-dependency tree for ‘Sarah has read this book in order to understand

LFG.’

Tmod - temporal modifier

The dependency type “tmod” is another subtype of “mod” for cases where the dependent is the head of a temporal modifier. In the figure below, the temporal adverbial phrase ‘last night’ modifies the verb ‘read.’ In other words, the main predicate of the temporal adverbial ‘night’ depends on the verb ‘read’ with the dependency type “tmod.”

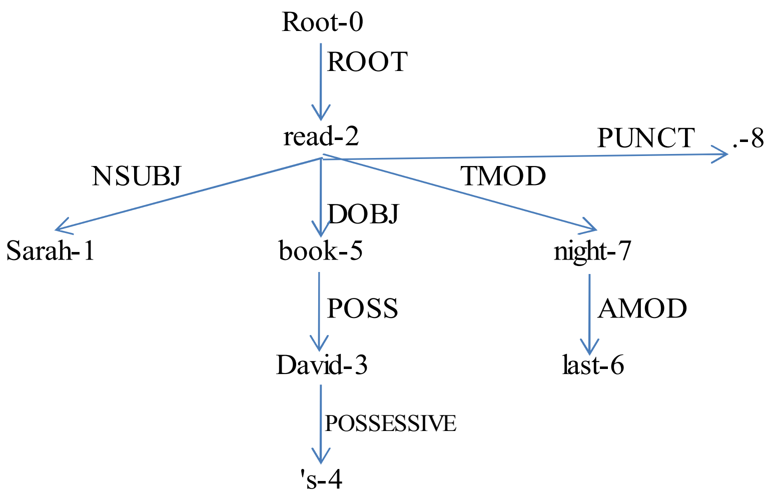


Figure 5.185. The typed-dependency tree for ‘Sarah read David’s book last night.’

```
     |       |PRED     'read<NSUBJ,DOBJ>'                          | |
     |       |NSUBJ   |PRED          'Sarah'      |                | |
     |       |        |PERSON        3RD          |                | |
     |       |        |NUMBER        SINGULAR     |                | |
     |       |        |GENDER        FEMININE     |                | |
     |       |                                                     | |
     |       |DOBJ    |PRED          'book'                      | | |
     |       |        |PERSON        3RD                         | | |
     |       |        |NUMBER        SINGULAR                    | | |
     |       |        |GENDER        NEUTRAL                     | | |
     |       |        |POSS         |PRED        'David'        || | |
     |       |        |             |PERSON      3RD            || | |
     |       |        |             |NUMBER      SINGULAR       || | |
     |       |        |             |GENDER      MASCULINE      || | |
     |       |        |             |POSSESSIVE |FORM   's'  | || | |
     |       |                                                     | |
     |       |TMOD  { |PRED          'night'                    |} | |
     |       |        |AMOD         |PRED        'last'   |      |} | |
     |       |TENSE    PAST                                        | |
     |       |PUNCT   |FORM          '.'          |                | |
     |ROOT   |        |STMT-TYPE     DECLARATIVE  |                | |
```

Figure 5.186. The functional structure for ‘Sarah read David’s book last night.’

The fact that the head of this dependency can be either a noun or a verb results in ambiguous syntactic analyses for the same sentence, similar to prepositional modifiers. For example, the temporal adverbial phrase ‘last night’ can modify either the verb or the noun. In the figure below, the temporal adverbial phrase ‘last night’ depends on the word ‘cancelled.’ Therefore, in this analysis, this sentence means that Sarah’s cancellation of the meeting took place last night.

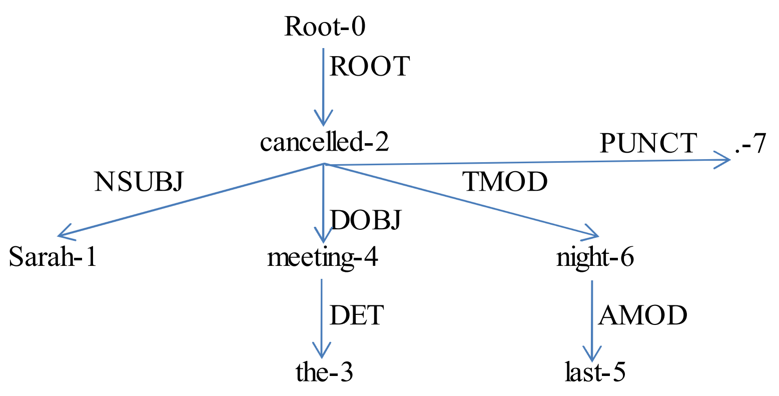


Figure 5.187. The typed-dependency tree for ‘Sarah cancelled the meeting last night.’ in the Stanford Parser output

```
|ROOT  | PRED    'cancel<NSUBJ,DOBJ>'
       | NSUBJ  | PRED       'Sarah'     |
       |        | PERSON     3RD         |
       |        | NUMBER     SINGULAR    |
       |        | GENDER     FEMININE    |
       |
       | DOBJ   | PRED       'meeting'                    |
       |        | PERSON     3RD                          |
       |        | NUMBER     SINGULAR                     |
       |        | GENDER     NEUTRAL                      |
       |        | DET       | FORM     'the'        |    |
       |        |           | TYPE     DEFINITE     |    |
       |
       | TMOD  {| PRED       'night'              | }
       |       {| AMOD      | PRED     'last'  |  | }
       | TENSE    PAST
       | PUNCT  | FORM       '.'               |
       |        | STMT-TYPE  DECLARATIVE       |
```

Figure 5.188. The functional structure equivalent to the typed-dependency tree for ‘Sarah cancelled the meeting last night.’ in the Stanford Parser output

On the other hand, in the figure below, the temporal adverbial phrase ‘last night’ depends on the word ‘meeting.’ In this analysis, this sentence means that Sarah’s cancellation of the meeting took place *before* last night.

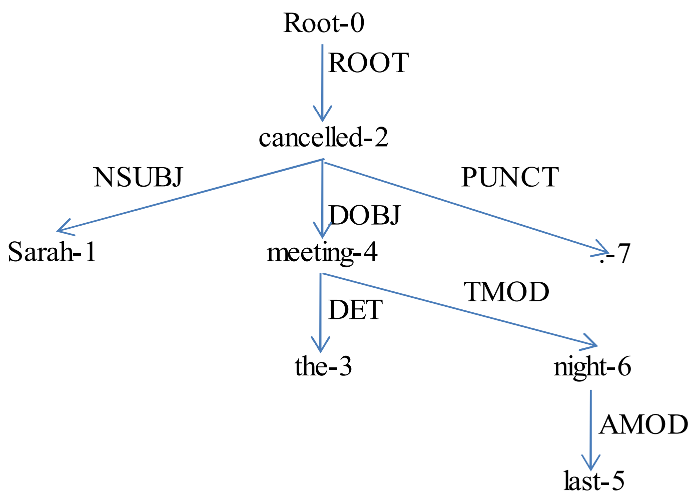


Figure 5.189. Another typed-dependency tree for ‘Sarah cancelled the meeting last night.’

```
|          |PRED      'cancel<NSUBJ,DOBJ>'                               ||
|          |NSUBJ     |PRED      'Sarah'     |                           ||
|          |          |PERSON    3RD         |                           ||
|          |          |NUMBER    SINGULAR    |                           ||
|          |          |GENDER    FEMININE    |                           ||
|          |                                                             ||
|          |DOBJ      |PRED      'meeting'                          |    ||
|          |          |PERSON    3RD                                |    ||
|          |          |NUMBER    SINGULAR                           |    ||
|          |          |GENDER    NEUTRAL                            |    ||
|          |          |DET       |FORM      'the'          |       |    ||
|          |          |          |TYPE      DEFINITE       |       |    ||
|          |          |                                            |    ||
|          |          |TMOD     [|PRED      'night'           |]   |    ||
|          |          |         [|AMOD      |PRED  'last'  |  |]   |    ||
|          |                                                             ||
|          |TENSE     PAST                                               ||
|          |PUNCT     |FORM      '.'         |                           ||
|ROOT      |          |STMT-TYPE DECLARATIVE |                           ||
```

Figure 5.190. The functional structure for ‘Sarah cancelled the meeting last night.’

This dependency type does not follow Mel’čuk’s original Criterion A for SSyntRel because a verb and its temporal modifier do not constitute a prosodic unit (e.g., ‘read… night’ in the example sentence above). In addition, the linear order of the words in this dependency type cannot be determined because it is possible to say ‘Sarah read David’s book last night’ or ‘Last night, Sarah read David’s book.’ However, this type forms a semantic unit in which the temporal adverb modifies the meaning of the verb. Thus, this type follows the revised Criterion A discussed in Section 2.4.3.1.

This dependency type also follows Mel’čuk’s Criterion B because the verb, not its temporal modifier, determines the passive valence of the phrase. For example, the dependency relation between ‘read’ and ‘night’ in the sentence ‘Sarah read David’s book last night’ shows that ‘night’ depends on the verb ‘read,’ which can be subordinated to the root of the sentence.

In addition, this dependency type follows the revised Criterion C because it implies a certain kind of semantic relationship between a verb and a temporal modifier. This semantic relationship cannot be expressed by any other dependency type, and the prototypical dependent of this dependency type is a temporal adverbial or a temporal noun. Therefore, this dependency type follows the revised Criteria C1 and C2 proposed in Section 2.4.3.3.

Prt - phrasal verb particle

The dependency type “prt” is an additional subtype of “mod” and represents cases where the head is a verb and the dependent is a particle, which creates a phrasal verb. In the figure below, the verb ‘worked’ and the particle ‘out’ create a phrasal verb ‘worked out.’ In other words, the particle ‘out’ depends on the verb ‘worked’ with the dependency type “prt.”

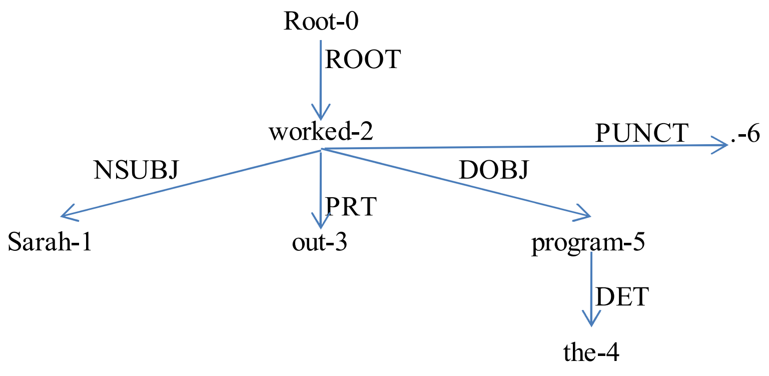


Figure 5.191. The typed-dependency tree for ‘Sarah worked out the program.’

```
     | PRED    'work<NSUBJ,DOBJ>'                                  ||
     | NSUBJ  | PRED       'Sarah'     |                           ||
     |        | PERSON     3RD         |                           ||
     |        | NUMBER     SINGULAR    |                           ||
     |        | GENDER     FEMININE    |                           ||
     |                                                             ||
     | DOBJ   | PRED       'program'                         |     ||
     |        | PERSON     3RD                               |     ||
     |        | NUMBER     SINGULAR                          |     ||
     |        | GENDER     NEUTRAL                           |     ||
     |        | DET       | FORM     'the'           |       |     ||
     |        |           | TYPE     DEFINITE        |       |     ||
     |                                                             ||
     | PRT    | FORM       'out'       |                           ||
     | TENSE    PAST                                               ||
     | PUNCT  | FORM       '.'         |                           ||
|ROOT|        | STMT-TYPE  DECLARATIVE |                           ||
```

Figure 5.192. The functional structure for ‘Sarah worked out the program.’

Transitive phrasal verbs can have another word order in which the particle follows the direct object. This word order is obligatory when the direct object is a pronoun. In the figure below, the number next to each word indicates that the direct object ‘the program’ precedes the particle.

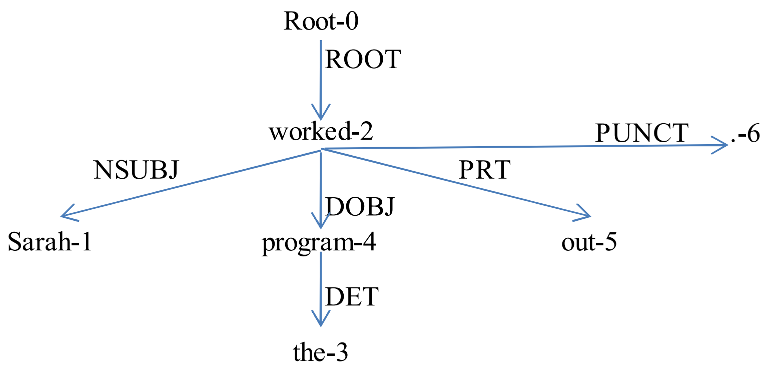


Figure 5.193. The typed-dependency tree for ‘Sarah worked the program out.’

In the figure below, the direct object is a pronoun, so it must precede the particle.

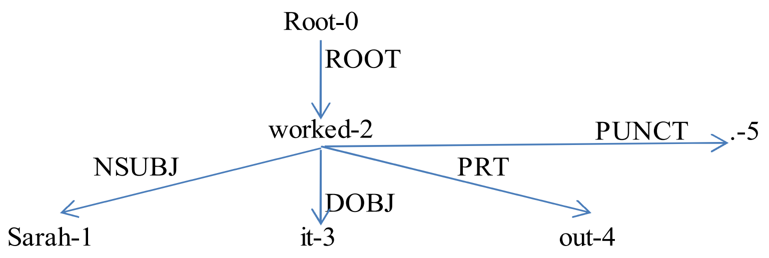


Figure 5.194. The typed-dependency tree for ‘Sarah worked it out.’

There are instances in which a verb and a preposition form a phrasal verb. For example, ‘read through’ can take its direct object either before or after the word ‘through,’ as shown in the following figures.

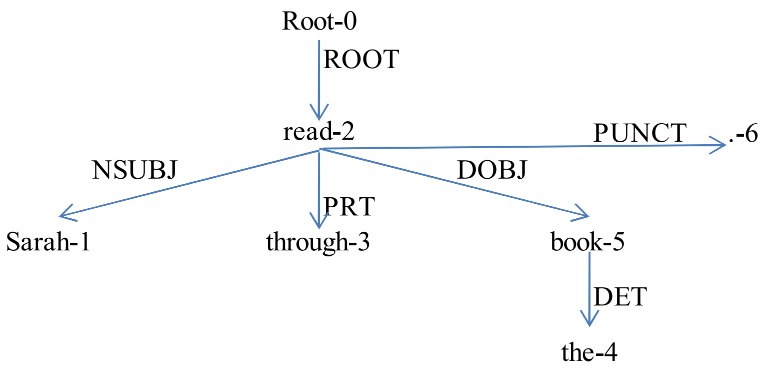


Figure 5.195. The typed-dependency tree for ‘Sarah read through the book.’

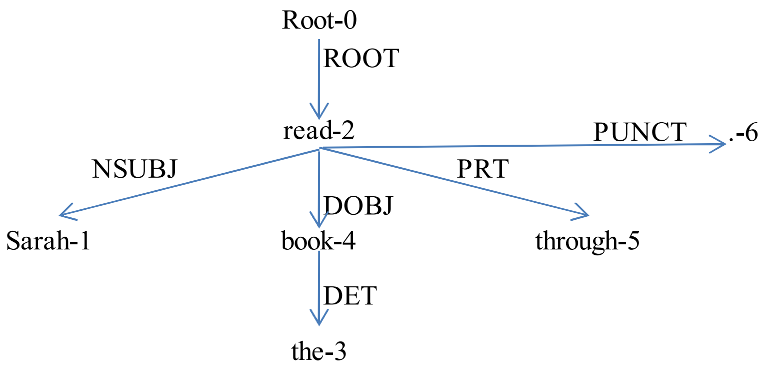


Figure 5.196. The typed-dependency tree for ‘Sarah read the book through.’

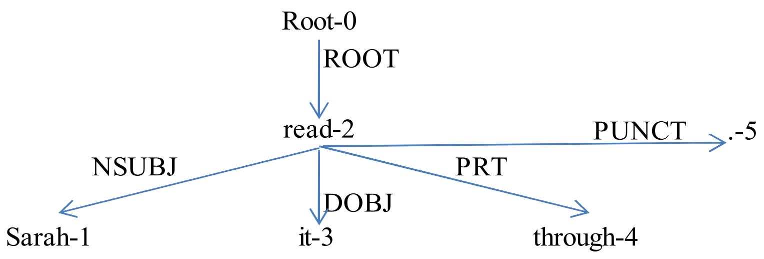


Figure 5.197. The typed-dependency tree for ‘Sarah read it through.’

The Stanford Parser parses the word ‘through’ as a preposition. In the example sentence ‘Sarah read through the book,’ the word ‘book’ is parsed as the prepositional object, as shown below.

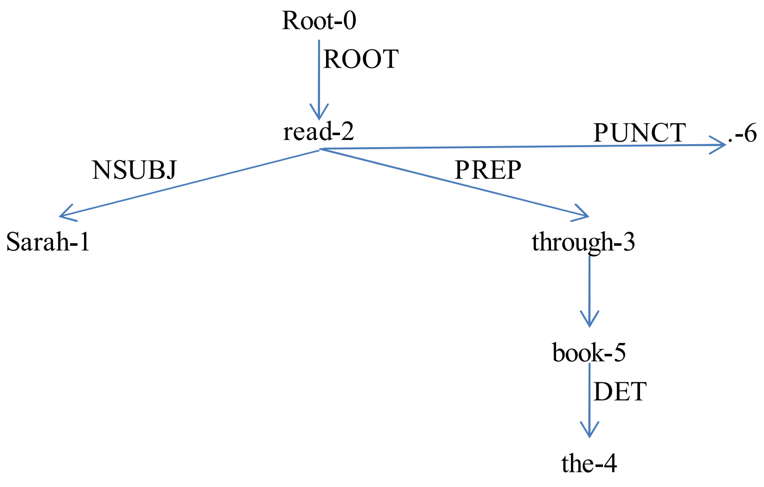


Figure 5.198. The typed-dependency tree for ‘Sarah read through the book.’ in the Stanford Parser output

When the word ‘through’ is put after the direct object, it is parsed to be a preposition which lacks its object, as shown in the figure below.

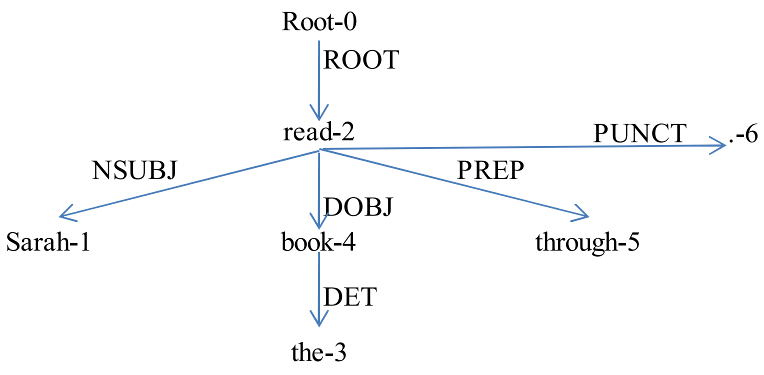


Figure 5.199. An incorrect typed-dependency tree for ‘Sarah read the book through.’ in the Stanford Parser output

The Stanford Parser yields the same incorrect parse when the word 'through' is placed after the pronominal direct object, as shown in the figure below.

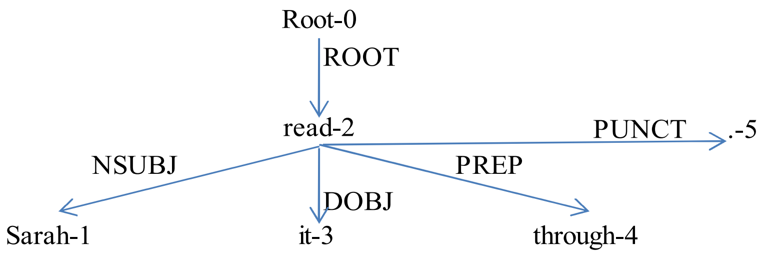


Figure 5.200. An incorrect typed-dependency tree for 'Sarah read it through.' in Stanford Parser output

This dependency type does not follow Mel'čuk's Criterion A completely. The linear order of the head and the dependent is fixed (the verb always precedes the particle), yet phrasal verbs do not always constitute a prosodic unit. For example, when the particle immediately follows the verb (e.g., 'worked out' in the example sentence above), they constitute a prosodic unit; however, when the particle follows the direct object, they do not constitute a prosodic unit (worked … out). On the other hand, this type forms a semantic unit because the particle modifies the meaning of the verb. In this way, this type follows the revised Criterion A proposed in Section 2.4.3.1.

This dependency type follows Mel'čuk's Criterion B as well. The verb, not the particle, determines the passive valence of the phrasal verb. For example, the dependency relation between 'worked' and 'out' in the sentence 'Sarah worked out the program' is such that 'out' depends on the verb 'worked,' which can be subordinated to the root of the sentence.

This dependency type also follows the revised Criterion C because it implies a certain kind of semantic relationship between a verb and a particle. This semantic relationship cannot be expressed by any other dependency type, and the prototypical dependent of this dependency type is a particle. Therefore, this dependency type follows the revised Criteria C1 and C2 proposed in Section 2.4.3.3.

Appos - appositional modifier

The dependency type "appos" is another subtype of "mod" whereby the dependent is an appositional modifier. An appositional modifier is a noun phrase immediately to the right of another noun phrase "that serves to define or modify" the noun phrase (de Marneffe & Manning 2012, p.3). In the figure below, the noun phrase 'David's niece' defines the word 'Sarah.' In other words, the word 'niece,' which is the main predicate of the noun phrase 'David's niece,' depends on the word 'Sarah' with the dependency type "appos."

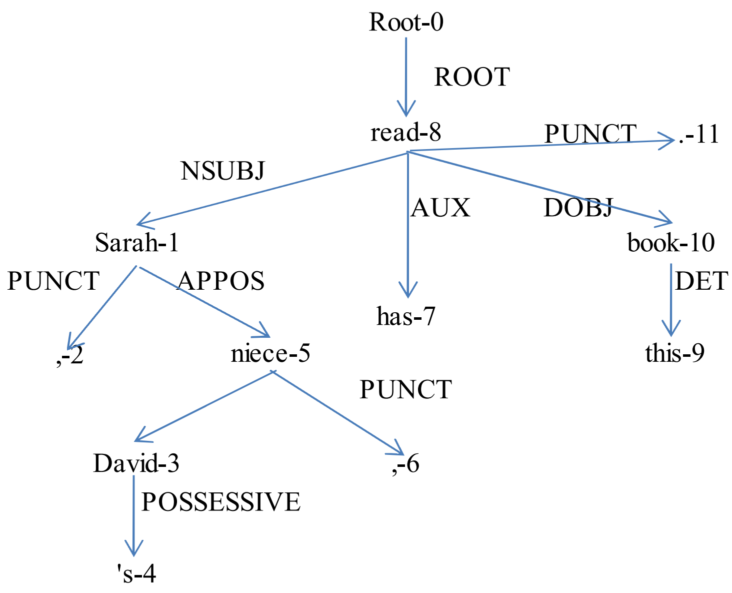


Figure 5.201. The typed-dependency tree for 'Sarah, David's niece, has read this book.'

```
      PRED    'read<NSUBJ,DOBJ>'
      NSUBJ   PRED        'Sarah'
              PERSON      3RD
              NUMBER      SINGULAR
              GENDER      FEMININE
              PUNCT       FORM         ','
              APPOS       PRED         'niece'
                          PERSON       3RD
                          NUMBER       SINGULAR
                          GENDER       FEMININE
                          POSS         PRED          'David'
                                       PERSON        3RD
                                       NUMBER        SINGULAR
                                       GENDER        MASCULINE
                                       POSSESSIVE    FORM        ''s'

                          PUNCT        FORM          ','

      DOBJ    PRED        'book'
              PERSON      3RD
              NUMBER      SINGULAR
              GENDER      NEUTRAL
              DET         FORM         'this'
                          TYPE         DEFINITE

      AUX     FORM        'has'
              TENSE       PRESENT
              ASPECT      PERFECT

      PUNCT   FORM        '.'
ROOT          STMT-TYPE   DECLARATIVE
```

Figure 5.202. The functional structure for ‘Sarah, David’s niece, has read this book.’

In the figure below, the noun phrase ‘her friend’ defines the word ‘Isaac.’ In other words, the word ‘friend,’ which is the main predicate of the noun phrase ‘her friend,’ depends on the word ‘Isaac’ with the dependency type “appos.”

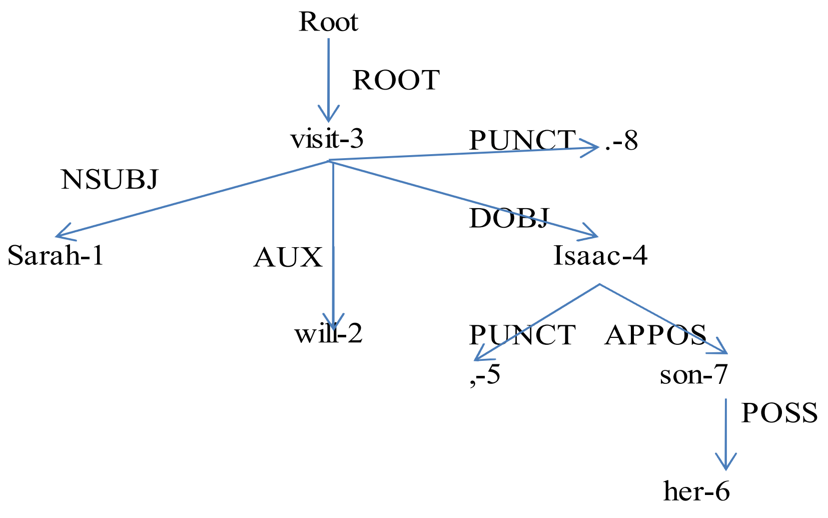


Figure 5.203. The typed-dependency tree for ‘Sarah will visit Isaac, her friend.’

```
     | PRED    'visit<NSUBJ,DOBJ>'                                                   |
     | NSUBJ  | PRED      'Sarahᵢ'     |                                             |
     |        | PERSON    3RD          |                                             |
     |        | NUMBER    SINGULAR     |                                             |
     |        | GENDER    FEMININE     |                                             |
     |                                                                               |
     | DOBJ   | PRED      'Issac'                                                  | |
     |        | PERSON    3RD                                                      | |
     |        | NUMBER    SINGULAR                                                 | |
     |        | GENDER    MASCULINE                                                | |
     |        | APPOS   [| PRED       'friend'                                  | ]| |
     |        |          | PERSON     3RD                                       |  | |
     |        |          | NUMBER     SINGULAR                                  |  | |
     |        |          | GENDER     MASCULINE                                 |  | |
     |        |          | POSS      | PRED      'PROᵢ'                      |  |  | |
     |        |          |           | PERSON    3RD                         |  |  | |
     |        |          |           | NUMBER    SINGULAR                    |  |  | |
     |        |          |           | GENDER    FEMININE                    |  |  | |
     |        |          |           | FORM     |'her'                   |  |  |  | |
     |        |          |                                                   |  | |
     |        |          | PUNCT     | FORM      ','                    |     |  ]| |
     |                                                                               |
     | AUX    | FORM      'will'       |                                             |
     |        | TENSE     FUTURE       |                                             |
     |                                                                               |
     | PUNCT  | FORM      '.'          |                                             |
ROOT |        | STMT-TYPE  DECLARATIVE |                                             |
```

Figure 5.204. The functional structure for ‘Sarah will visit Isaac, her friend.’

This dependency type does not follow Mel'čuk's Criterion A because nouns in apposition do not constitute prosodic units (e.g., 'Sarah… niece' in the example sentence above). However, these nouns do form a semantic unit, because one of the nouns is a paraphrase of the other noun. Thus, this type follows the revised Criterion A discussed in Section 2.4.3.1.

This dependency type follows Mel'čuk's Criterion B. The first noun, not the second one, determines the passive valence of an appositional phrase. For example, the dependency relation between 'Sarah' and 'niece' in the sentence 'Sarah, David's niece, has read the book' shows that 'niece' depends on 'Sarah,' which can be subordinated to the root of the sentence.

This dependency type also follows the revised Criterion C given that it implies a certain kind of semantic relationship between nouns. This semantic relationship cannot be expressed by any other dependency type, and the prototypical dependent of this dependency type is a noun. Therefore, this dependency type follows the revised Criteria C1 and C2 proposed in Section 2.4.3.3.

Predet – predeterminer

The dependency type "predet" is another subtype of "mod" whereby the head is a noun and the dependent is a predeterminer, as shown in the figures below.

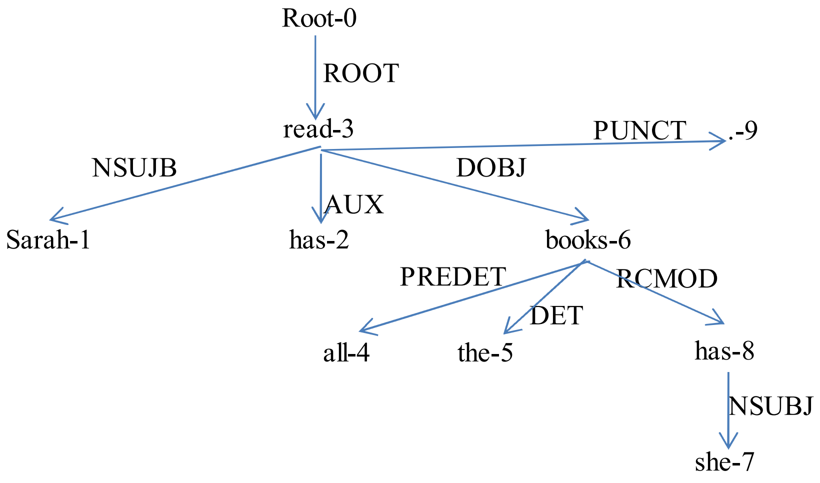


Figure 5.205. The typed-dependency tree for ‘Sarah has read all the books she has.’

```
      PRED    'read<NSUBJ, DOBJ>'
      NSUBJ   PRED     'Sarah_j'
              PERSON   3RD
              NUMBER   SINGULAR
              GENDER   FEMININE

      DOBJ    PRED     'books_i'
              PERSON   3RD
              NUMBER   PLURAL
              GENDER   NEUTRAL
              DET      FORM      'the'
                       TYPE      DEFINITE

              PREDET   FORM      'all'

              RCMOD    PRED      'have<NSUBJ,DOBJ>'
                       NSUBJ     PRED     'PRO_j'
                                 PERSON   3RD
                                 NUMBER   SINGULAR
                                 GENDER   FEMININE
                                 FORM     'she'

                       DOBJ      PRED     'PRO_i'
                                 TYPE     ZERO
                       TENSE     PRESENT
      AUX     FORM     'has'
              TENSE    PRESENT
              ASPECT   PERFECT

      PUNCT   FORM     '.'
ROOT          STMT-TYPE DECLARATIVE
```

Figure 5.206. The functional structure for ‘Sarah has read all the books she has.’

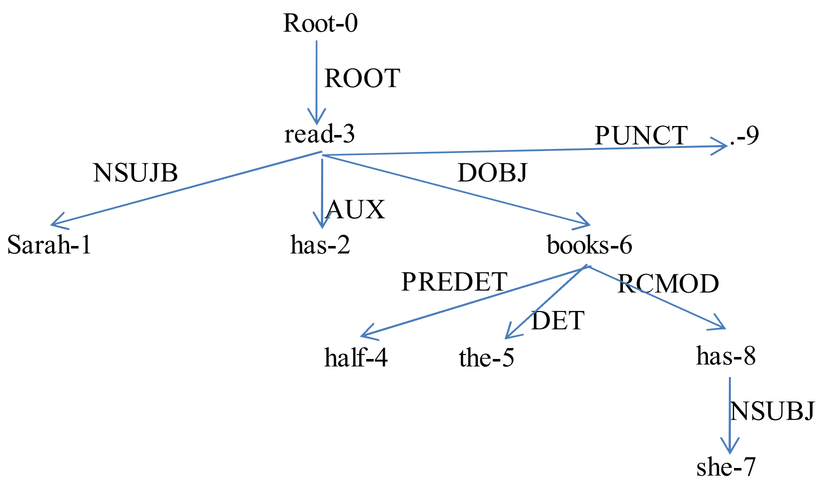


Figure 5.207. The typed-dependency tree for ‘Sarah has read half the books she has.’

```
      PRED    'read<NSUBJ, DOBJ>'
      NSUBJ   PRED      'Sarah$_j$'
              PERSON    3RD
              NUMBER    SINGULAR
              GENDER    FEMININE

      DOBJ    PRED      'books$_i$'
              PERSON    3RD
              NUMBER    PLURAL
              GENDER    NEUTRAL
              DET       FORM      'the'
                        TYPE      DEFINITE

              PREDET    FORM      'half'

              RCMOD     PRED      'have<NSUBJ,DOBJ>'
                        NSUBJ     PRED      'PRO$_j$'
                                  PERSON    3RD
                                  NUMBER    SINGULAR
                                  GENDER    FEMININE
                                  FORM      'she'

                        DOBJ      PRED      'PRO$_i$'
                                  TYPE      ZERO
                        TENSE     PRESENT
      AUX     FORM      'has'
              TENSE     PRESENT
              ASPECT    PERFECT

      PUNCT   FORM      '.'
ROOT          STMT-TYPE DECLARATIVE
```

Figure 5.208. The functional structure for ‘Sarah has read half the books she has.’

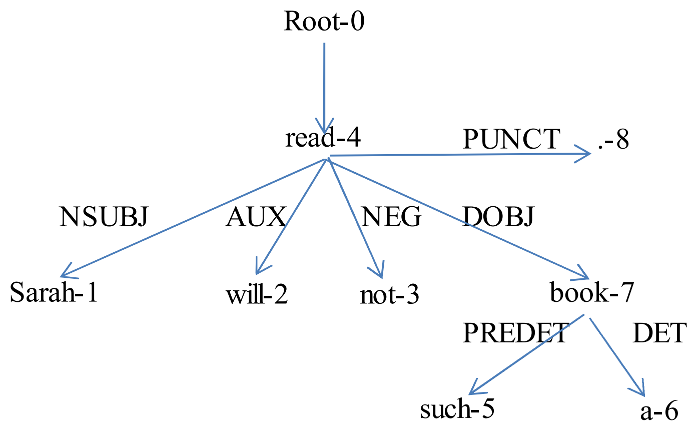


Figure 5.209. The typed-dependency tree for ‘Sarah will not read such a book.’

```
     | PRED    'read<NSUBJ, DOBJ>'                                 ||
     | NSUBJ  | PRED      'Sarah_j'          |                     ||
     |        | PERSON    3RD                |                     ||
     |        | NUMBER    SINGULAR           |                     ||
     |        | GENDER    FEMININE           |                     ||
     |                                                             ||
     | DOBJ   | PRED      'book'                           |       ||
     |        | PERSON    3RD                              |       ||
     |        | NUMBER    SINGULAR                         |       ||
     |        | GENDER    NEUTRAL                          |       ||
     |        | DET      | FORM     'a'          |         |       ||
     |        |          | TYPE     INDEFINITE   |         |       ||
     |        |                                           |       ||
     |        | PREDET   | FORM     'such'       |         |       ||
     |                                                             ||
     | AUX    | FORM      'will'             |                     ||
     |        | TENSE     FUTURE             |                     ||
     |                                                             ||
     | NEG    | FORM      'not'              |                     ||
     |                                                             ||
     | PUNCT  | FORM      '.'                |                     ||
|ROOT|        | STMT-TYPE DECLARATIVE        |                     ||
```

Figure 5.210. The functional structure for ‘Sarah will not read such a book.’

This dependency type does not completely follow Mel’čuk’s Criterion A. The linear order of the words in this type is such that the predeterminer precedes the determiner. However, a predeterminer and a noun do not constitute a prosodic unit (e.g., ‘all … books’ in the example sentence above). Therefore, this type does not follow Criterion A. On the other hand, this type forms a semantic unit because the predeterminer modifies the meaning of the noun. In this way, this type follows the revised Criterion A proposed in Section 2.4.3.1.

This dependency type follows Mel’čuk’s Criterion B because the passive valence of the phrase is determined by the noun. For example, the dependency relation between ‘all’ and ‘noun’ in the sentence ‘Sarah has read all the books she has’ is such that ‘all’ depends on ‘noun,’ which can be subordinated to the main predicate of the sentence.

This dependency type also follows the revised Criterion C, because it implies a certain kind of semantic relationship between a noun and a predeterminer. This semantic relationship cannot be expressed by any other dependency type, and the prototypical dependents of this dependency type include words such as ‘all’ and ‘such.’ Therefore, this dependency type follows the revised Criteria C1 and C2 proposed in Section 2.4.3.3.

Preconj – preconjunct

The dependency type “preconj” is a subtype of “mod” whereby the dependency head is one of the conjuncts and the dependent is a preconjunct, such as ‘both,’ ‘either,’ or ‘neither.’ As previously mentioned, the Stanford Parser treats conjuncts asymmetrically, i.e., one conjunct depends on the other. For the dependency type “preconj,” the first conjunct acts as the head of the dependency, and the preconjunct word acts as the tail. The following three typed-dependency trees present examples of this type.

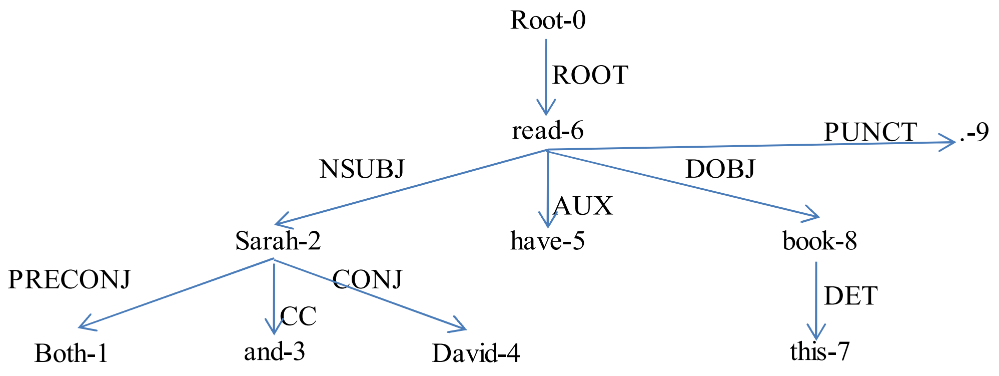


Figure 5.211. The asymmetric typed-dependency tree for ‘Both Sarah and David have read this book.’

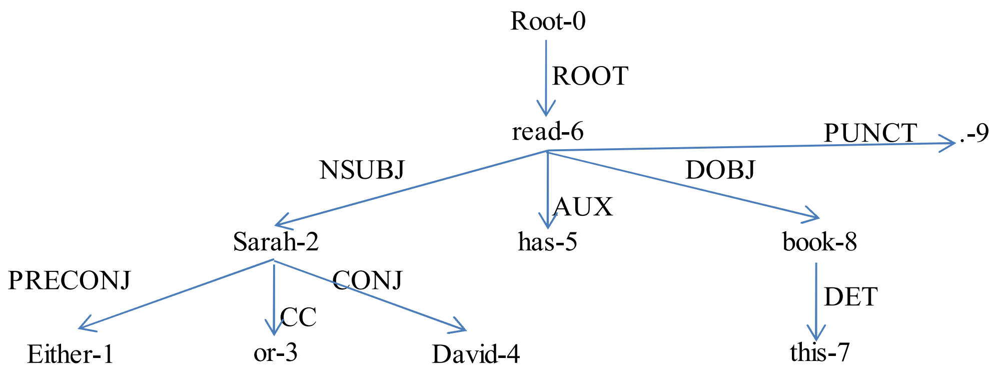


Figure 5.212. The asymmetric typed-dependency tree for ‘Either Sarah or David has read this book.’

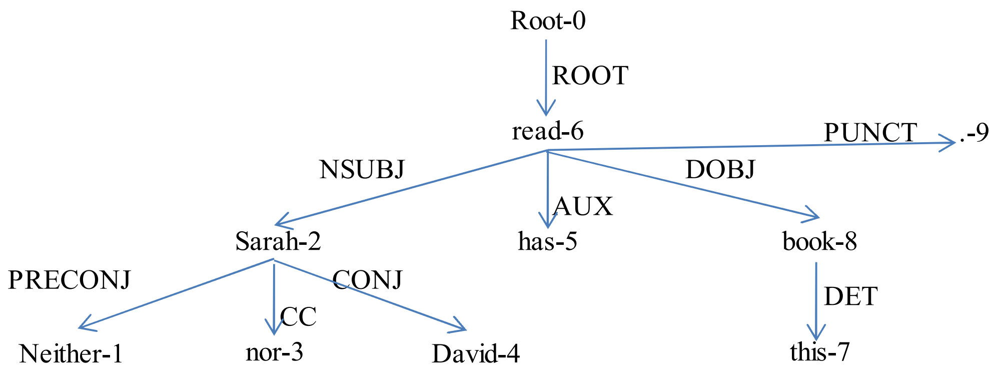


Figure 5.213. The asymmetric typed-dependency tree for ‘Neither Sarah nor David has read this book.’

The correct typed-dependency tree, however, has a *symmetrical* structure in which the conjuncts depend on the same head with the same dependency type. This study applies this symmetrical-structure analysis to the dependency type “preconj.” This means that the dependency type “preconj” should be replaced with other dependency types. For example, in the tree above, “conj” should be replaced with “nsubj.” In addition, the dependency relationship must be changed. Thus, in the example above, ‘Sarah’ and ‘David’ should be analyzed as conjuncts depending on ‘both.’ In this way, the dependency types between ‘both’ and ‘Sarah’ and between ‘both’ and ‘David’ are both considered “conj.” The modified tree as a result of these changes is shown in the figure below.

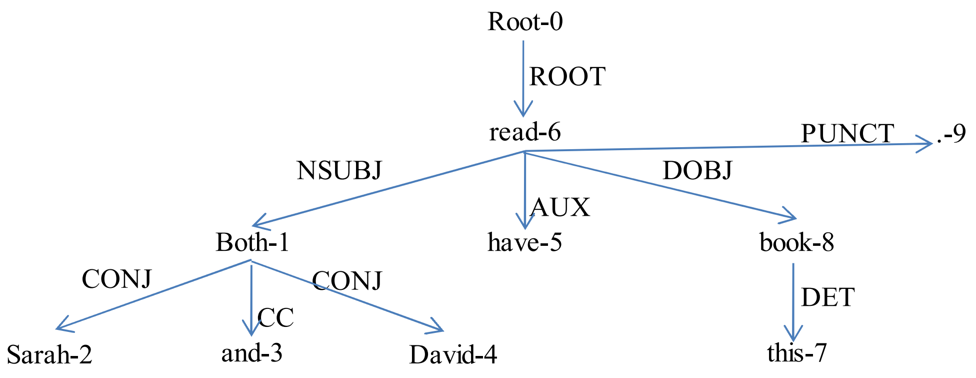


Figure 5.214. The symmetric typed-dependency tree for ‘Both Sarah and David have read this book.’

The typed-dependency tree above is equivalent to the functional structure below.

```
      | PRED    'read<NSUBJ, DOBJ>'                                   | |
      | NSUBJ   | PRED     'PRO'                                    |   | |
      |         | PERSON   3RD                                      |   | |
      |         | NUMBER   PLURAL                                   |   | |
      |         | FORM     'both'                                   |   | |
      |         | CONJ     | PRED      'Sarah_j'      |           |   | |
      |         |          | PERSON    3RD            |           |   | |
      |         |          | NUMBER    SINGULAR       |           |   | |
      |         |          | GENDER    FEMININE       |           |   | |
      |         | CC       FORM        'and'                      |   | |
      |         | CONJ     | PRED      'David_j'      |           |   | |
      |         |          | PERSON    3RD            |           |   | |
      |         |          | NUMBER    SINGULAR       |           |   | |
      |         |          | GENDER    MASCULINE      |           |   | |
      |                                                                 | |
      | DOBJ    | PRED     'book'                                   |   | |
      |         | PERSON   3RD                                      |   | |
      |         | NUMBER   SINGULAR                                 |   | |
      |         | GENDER   NEUTRAL                                  |   | |
      |         | DET      | FORM      'this'         |           |   | |
      |         |          | TYPE      DEFINITE       |           |   | |
      |                                                                 | |
      | AUX     | FORM      'have'       |                              | |
      |         | TENSE     PRESENT      |                              | |
      |         | ASPECT    PERFECT      |                              | |
      |                                                                 | |
      | PUNCT   | FORM      '.'          |                              | |
|ROOT |         | STMT-TYPE DECLARATIVE  |                              | |
```

Figure 5.215. The functional structure for ‘Both Sarah and David have read this book.’

Num (numeric modifier) and Quantmod (quantifier modifier)

The dependency type “num” is another subtype of “mod” and describes cases where the dependency head is a noun and the dependent is a numeric modifier. The dependency type “quantmod” is also a subtype of “mod” and represents cases where the dependency head is a number and the dependent is any element modifying the number (known as a *quantifier modifier*, or quantifier for short). Examples of these dependency types are shown below. In the figure below, the numeric modifier “200” modifies the noun “books.” In other words, the numeric modifier “200” depends on the noun “books” with the dependency type “num.” In addition, the word “about” quantifies the numeric modifier “200.” In other words, the word ‘about’ depends on ‘200’ with the dependency type “quantmod.”

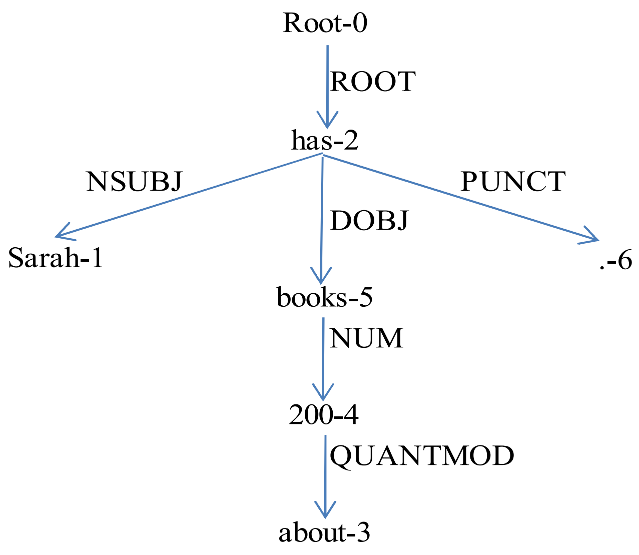


Figure 5.216. The typed-dependency tree for ‘Sarah has about 200 books.’

```
|    |PRED    'have<NSUBJ, DOBJ>'                          ||
|    |NSUBJ  |PRED      'Sarah'    |                       ||
|    |       |PERSON    3RD        |                       ||
|    |       |NUMBER    SINGULAR   |                       ||
|    |       |GENDER    FEMININE   |                       ||
|    |                                                     ||
|    |DOBJ   |PRED      'books'                        |   ||
|    |       |PERSON    3RD                            |   ||
|    |       |NUMBER    PLURAL                         |   ||
|    |       |GENDER    NEUTRAL                        |   ||
|    |       |NUM      |PRED       '200'            |  |   ||
|    |       |         |QUANTMOD  |PRED  'about'|   |  |   ||
|    |TENSE   PRESENT                                      ||
|    |PUNCT  |FORM      '.'         |                      ||
|ROOT|       |STMT-TYPE DECLARATIVE |                      ||
```

Figure 5.217. The functional structure for ‘Sarah has about 200 books.’

More than one quantifier modifier can modify a noun. In the figure below, the phrase ‘less than’ modifies the numeric modifier ‘200.’ Therefore, each word in the phrase ‘less than’ depends on ‘200’ with the dependency type “quantmod.”

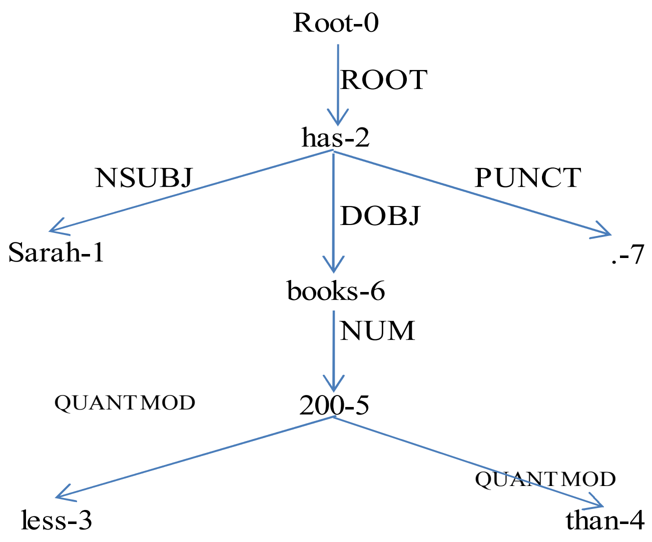


Figure 5.218. The typed-dependency tree for ‘Sarah has less than 200 books.’

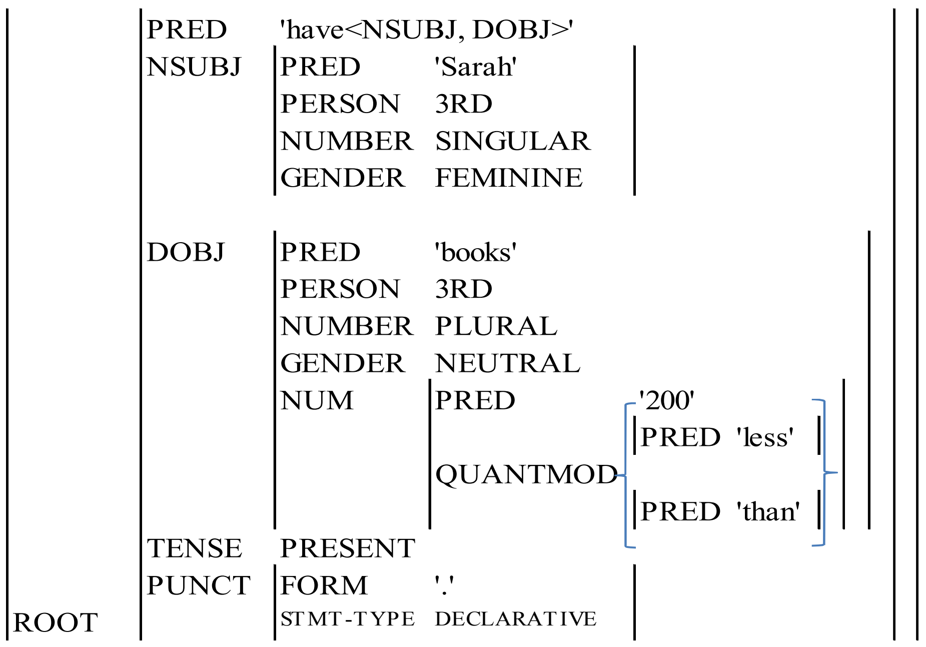


Figure 5.219. The functional structure for ‘Sarah has less than 200 books.’

The dependency type ‘num’ follows Mel’čuk’s original Criterion A, yet the dependency type “quantmod” does not always follow Criterion A. A numerical modifier and a noun constitute a prosodic unit (e.g., ‘200 books’ in the example sentence above). On the other hand, there are instances in which a quantifier and a numerical modifier do not constitute a prosodic unit (e.g., ‘less … 200’ in the example above). However, both of these types form a semantic unit because a numerical modifier modifies the meaning of the noun, and a quantifier modifies the meaning of a numerical modifier. In addition, the linear order of the words in these dependency relations is such that the quantifier or the numerical modifier precedes the noun. Therefore, these types follow Criterion A.

These dependency types also follow Mel’čuk’s Criterion B. For example, the dependency relation between ‘200’ and ‘book’ in the example sentence ‘Sarah has about 200 books’ shows that ‘200’ depends on ‘books,’ which can be subordinated to the verb ‘has.’ The passive valence of the phrase is determined by the noun ‘books.’ Similarly, the dependency relation between ‘about’ and ‘200’ in the same sentence shows that ‘about’ depends on ‘200.’

In addition, these dependency types follow the revised Criterion C because “num” implies a certain kind of semantic relationship between a noun and a numerical modifier, and “quantmod” implies a semantic relationship between a numerical modifier and a quantifier. These semantic relationships cannot be expressed by any other dependency type, and the prototypical dependents of “num” are numbers, and those of “quantmod” include adverbs such as ‘about’ or ‘almost.’ Therefore, this dependency type follows the revised Criteria C1 and C2 proposed in Section 2.4.3.3.

Number - element of compound number

The dependency type “number” is another subtype of “mod” whereby the dependent is an element of a compound number. The typed-dependency trees below present examples of this subtype.

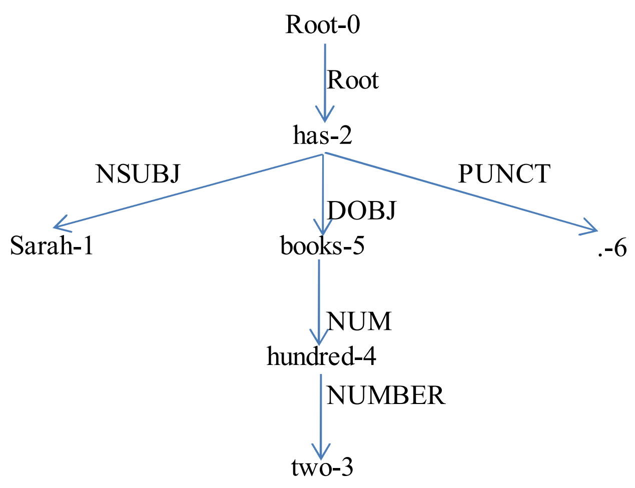


Figure 5.220. The typed-dependency tree for ‘Sarah has two hundred books.’

```
|         | PRED    'have<NSUBJ, DOBJ>'                          | |
|         | NSUBJ   | PRED       'Sarah'    |                   | |
|         |         | PERSON     3RD        |                   | |
|         |         | NUMBER     SINGULAR   |                   | |
|         |         | GENDER     FEMININE   |                   | |
|         |                                                     | |
|         | DOBJ    | PRED       'books'                      | | |
|         |         | PERSON     3RD                          | | |
|         |         | NUMBER     PLURAL                       | | |
|         |         | GENDER     NEUTRAL                      | | |
|         |         | NUM        | PRED       'hundred'     | | | |
|         |         |            | NUMBER     |PRED 'two' | | | | |
|         | TENSE   PRESENT                                     | |
|         | PUNCT   | FORM       '.'          |                 | |
|ROOT     |         | STMT-TYPE  DECLARATIVE  |                 | |
```

Figure 5.221. The functional structure for ‘Sarah has two hundred books.’

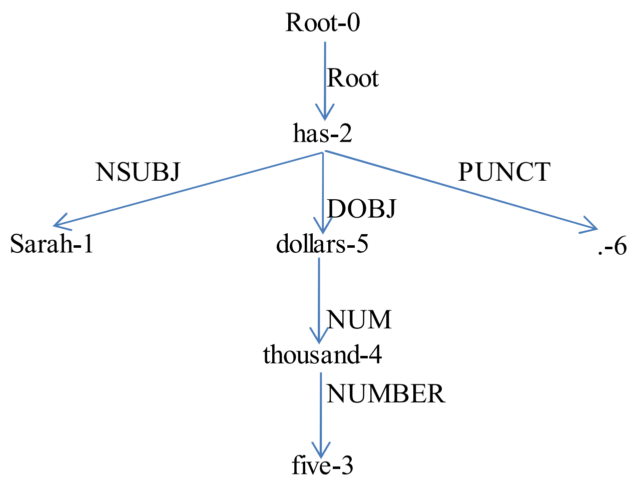


Figure 5.222. The typed-dependency tree for ‘Sarah has five thousand dollars.’

```
     PRED     'have<NSUBJ, DOBJ>'
     NSUBJ    PRED     'Sarah'
              PERSON   3RD
              NUMBER   SINGULAR
              GENDER   FEMININE

     DOBJ     PRED     'dollars'
              PERSON   3RD
              NUMBER   PLURAL
              GENDER   NEUTRAL
              NUM      PRED      'thousand'
                       NUMBER    PRED 'five'
     TENSE    PRESENT
     PUNCT    FORM     '.'
ROOT          STMT-TYPE DECLARATIVE
```

Figure 5.223. The functional structure for ‘Sarah has five thousand dollars.’

The dependency type “number” follows Mel’čuk’s Criterion A because a compound number constitutes a prosodic unit (e.g., ‘five thousand’ in the example sentence above), and the word order is fixed.

This dependency type also follows Mel'čuk's Criterion B. For example, the dependency relation between 'five' and 'million' in the example sentence 'Sarah has five million dollars' is such that 'five' depends on 'million,' which can be subordinated to the noun 'dollars.' The passive valence of the phrase is determined by the noun 'million.'

In addition, this dependency type follows the revised Criterion C because it implies a semantic relationship between numbers that cannot be expressed by any other dependency type, and the prototypical dependents of "number" are numbers. Therefore, this dependency type follows the revised Criteria C1 and C2 proposed in Section 2.4.3.3. Notice that the difference between "num" and "number" resides in the different prototypical heads for these dependency types. That is, the head of "num" is a noun, while that of "number" is another number.

Nn - noun compound modifier

This dependency type is a subtype of "mod" that describes cases where the dependent is an element in a noun compound. An example of this subtype is shown below. In the figure below, the noun 'language' modifies the word 'acquisition' as a noun compound. In other words, the word 'language' depends on 'acquisition' with the dependency type "nn."

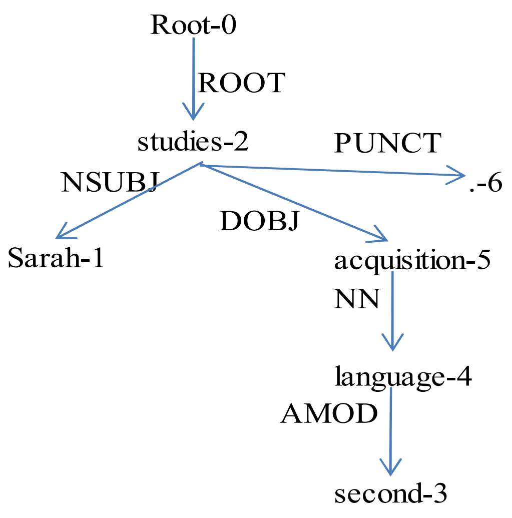


Figure 5.224. The typed-dependency tree for ‘Sarah studies second language acquisition.’

```
|     |PRED    'study<NSUBJ, DOBJ>'                                 | |
|     |NSUBJ  |PRED      'Sarah'    |                               | |
|     |       |PERSON    3RD        |                               | |
|     |       |NUMBER    SINGULAR   |                               | |
|     |       |GENDER    FEMININE   |                               | |
|     |                                                             | |
|     |DOBJ   |PRED      'acquisition'                          |   | |
|     |       |PERSON    3RD                                    |   | |
|     |       |NUMBER    SINGULAR                               |   | |
|     |       |GENDER    NEUTRAL                                |   | |
|     |       |NN       |PRED      'language'               |   |   | |
|     |       |         |PERSON    3RD                      |   |   | |
|     |       |         |NUMBER    SINGULAR                 |   |   | |
|     |       |         |GENDER    NEUTRAL                  |   |   | |
|     |       |         |AMOD     |PRED  'second'|          |   |   | |
|     |TENSE   PRESENT                                            | |
|     |PUNCT  |FORM      '.'        |                               | |
|ROOT |       |STMT-TYPE DECLARATIVE|                               | |
```

Figure 5.225. The functional structure for ‘Sarah studies second language acquisition.’

A compound noun can be formed by more than two nouns. In the typed-dependency tree below, the words ‘noun,’ ‘compound’ and ‘modifiers’ form a compound noun ‘noun compound modifiers.' In this compound noun, the nouns are aligned *asymmetrically*. In other words, the first noun depends on the second, and the second noun depends on the third.

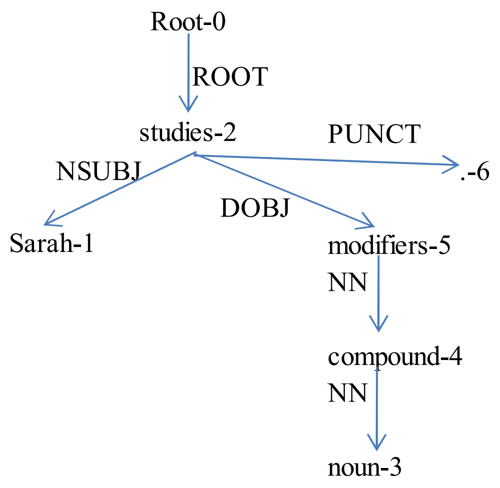


Figure 5.226. The typed-dependency tree for 'Sarah studies noun compound modifiers.'

However, the Stanford Parser parses a noun compound *symmetrically*. In other words, both the first noun and the second noun depend on the third noun.

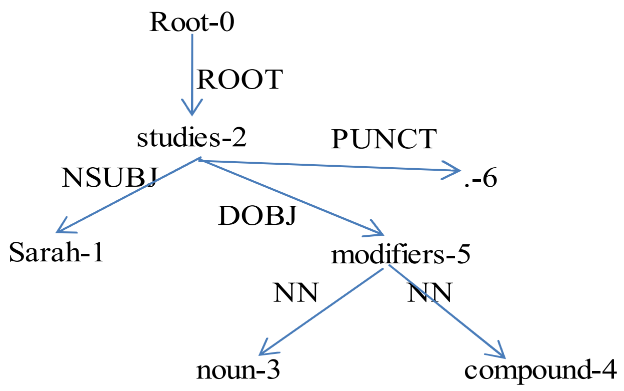


Figure 5.227. The typed-dependency tree for 'Sarah studies noun compound modifiers' in Stanford Parser output.

De Marneffe & Manning (2012, p.6) state that all the nouns in a compound noun are parsed to modify the rightmost noun of the compound noun in the current version of the Stanford Parser, and that this will be fixed when the Penn Treebank, the corpus with which the parser has been trained, represents the branching structure of NPs.

This dependency type follows Mel'čuk's Criterion A because a noun compound constitutes a prosodic unit (e.g., 'language acquisition' in the example sentence below), and the word order is fixed.

This dependency type also follows Mel'čuk's Criterion B. For example, the dependency relation between 'language' and 'acquisition' in the example sentence 'Sarah is learning language acquisition' shows that 'language' depends on 'acquisition,' which can be subordinated to the verb 'learning.' The passive valence of the phrase is determined by the noun 'acquisition.'

In addition, this dependency type follows the revised Criterion C, because it implies a certain kind of semantic relationship between nouns that cannot be expressed by any other dependency type, and the prototypical dependents of "nn" are nouns. Therefore, this dependency type follows the revised Criteria C1 and C2 proposed in Section 2.4.3.3.

Abbrev - abbreviation modifier

This dependency type is another subtype of "mod" and it describes cases where the dependent is an abbreviation represented by a parenthesized NP. In the parser output below, the expression "-lrb-" stands for 'left round bracket' and "-rrb-" stands for 'right round bracket.'

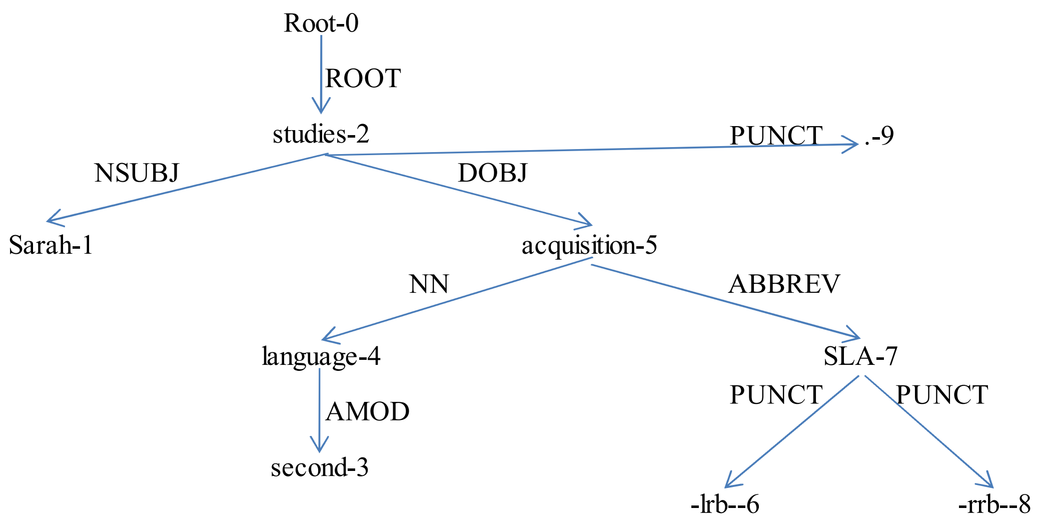


Figure 5.228. The typed-dependency tree for ‘Sarah studies second language acquisition (SLA).’

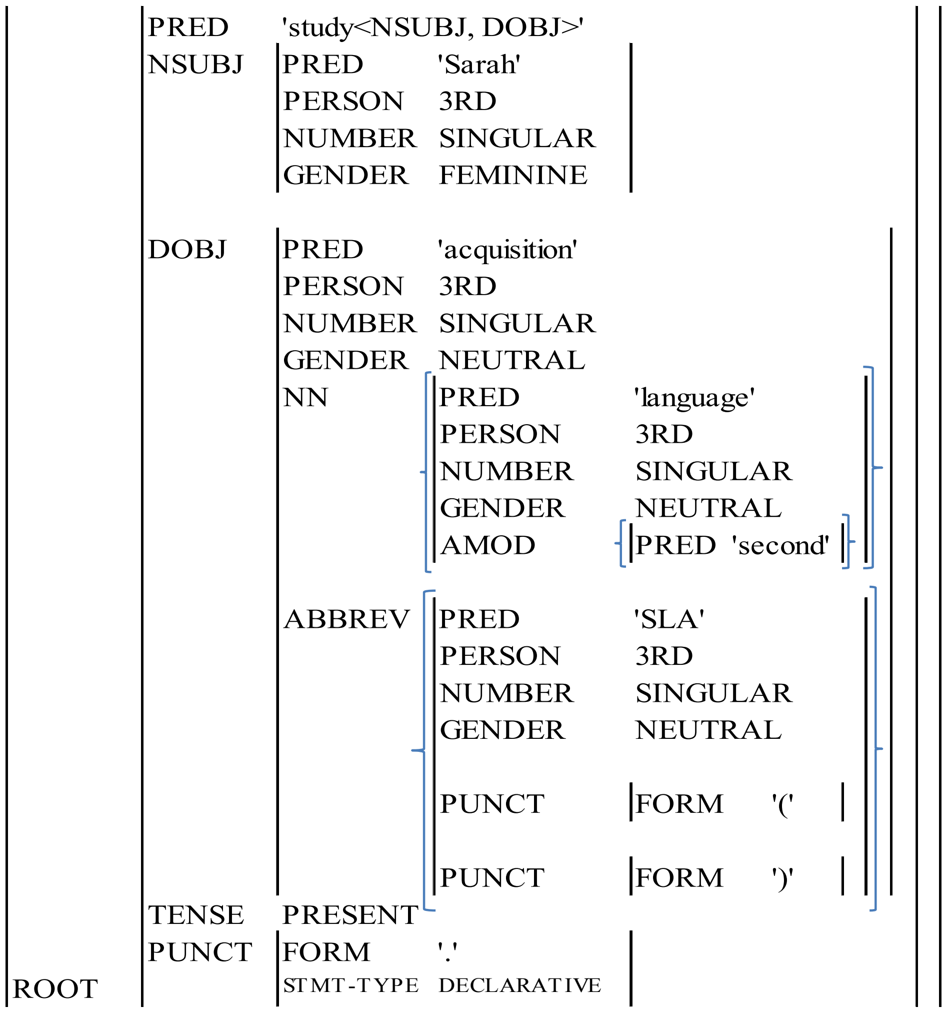


Figure 5.229. The functional structure for ‘Sarah studies second language acquisition (SLA).’

The dependency type “abbrev” does not follow Mel’čuk’s Criterion A, because an abbreviation depends on the last noun in a noun compound, and this noun and the abbreviation do not constitute a prosodic unit (e.g., ‘acquisition’ and ‘SLA’ in the example sentence). Semantically, an abbreviation is a type of apposition, i.e., the noun compound and its abbreviation are in apposition. For example, the sentence ‘Sarah studies second language acquisition (SLA)’ can be paraphrased as ‘Sarah studies second language acquisition, or SLA.’

This dependency type follows Mel’čuk’s Criterion B. It is not the abbreviation, but the noun, that determines the passive valence of the phrase. For example, the dependency relation between ‘acquisition’ and ‘SLA’ in the sentence ‘Sarah studies second language acquisition (SLA)’ shows that ‘SLA’ depends on the adjective ‘acquisition,’ which can be subordinated to the verb ‘studies.’

This dependency type follows the revised Criteria C1 and C2 proposed in Section 2.4.3.3. This dependency type must be treated differently from “conj” because it implies a particular semantic relationship between a noun compound and its abbreviation. That is, the abbreviation of the noun compound is a kind of paraphrase because it refers to the same entity using different words, and this semantic relationship cannot be expressed by the dependency type “conj.”

Npadvmod – noun phrase adverbial modifier

The dependency type “npadvmod” is a subtype of “mod” whereby the dependency head is an adjective and the dependent is a noun that adverbially modifies the head adjective, as shown in the figures below.

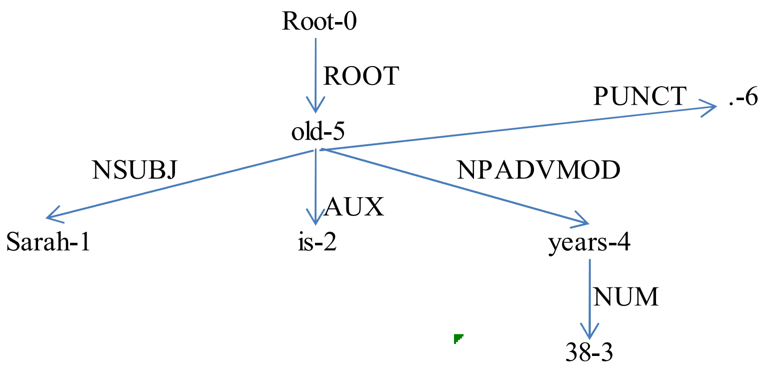


Figure 5.230. The typed-dependency tree for ‘Sarah is 38 years old.’

```
|       |PRED          'old'                          | |
|       |NSUBJ         |PRED      'Sarah'     |       | |
|       |              |PERSON    3RD         |       | |
|       |              |NUMBER    SINGULAR    |       | |
|       |              |GENDER    FEMININE    |       | |
|       |                                             | |
|       |COP           |FORM      'is'        |       | |
|       |              |TENSE     PRESENT     |       | |
|       |                                             | |
|       |NPADVMOD      |PRED      'years'       |     | |
|       |              |PERSON    3RD           |     | |
|       |              |NUMBER    PLURAL        |     | |
|       |              |GENDER    NEUTRAL       |     | |
|       |              |NUM       |PRED '38'  | |     | |
|       |                                             | |
|       |PUNCT         |FORM      '.'         |       | |
|ROOT   |              |STMT-TYPE DECLARATIVE |       | |
```

Figure 5.231. The functional structure for ‘Sarah is 38 years old.’

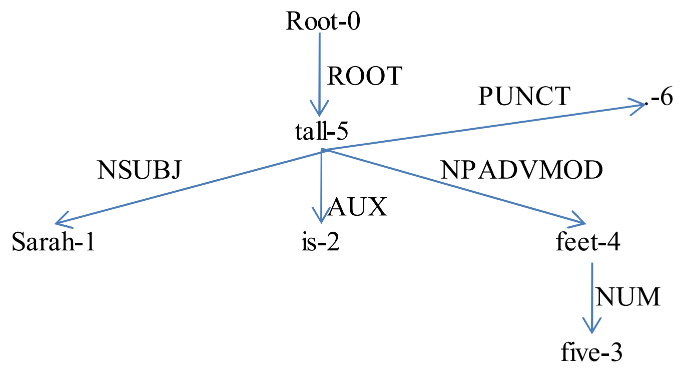


Figure 5.232. The typed-dependency tree for ‘Sarah is five feet tall.’

$$\left[\begin{array}{ll} & \left[\begin{array}{ll} \text{PRED} & \text{'tall'} \\ \text{NSUBJ} & \left[\begin{array}{ll} \text{PRED} & \text{'Sarah'} \\ \text{PERSON} & \text{3RD} \\ \text{NUMBER} & \text{SINGULAR} \\ \text{GENDER} & \text{FEMININE} \end{array}\right] \\ \text{COP} & \left[\begin{array}{ll} \text{FORM} & \text{'is'} \\ \text{TENSE} & \text{PRESENT} \end{array}\right] \\ \text{NPADVMOD} & \left[\begin{array}{ll} \text{PRED} & \text{'feet'} \\ \text{PERSON} & \text{3RD} \\ \text{NUMBER} & \text{PLURAL} \\ \text{GENDER} & \text{NEUTRAL} \\ \text{NUM} & \left[\begin{array}{l} \text{PRED 'five'} \end{array}\right] \end{array}\right] \\ \text{PUNCT} & \left[\begin{array}{ll} \text{FORM} & \text{'.'} \\ \text{STMT-TYPE} & \text{DECLARATIVE} \end{array}\right] \end{array}\right] \\ \text{ROOT} & \end{array}\right]$$

Figure 5.233. The functional structure for ‘Sarah is five feet tall.’

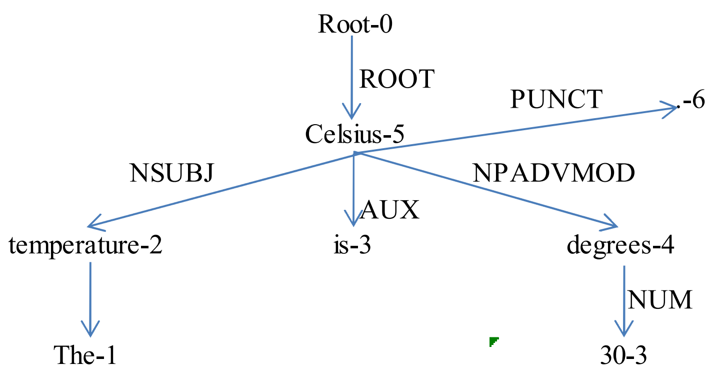


Figure 5.234. The typed-dependency tree for ‘The temperature is 30 degrees Celsius.’

```
|     |PRED       'Celsius'                                   ||
|     |NSUBJ      |PRED     'temperature'                   | ||
|     |           |PERSON   3RD                             | ||
|     |           |NUMBER   SINGULAR                        | ||
|     |           |GENDER   NEUTRAL                         | ||
|     |           |DET      |FORM       'the'      |        | ||
|     |           |         |TYPE       DEFINITE   |        | ||
|     |                                                       ||
|     |COP        |FORM     'is'        |                     ||
|     |           |TENSE    PRESENT     |                     ||
|     |                                                       ||
|     |NPADVMOD   |PRED     'degrees'     |                   ||
|     |           |PERSON   3RD           |                   ||
|     |           |NUMBER   PLURAL        |                   ||
|     |           |GENDER   NEUTRAL       |                   ||
|     |           |NUM      |PRED '30'  | |                   ||
|     |                                                       ||
|     |PUNCT      |FORM       '.'         |                   ||
|ROOT |           |STMT-TYPE  DECLARATIVE |                   ||
```

Figure 5.235. The functional structure for ‘The temperature is 30 degrees Celsius.’

The dependency type “npadvmod” does not completely follow Mel’čuk’s Criterion A because an adjective and the noun that modifies it do not always constitute a prosodic unit (e.g., ‘feet tall’); however, the linear order is fixed. On the other hand, this dependency type follows the revised Criterion A because it forms a semantic unit.

This dependency type follows Mel'čuk's Criterion B. The adjective determines the passive valence of the phrase. For example, the dependency relation between 'tall' and 'feet' in the sentence 'Sarah is five feet tall' shows that 'feet' depends on the adjective 'tall,' which can be subordinated to the root node.

This dependency type also follows the revised Criterion C, because it implies a certain kind of semantic relationship between an adjective and a noun that cannot be expressed by any other dependency type. The prototypical dependent of this dependency type is a noun.

Mwe – multi-word modifier

The dependency type "mwe" is a subtype of "mod" for cases where the dependency head contains "certain multi-word idioms that behave like a single function word" (de Marneffe & Manning 2012, p.6). They state that the Stanford Parser at present uses this dependency type for the following multi-word idioms: 'rather than,' 'as well as,' 'instead of,' 'such as,' 'because of,' 'in addition to,' 'all but,' 'such as,' and 'due to.' However, the Stanford Parser seems to have difficulties to parse sentences containing one of these multi-word idioms, because the parsed outputs for such sentences are often incorrect. For example, the figure below shows the parsed output for a sentence with the phrase 'as well as.' The noun 'dogs' and 'cats' are aligned asymmetrically, and the first 'as' depends on the word 'well' with the dependency type "advmod" whereas the second 'as' also depends on the word 'well' with the dependency type "mwe." The word 'well' depends on the noun 'dogs,' the first conjunct in this sentence, with the dependency type "cc" (coordination).

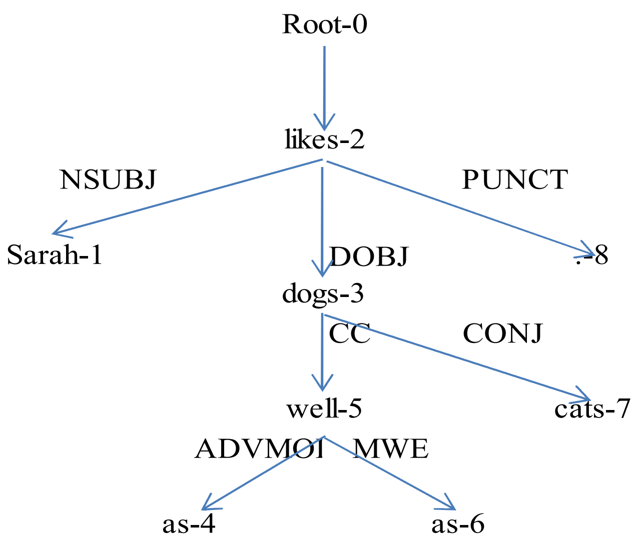


Figure 5.236. The typed-dependency tree for ‘Sarah likes dogs as well as cats.’ in the Stanford Parser output

A better typed-dependency tree is such that the conjuncts are aligned symmetrically, the phrase ‘as well as’ depends on the verb with the dependency type “mwe”, and the first and the second ‘as’ both depends on the word ‘well’ with the dependency type “mwe”, as shown below. The fact that the first and second ‘as’ both depend on the word ‘well’ with the dependency type “mwe” ensures that these three words are united through the same dependency types.

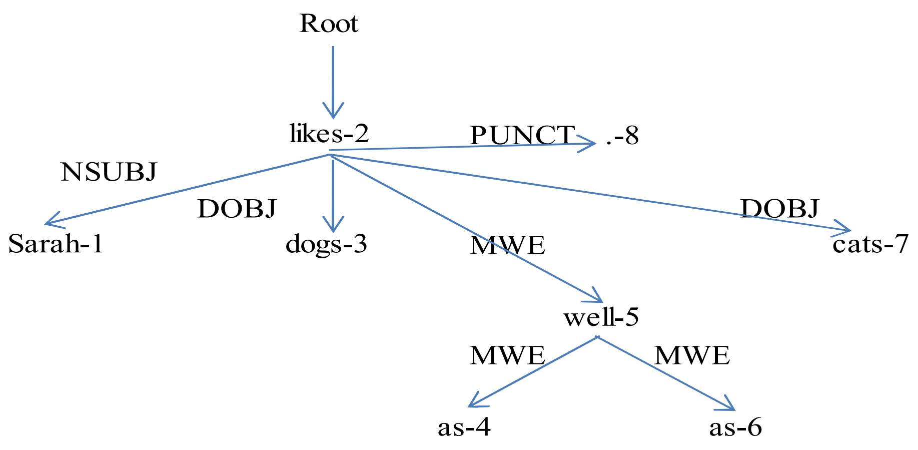


Figure 5.237. A better typed-dependency tree for ‘Sarah likes dogs as well as cats.’

The typed-dependency tree above is equivalent to the functional structure below. Notice that the “dobj” attribute has a set of values, whose elements are ‘cats’ and ‘dogs.’

```
|          |PRED    'like<NSUBJ,DOBJ>'                               | |
|          |NSUBJ   |PRED        'Sarah'     |                       | |
|          |        |PERSON      3RD         |                       | |
|          |        |NUMBER      SINGULAR    |                       | |
|          |        |GENDER      FEMININE    |                       | |
|          |                                                         | |
|          |       { |PRED        'dogs'      | }                    | |
|          |       { |PERSON      3RD         | }                    | |
|          |       { |NUMBER      PLURAL      | }                    | |
|          |       { |GENDER      NEUTRAL     | }                    | |
|          |DOBJ - {                            }-                   | |
|          |       { |PRED        'cats'      | }                    | |
|          |       { |PERSON      3RD         | }                    | |
|          |       { |NUMBER      PLURAL      | }                    | |
|          |       { |GENDER      NEUTRAL     | }                    | |
|          |MWE     |FORM        'well'                 |            | |
|          |        |MWE         |FORM        'as' |    |            | |
|          |        |                                   |            | |
|          |        |            |FORM        'as' |    |            | |
|          |TENSE   PRESENT                                          | |
|          |PUNCT   |FORM        '.'         |                       | |
|ROOT      |        |STMT-TYPE   DECLARATIVE |                       | |
```

Figure 5.238. The functional structure for ‘Sarah likes dogs as well as cats.’

Other multi-word expressions can be represented in the same way described above as well. The Stanford Parser output for a sentence including the multi-word expression ‘rather than’ is shown below.

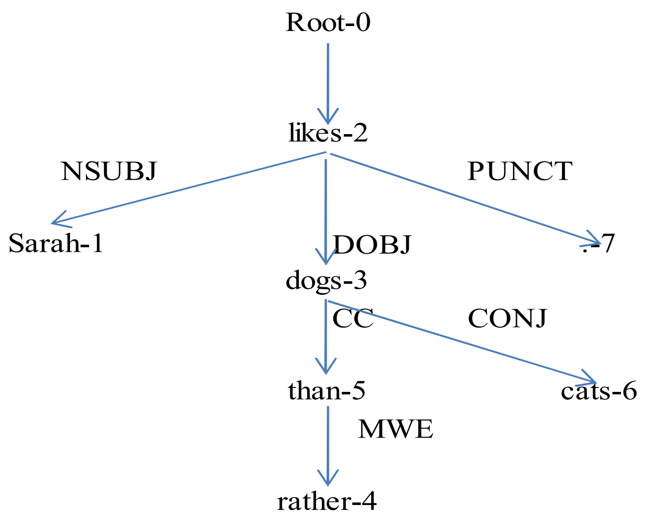


Figure 5.239. The typed-dependency tree for ‘Sarah likes dogs rather than cats.’ in the Stanford Parser output

A better typed-dependency tree for the same sentence is shown below, in which the word ‘rather’ depends on ‘likes,’ the word ‘than’ depends on ‘rather,’ and the word ‘cats’ depends on ‘than.’ The type of all of these dependencies is “mwe.” This analysis is chosen in this study, and in the manual correction of the Stanford Parser output (see Section 7.3), the parsed output for the ‘rather than’ construction will be corrected as such.

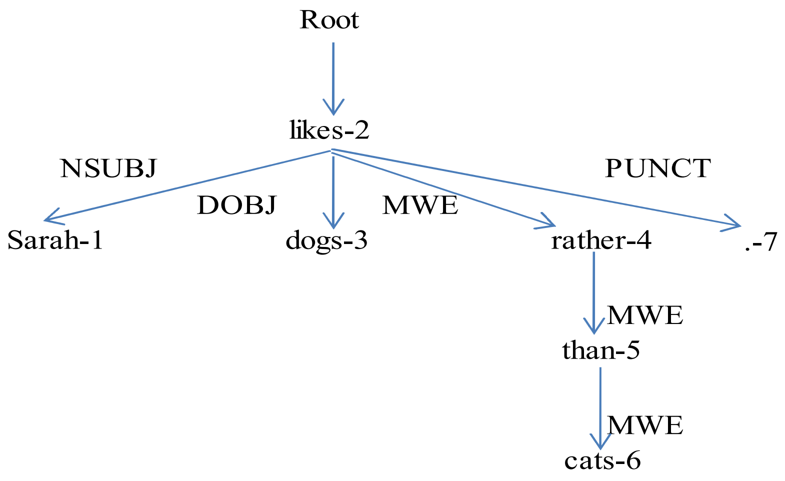


Figure 5.240. A better typed-dependency tree for ‘Sarah likes dogs rather than cats.’

```
|        |PRED    'like<NSUBJ,DOBJ>'                                      | |
|        |NSUBJ  |PRED        'Sarah'    |                                | |
|        |       |PERSON      3RD        |                                | |
|        |       |NUMBER      SINGULAR   |                                | |
|        |       |GENDER      FEMININE   |                                | |
|        |                                                                | |
|        |DOBJ   |PRED        'dogs'     |                                | |
|        |       |PERSON      3RD        |                                | |
|        |       |NUMBER      PLURAL     |                                | |
|        |       |GENDER      NEUTRAL    |                                | |
|        |                                                                | |
|        |MWE   [|FORM        'rather'                                 ||]| |
|        |       |MWE        [|FORM       'than'                     |]|  | |
|        |       |            |MWE       |PRED      'cats'         |  |  | |
|        |       |            |          |PERSON    3RD            |  |  | |
|        |       |            |          |NUMBER    PLURAL         |  |  | |
|        |       |            |          |GENDER    NEUTRAL        |]|]| | |
|        |TENSE   PRESENT                                                 | |
|        |PUNCT  |FORM        '.'          |                              | |
|ROOT    |       |STMT-TYPE   DECLARATIVE  |                              | |
```

Figure 5.241. The functional structure for ‘Sarah likes dogs rather than cats.’

5.3.5 Parataxis

This dependency type is used to represent a parataxis relation between clauses, as shown in the typed-dependency tree below.

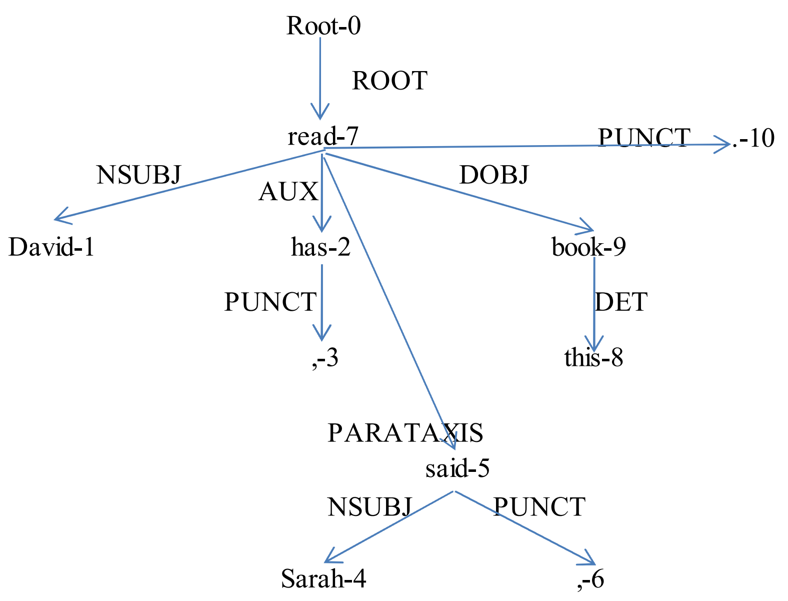


Figure 5.242. The typed-dependency tree for ‘David has, Sarah said, read this book.’

The typed-dependency tree above is equivalent to the functional structure below.

```
|     |PRED       'read<NSUBJ,DOBJ>'                          ||
|     |NSUBJ      |PRED 'Dabid'              |                 ||
|     |           |PERSON  3RD               |                 ||
|     |           |NUMBER SINGULAR           |                 ||
|     |           |GENDER  MASCULINE         |                 ||
|     |                                                        ||
|     |DOBJ       |PRED 'book'                     |           ||
|     |           |PERSON  3RD                     |           ||
|     |           |NUMBER SINGULAR                 |           ||
|     |           |GENDER  NEUTRAL                 |           ||
|     |           |DET     |FORM 'this'     |      |           ||
|     |           |        |TYPE DEFINITE   |      |           ||
|     |                                                        ||
|     |AUX        |FORM     'has'                  |           ||
|     |           |TENSE    PRESENT                |           ||
|     |           |ASPECT   PERFECT                |           ||
|     |           |PUNCT   |FORM          ','  |   |           ||
|     |                                                        ||
|     |PARATAXIS [|PRED     'say<NSUBJ>'                     |]||
|     |           |NSUBJ   |PRED         'Sarah'       |     | ||
|     |           |        |PERSON       3RD           |     | ||
|     |           |        |NUMBER       SINGULAR      |     | ||
|     |           |        |GENDER       FEMININE      |     | ||
|     |           |TENSE    PAST                             | ||
|     |           |PUNCT   |FORM          ','  |           |]||
|     |                                                        ||
|     |PUCNT      |FORM     '.'                    |           ||
|ROOT |           |STMT-TYPE DECLARATIVE           |           ||
```

Figure 5.243. The typed-dependency tree for ‘David has, Sarah said, read this book.’

The dependency type “parataxis” does not follow Mel’čuk’s original Criterion A because the verb in the main clause and the verb in the paratactic clause do not form a prosodic unit. This is the case for the verbs ‘read’ and ‘said’ in the above example. However, the main clause verb and the subordinate verb form a semantic unit, and therefore this type follows the revised Criterion A proposed in Section 2.4.3.1.

This dependency type also follows Mel’čuk’s Criterion B. In the above example, the passive valence of the phrase ‘read … said’ is determined by the verb ‘read,’ because this verb further depends on the Root.

In addition, this dependency type follows the revised Criterion C given that it implies a certain kind of semantic relationship between verbs that cannot be expressed by any other dependency type, and the prototypical dependent of this dependency type is a verb. Therefore, this dependency type follows the revised Criteria C1 and C2 proposed in Section 2.4.3.3.

5.3.6. Punct

This dependency type is used to show the dependency relationship between a word and punctuation marks. As already shown in the example typed-dependency trees, a period and a question mark depend on the main predicate of a sentence, while a comma depends on the word that immediately precedes it. A period specifies the *statement type* (STMT-TYPE for short) of the sentence as *declarative*, while a question mark specifies the statement type of the sentence as *interrogative*[34]. A comma has a number of functions that we will not include in the functional-structure representation of a sentence.

5.4 Different Typed-Dependency Output Styles

The Stanford Parser provides us with five different output styles: basic dependencies, collapsed dependencies, collapsed dependencies with propagation of conjunct dependencies, collapsed

[34] Butt et al (1999, p.18) introduces the attribute-value pair STMT-TYPE DECLARATIVE as the default, and special constructions like interrogatives and imperatives provide their own statement types. As far as written texts are concerned, it is appropriate to consider that a period provides a sentence with the value DECLARATIVE for the attribute STMT-TYPE, and a question mark with the value INTERROGATIVE for the same attribute, because a sentence not in the interrogative construction can function as a question if accompanied by a question mark (e.g., ‘OK?’, ‘Sarah read this book?’).

dependencies preserving a tree structure, and non-collapsed dependencies. One of these styles is chosen as an option for parsing input text, and the default style is basic dependencies. Different styles are employed for different purposes. For example, if the acyclic-graph representation of dependencies is required, output styles that preserve acyclic-graph structure must be chosen. If the focus of the analysis is the semantic relationships among content words, collapsed dependencies provide more concise representation than the non-collapsed style. Each output style is discussed in detail below.

5.4.1 Basic dependencies

Basic dependencies are the output style shown in Section 3.3 whereby each word in a sentence, with the exception of the root, depends on another word. This output style preserves acyclic-graph structure.

In the output style "basic dependencies", relative pronouns are analyzed as they occur, and their referents are not specified, as shown in the figure below.

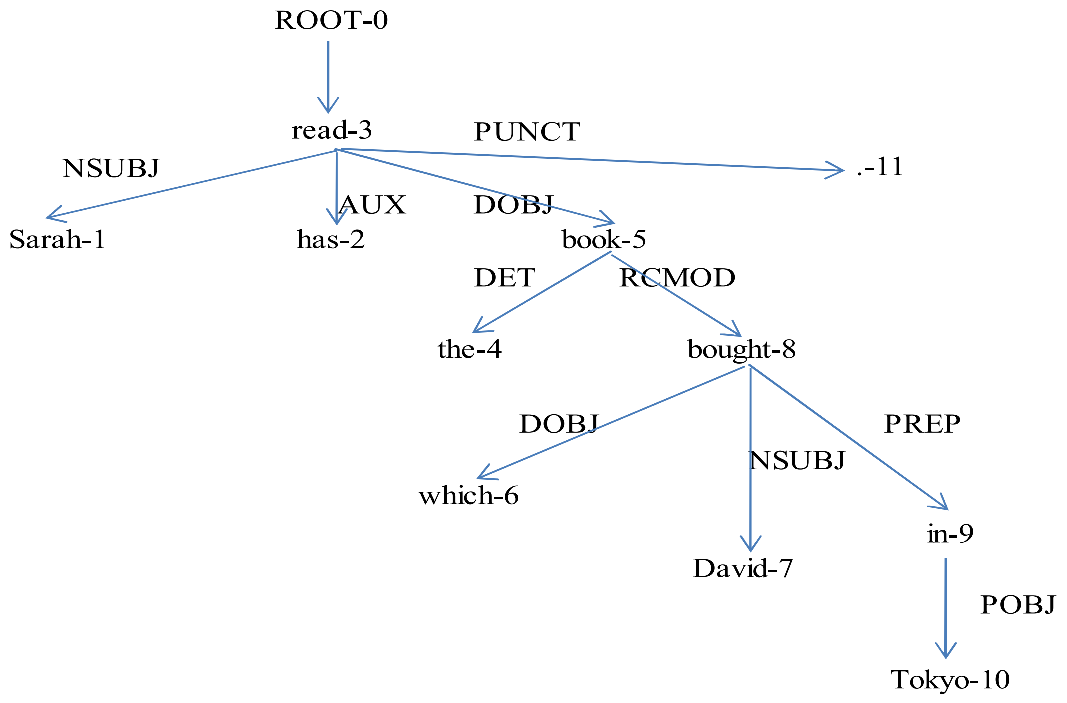


Figure 5.244. The typed-dependency tree for ‘Sarah has read the book which David bought in Tokyo.’ (in the output style “Basic dependencies”)

In this output style, conjuncts are analyzed asymmetrically. For example, in the figure below, the second ‘book’ depends on the first ‘book.’

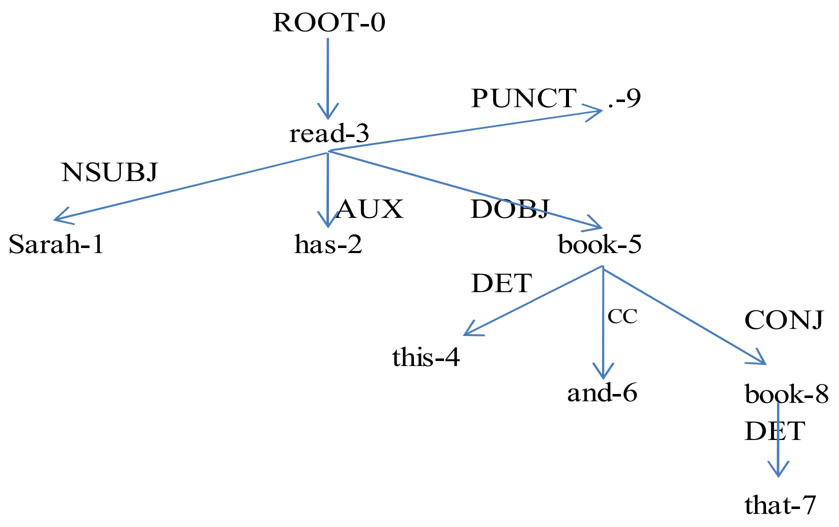


Figure 5.245. The typed-dependency tree for ‘Sarah has read this book and that book.’ (In the output style “Basic dependencies”)

5.4.2 Collapsed dependencies

Collapsed dependencies are designed where some of dependencies are collapsed to represent “direct dependencies between content words” (de Marneffe & Manning 2012, p.13). The collapsed dependencies in this output style are prepositions, conjuncts, and the referents of relative clauses. This output style does not preserve acyclic-graph structure because there is a cyclical dependency between the head of a relative clause and the word on which the relative clause depends. For example, in the dependency representation in the figure below, there is a cyclical relationship between ‘bought’ and ‘book’ where ‘bought’ depends on ‘book’ with the type “rcmod,” while ‘book’ depends on ‘bought’ with the type “dobj.” In addition to this cyclical analysis, this output style includes an extra dependency type “rel.” This dependency type is such that the head functions as the head of a relative clause, and the dependent functions as the relative pronoun that introduces the relative clause (see Section 5.3.4).

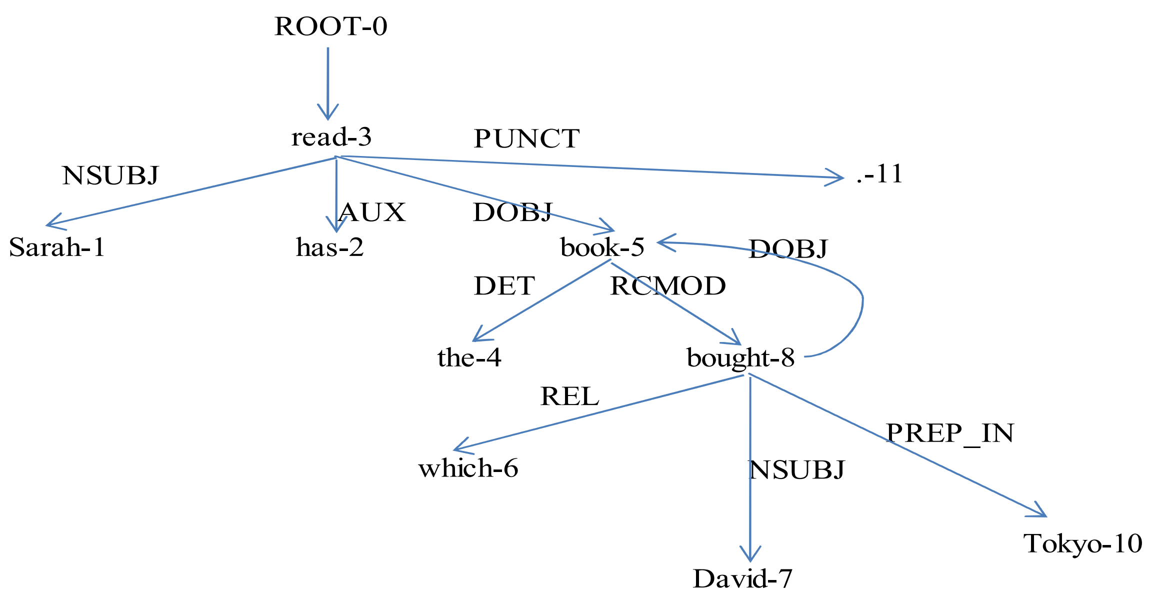


Figure 5.246. The dependency tree for 'Sarah has read the book which David bought in Tokyo.'

(In the output style "Collapsed dependencies")

In this output style, prepositions are collapsed and fused with the dependency type "prep." In the example above, the preposition 'in' is fused with the dependency type "prep" to yield another dependency type "prep_in," and the word 'Tokyo' depends on 'bought' with this dependency type.

In this output style, conjuncts are analyzed asymmetrically; however, the dependency between one of the conjuncts and a conjunction is collapsed, as shown in the figure below.

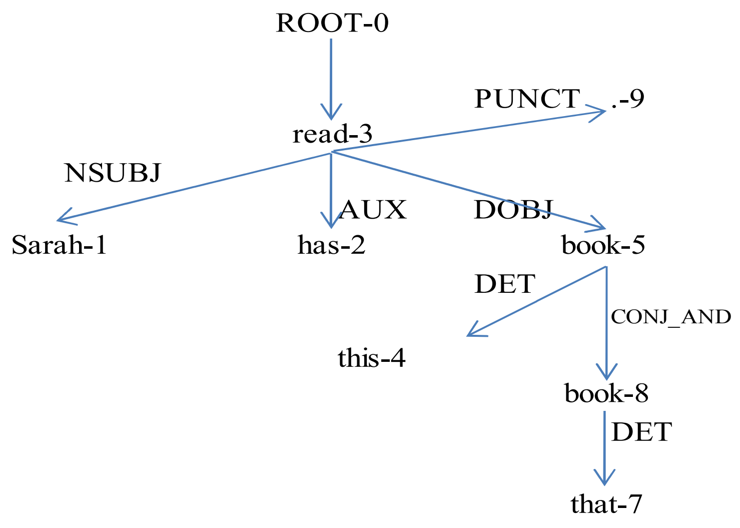


Figure 5.247. The typed-dependency tree for ‘Sarah has read this book and that book.’ (in the output style “Collapsed dependencies”)

The following multi-word conjunct-relation expressions are collapsed to one dependency type “conj_and”: ‘as well as,’ ‘not to mention,’ ‘but also,’ and ‘&.’ The following multi-word negative conjunct-relation expressions are collapsed to one dependency type “conj_negcc”: ‘but not,’ ‘instead of,’ ‘rather than,’ and ‘but rather.’

5.4.3 Collapsed dependencies with propagation of conjunct dependencies

This output type is an extension of the collapsed dependencies discussed in the previous section. Similar to collapsed dependencies, the first two conjuncts in this output style are analyzed both symmetrically and asymmetrically. In the example below, the first and second conjunct both depend on the word ‘read.’

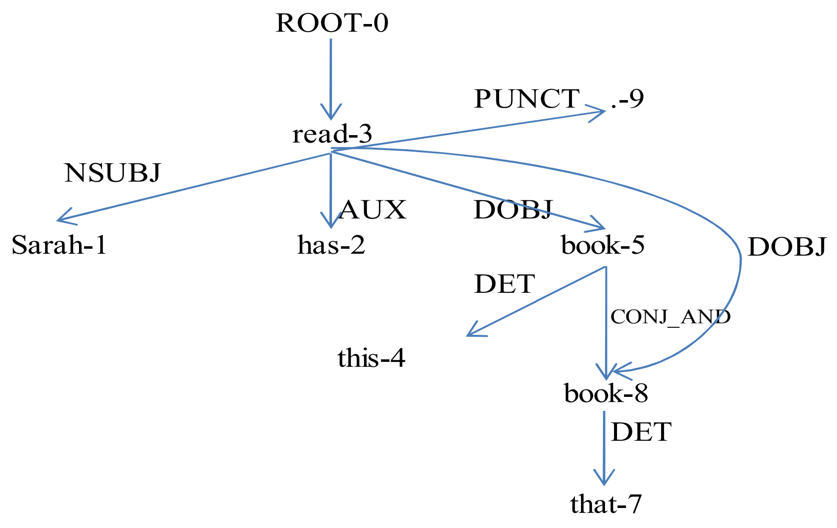


Figure 5.248. The dependency tree for ‘Sarah has read this book and that book.’ (in the output style “Collapsed dependencies with propagation of conjunct dependencies”)

5.4.4 Collapsed tree

This output style preserves the tree structure of dependencies because it ignores the cyclical dependency between the head of a relative clause and the word on which the relative clause depends. On the other hand, prepositions and conjuncts are collapsed to show the direct dependency between content words, as shown in the figures below.

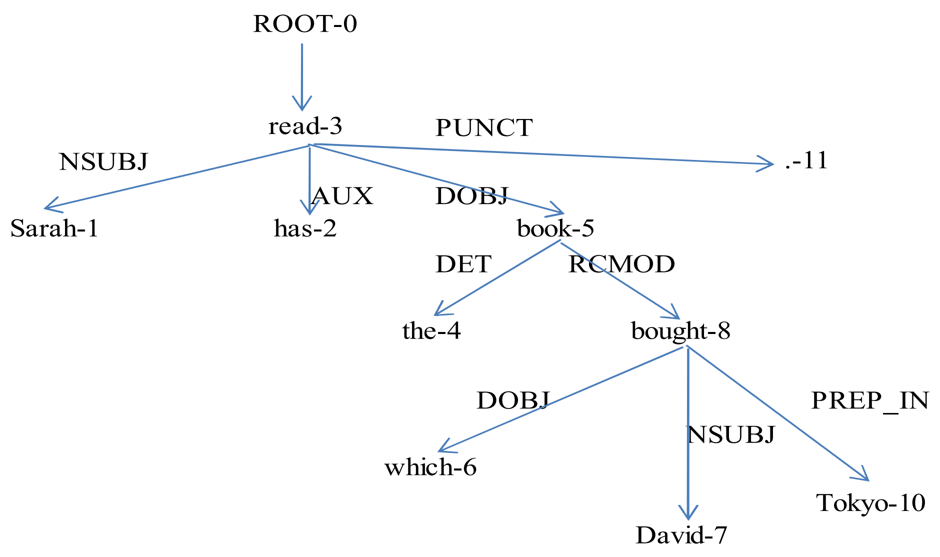


Figure 5.249. The typed-dependency tree for ‘Sarah has read the book which David bought in Tokyo.’ (In the output style “Collapsed tree”)

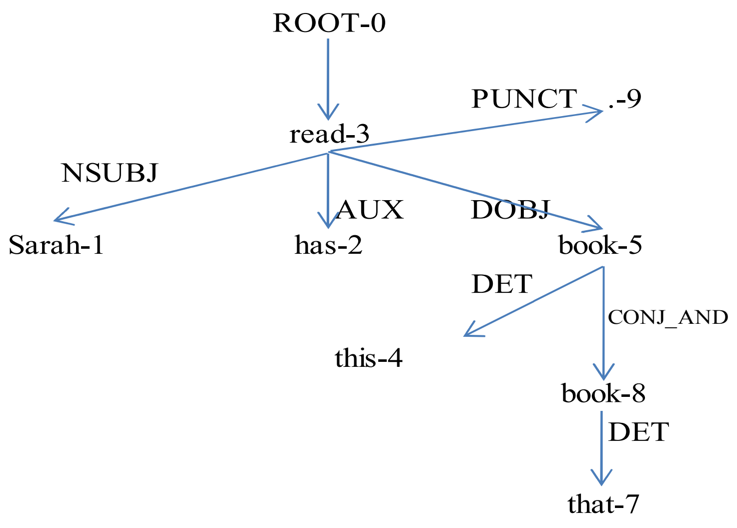


Figure 5.250. The typed-dependency tree for ‘Sarah has read this book and that book.’ (In the output style “Collapsed tree”).

5.4.5 Non-collapsed dependencies

This output style is very similar to the style called basic dependencies, with the addition of the dependency types "ref" and "rel." The dependency type "ref" is relevant when the head is a noun and the dependent is a relative pronoun that refers to the noun (see Section 5.3.4). In the example below, note that the dependency between the head of the relative clause ('bought') and the relative pronoun ('which') is typed as "dobj" and "rel" at the same time.

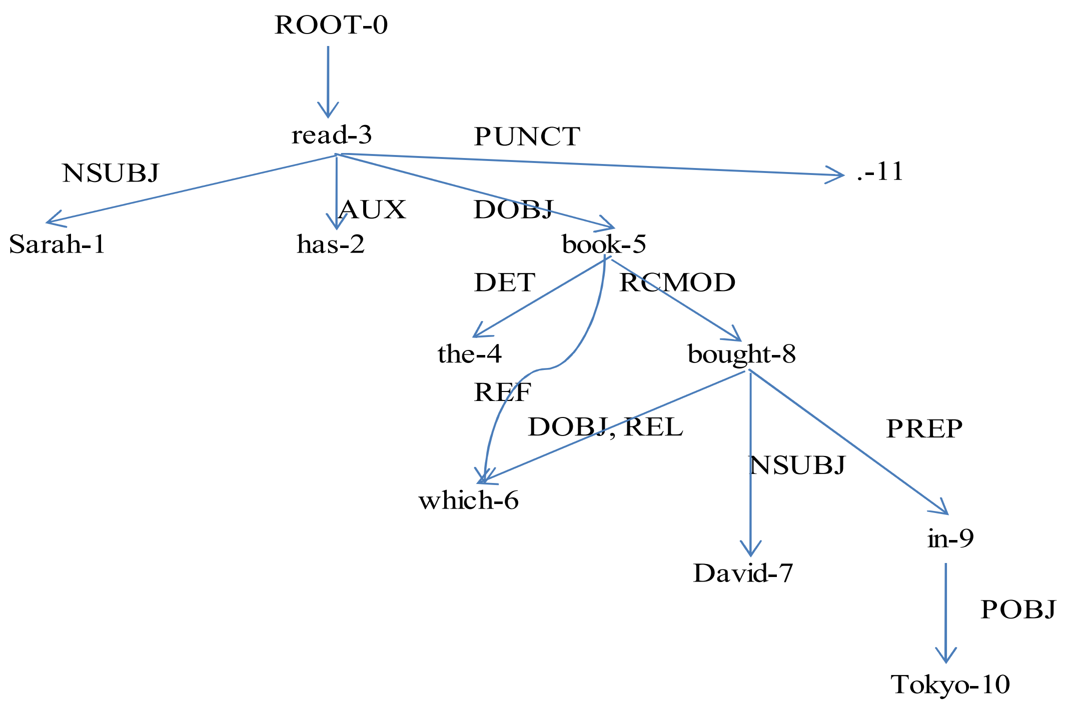


Figure 5.251. The dependency tree for 'Sarah has read the book which David bought in Tokyo.' (In the output style "Non-collapsed tree").

5.5 Summary

This chapter introduced the Stanford Parser (de Marneffe & Manning 2012), which is a state-of-the-art parser used in this study for acquiring typed-dependency tree representations for

English sentences. Section 5.2 described the output format of the Stanford Parser. Section 5.3 provided the definition of each dependency type used in the parsed output of the Stanford Parser, with reference to the criteria for surface syntactic relations by Mel'čuk (2009, 2011), along with example sentences for each of the dependency types, their typed dependency trees, and the functional structure representations equivalent to these trees. By doing this, each of the dependency types in Stanford Dependencies is given a theoretical backbone based on Mel'čuk's Criteria, and the parse output of Stanford Parser is shown to be equivalent to functional-structure representation in the framework of LFG. Section 5.4 explained the differences among the different output styles of the Stanford Parser.

6. Dependency Parsing of Japanese Sentences by KNP

6.1 Introduction

The previous chapter discussed the details of the Stanford Parser for the syntactic analysis of English sentences. This chapter introduces another parser for the Japanese language, called KNP (Kurohashi & Nagao 1992, 1994, 1998; Kawahara & Kurohashi 2007), and also introduces the dependency-type annotation to KNP output, along with definitions for each dependency type.

KNP is a rule-based dependency parser used for generating automatic dependency tree representations for Japanese sentences. The accuracy of this parser was improved during the time in which it was used for the development of Kyoto University Text Corpus ver. 4, a parsed corpus of Japanese (Kurohashi & Nagao 1998).

Because the parsed output of KNP does not contain the type of each dependency, it is necessary to annotate them to the parsed output; this will allow us to obtain cross-linguistic typed-dependency tree representations of Japanese based on the KNP output. As mentioned in Section 2.4.4, by defining each of the dependency types with respect to Mel'čuk's criteria for surface syntactic relations (SSyntRels in Mel'čuk's terms), these types can be established in the tradition of dependency grammar which was started by Tesnière and developed by Mel'čuk.

The structure of this chapter is as follows. Section 6.2 briefly introduces KNP and its output format. Section 6.3 describes the process through which KNP parsed output is annotated with dependency types. Section 6.4 deals with zero pronouns in elliptic sentences often used in Japanese. In section 6.5, each dependency type used with this parser in this study is defined with reference to the criteria for surface syntactic relations by Mel'čuk (2009, 2011). Similar to the dependency types of English in Chapter 5, these Japanese dependency types are provided with a traditional backbone started by Tesnière and developed by Mel'čuk.

6.2 The Output Format of KNP

This section introduces the output format of KNP. KNP starts with the output of JUMAN, a morphological analyzer for Japanese (the grammatical terms and analyses used in JUMAN are drawn from Masuoka & Takubo 1992). KNP builds on the output produced by JUMAN and adds information about the dependency relationships among the *syntactic units* (or *bunsetsu* in Japanese, which means 'the joints in a sentence') in the input sentence. In Japanese, syntactic units are the basic unit of syntactic dependency within a sentence (Hashimoto 1948). Each syntactic unit has a head, and the type of (or absence of) particle following the head determines the unit's grammatical function in the sentence (Nomura & Koike, 1992). For example, the sentence (6.1) 'Watashino aniga kono honwo yonda (My brother read this book)' has five syntactic units.

(6.1)

Watashi-no	ani-ga	kono	hon-wo	yon-da[35]
I-postp	*elder.brother-postp*	*this*	*book-postp*	*read-past*

'My brother read this book.'

[35] In this study, Japanese sentences are presented in the following format. In the first line, one sentence is presented with one space between syntactic units, and with a hyphen between a word and a postposition or a morpheme. In the second line, the gloss for each word, postposition or morpheme is indicated in italics. The last line provides the English translation of the Japanese sentence. The sentence-ending period in Japanese '。' and the phrase-segmenting comma '、' are not used in this dissertation.

(6.2) is the KNP parsed output for this sentence, and (6.3) is a simplified English translation of this KNP output, which contains information relevant to the topic here.

(6.2)

```
# S-ID:1 KNP:4.0-CF1.1 DATE:2013/01/01 SCORE:-8.15215
* 1D <体言><係:ノ格>
+ 1D <体言><係:ノ格>
私 わたし 私 名詞 6 普通名詞 1 * 0 * 0 “代表表記:私/わたし 漢字読み:訓 カテゴリ:人”
の の の 助詞 9 接続助詞 3 * 0 * 0 NIL
* 4D <体言><係:未格>
+ 4D <体言><係:未格>
兄 あに 兄 名詞 6 普通名詞 1 * 0 * 0 “代表表記:兄/あに 漢字読み:訓 カテゴリ:人 ドメイン:家庭・暮らし”
は は は 助詞 9 副助詞 2 * 0 * 0 NIL
* 3D <係:連体>
+ 3D <係:連体>
この この この 指示詞 7 連体詞形態指示詞 2 * 0 * 0 “疑似代表表記 代表表記:この/この”
* 4D <体言><係:ヲ格>
+ 4D <体言><係:ヲ格>
本 ほん 本 名詞 6 普通名詞 1 * 0 * 0 “代表表記:本/ほん 漢字読み:音 カテゴリ:人工物-その他;抽象物”
を を を 助詞 9 格助詞 1 * 0 * 0 NIL
* -1D <用言:動><レベル:C><ID:（文末）><係:文末>
+ -1D <用言:動><レベル:C><ID:（文末）><係:文末><格解析結果:読む/よむ:動 2:ガ/N/兄/1/0/1;ヲ/C/本/3/0/1;ニ/U/-/-/-;ト/U/-/-/-;デ/U/-/-/-;カラ/U/-/-/-;ヨリ/U/-/-/-;マデ/U/-/-/-;ヘ/U/-/-/-;時間/U/-/-/-;外の関係/U/-/-/-;ノ/U/-/-/-;修飾/U/-/-/-;トスル/U/-/-/-;ニツク/U/-/-/-;ニツヅク/U/-/-/-;トイウ/U/-/-/-;ニアワセル/U/-/-/-;ニヨル/U/-/-/-;ニトル/U/-/-/->
```

```
読んだ よんだ 読む 動詞 2 * 0 子音動詞マ行 9 タ形 10 “代表表記:読む/よむ”
。 。 。 特殊 1 句点 1 * 0 * 0 NIL
EOS
```

(6.3)

```
* 1D
watashi    noun
no         particle
* 2D
ani        noun
wa         particle
* 3D
kono       determiner
* 4D
hon        book
wo         particle
* -1D
yonda      verb
```

The first line ‘* 1D’ indicates that the syntactic unit ‘watashino’ depends on the first syntactic unit ‘aniwa’ (KNP starts counting the number of syntactic units with zero; therefore, the syntactic unit ‘watashino’ is the 0th syntactic unit of this sentence). The second line ‘watashi noun’ and the third line ‘no particle’ indicates that the 0th syntactic unit consists of two morphemes where the first morpheme ‘watashi’ is a noun and the second morpheme ‘wa’ is a particle. Thus, the KNP output indicates: (1) the syntactic units in the sentence; (2) the morphemes in each syntactic unit; and (3) the dependency relationships among these syntactic units.

6.3 Dependency-Type Annotation to KNP Output

This section introduces the process to annotate dependency types for each dependency relationship among the syntactic units in the output of KNP. Unlike the output of the Stanford Parser, the output of KNP does not specify the grammatical relations among syntactic units (such as “nsubj” or “rcmod”). In order to have cross-linguistic functional-structure representation for the Japanese language, Oya (2010a) introduced an automatic grammatical-relation annotation method for the output of KNP. This method employs the part-of-speech information for each morpheme in a syntactic unit so that all syntactic units are divided into the following four categories (Oya 2010a, p.212):

(6.4)

Particled inflective units: units which have at least one particle, and which have an inflecting element (verb, adjective, verbal suffix or copula) as their head

Particled non-inflective units: units which have at least one particle, and which do not have an inflecting element as their head.

Non-particled, inflective units: units which have no particle, and which have an inflecting unit as their head.

Non-particled, non-inflective units: units which have no particle, and which do not have an inflecting element as their head.

The syntactic units in each category are further divided into subcategories according to the particle or to their inflection form. One dependency type is assigned for each of the subcategories. For example, the syntactic unit ‘watashino’ in the sentence (6.1) is a particled non-inflective unit. The particle is ‘no,’ which functions to constitute a postpositional unit. Therefore, it is assigned the dependency type ‘postp_no.’ It depends on the syntactic unit ‘aniwa.’ The dependency relationship between ‘watashino’ and ‘aniwa’ is represented in the Stanford-Parser style triple shown below in (6.5).[36]

(6.5)

```
postp_no(aniwa-2, watashino-1)
```

(6.6) below is the Stanford-Parser style triples for the sentence (6.1) as a whole.

(6.6)

```
postp_no(aniwa-2, watashino-1)
topic(yonda-5, aniwa-2)
det(honwo-4, kono-3)
postp_wo(yonda-5, honwo-4)
root(root-0, yonda-5)
```

Dependency-type annotation for KNP output is automatically processed by an original Ruby script (see Appendix V).

[36] The numbering for syntactic unit in this study starts with zero, like Oya(2010a), but this study sets the root unit as 0th syntactic unit.

6.4 Treatment of Zero Pronouns[37]

This section deals with the treatment of zero pronouns in Japanese. Japanese uses elliptic[38] sentences quite often. Both in spoken and in written Japanese, it is common to find sentences whose main predicate lacks nouns with its core grammatical functions such as subject or object (Kanatani 2002; Mikami 1972; Toyama 1973, among others). In addition to this, the main predicate of the sentence does not have morphological means to indicate the person and the number of the subject, unlike head-marking languages such as Latin or Russian. In terms of functional well-formedness in the framework of LFG (see Section 3.2.2), these elliptic sentences are problematic because their functional structures seem to violate the completeness constraint. Oya (2010a) argued that the completeness constraint is observed in such elliptic sentences because they contain *zero pronouns* which are the values of the core grammatical-function attributes. In this section, we look at examples of sentences containing zero pronouns which belong to different semantic types, and explain how the completeness constraint is observed in the functional structures for these elliptic sentences.[39]

6.4.1 Examples of "elliptic" sentences

It is often the case that Japanese sentences do not have overt subjects or overt objects. For

[37] This section is based on Oya (2013d) and Oya (2014).

[38] Actually, it will be argued later in Section 6.4.2 that they are not "elliptic" sentences, because they lack nothing. The term "ellipsis" and "elliptic sentences" are used for a while in this section, in order to avoid misunderstandings.

[39] Oya (2010a) treated zero pronouns as nodes in the typed-dependency tree for an elliptic sentence. This study, however, does not do so, because it assumes that what is not syntactic should not enter into syntactic representations. Zero pronouns are concerned with verbal semantics that is independent from syntax, and their referents are specified contextually or conventionally.

example, consider the following dialogue (6.7).

(6.7)

Sarah: David, have you read this book?

David: Yes, I have.

Sarah: Is this interesting?

David: Yes, it is.

The Japanese translation of this dialogue is shown in (6.8). Each sentence in (6.8) is followed by an English gloss in parentheses.

(6.8)

Sarah: David, kono hon yonda?

(Sarah: David, this book read-past)

David: Un, yondayo.

(David: Yes read-past-ending)

Sarah: Omoshiroi?

(Sarah: interesting)

David: Un, omoshiroiyo.

(David: Yes interesting-ending)

There is no overt subject in all the sentences in the Japanese translation in (6.8). The absence of overt subjects does not pose any problem to native speakers of Japanese, because it is obvious for them from the context who read what. In this sense, native speakers of Japanese interpret sentences with more reference to the context than native speakers of English, which requires the presence of pronouns.[40]

The absence of overt subjects is an unmarked phenomenon in Japanese. Oya (2010a) reported that about 96% of the subjects of all the verbs in 500 sentences which are randomly selected from Kyoto University Text Corpus (Kurohashi & Nagao 1998) are not overt, or expressed as a topic (a noun with the topic marker '–wa'). See Section 6.4.4 for the reason why a noun with '-wa' depending on a verb is not the subject of the verb.

We also have to notice that the noun 'hon (book)' in the first utterance by Sarah lacks the postposition which is often considered as the object marker, yet it is natural to native speakers' ears. If the postposition '-wo' is the only means to indicate the grammatical function OBJ, it is unclear how to interpret the noun 'hon' in the sentence as the object of the verb 'yonda (read in the past tense).' In this sense, it is safe to argue that postpositions in Japanese are not the only means to indicate the grammatical functions of the nouns in a sentence.

6.4.2 Functional-structure representation of elliptic Japanese sentences

As indicated in the previous section, these elliptic sentences in Japanese seem to violate the completeness constraint, which states that a predicate and all its arguments must be present in a functional structure (Kaplan & Bresnan 1982, pp.211-212). For example, the subject and the object are absent from the sentence 'yondayo (I have read it)' in David's first utterance in the dialogue (6.6). The functional structure for this sentence would be like Figure 6.1 below. This functional structure violates the completeness constraint, because the value of the SUBJ

[40] Oya (2013d) points out that the frequent use of zero pronouns in Japanese language seems to reflect the *high-context culture* (Hall 1976) of Japanese people, while the frequent use of pronouns in English language seems to reflect the *low-context culture* of English-speaking people.

attribute and the value of the OBJ attribute are both missing; they are required by the verb ‘yondayo.’

```
| PRED    'yonda<SUBJ, OBJ>'  |
| SUBJ    [            ]      |
| OBJ     [            ]      |
| TENSE   PAST                |
```

Figure 6.1. The functional structure for ‘yondayo.’ (Oya 2013d, Oya 2014)

However, the fact that this sentence is natural to native speakers of Japanese suggests that the functional structure for an elliptic sentence such as ‘yondayo’ is actually complete and there are no missing elements; hence, we should not use the term ‘ellipsis’ for sentences like ‘yondayo.’

Oya (2010a) argued that the presence of *zero pronouns* fills the gap. The lexical information for the verb ‘yondayo’ is as follows. This lexical entry contains the equations for the subject zero pronouns and for the object zero pronouns.

(6.8)

(↑PRED)= ‘yonda<SUBJ,OBJ>’

(↑SUBJ PRED)= ‘PRO’

(↑SUBJ FORM)=ZERO

(↑OBJ PRED)= ‘PRO’

(↑OBJ FORM)=ZERO

(↑TENSE)=PAST

(↑ENDING)= ‘yo’[41]

The claim that verbal predicates contain zero pronouns for core arguments (subjects, direct objects and indirect objects) has been made in Bresnan (1982, p.385) and Bresnan (2001, p.295), and other works cited there. For example, Bresnan (2001, p.295) states that “null pronominals are provided as a lexical default to core arguments of verbal argument structures.” Bresnan’s (2001) term “null pronominals” is equivalent to zero pronouns.

This study is different from them in a number of ways. First, this study claims that Japanese core arguments are zero pronouns by default; they are better considered to be not only “lexical default,” but also discourse default. This claim is supported by the observation in Oya (2010a) that about 96% of subjects of the verbs in 500 sentences randomly chosen from a text corpus are zero pronouns, as mentioned in the previous section. Second, this study also claims that the postpositions on overt noun phrases such as ‘-ga,’ ‘-wo,’ or ‘-ni,’ which are considered to be so-called case markers for core arguments, actually do not function as case markers, because they have functions other than expressing subjects, direct objects, or indirect objects, and because they can be absent, especially in speaking (e.g., ‘Boku kono hon yonda’ I have read this book).

The typed-dependency tree for ‘yondayo.’ is shown below.

[41] The ‘-yo’ is a *sentence-final particle* which indicates that the speaker asserts the meaning of the sentence to the listener (Ono & Nakagawa 1997).

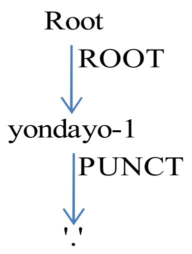


Figure 6.2. The typed-dependency tree for ‘yondayo.’ (Oya 2013d, Oya 2014)

The typed-dependency tree above is equivalent to the functional structure below. Notice that this functional structure contains the subject zero pronoun and the object zero pronoun; therefore, this functional structure observes the completeness constraint.

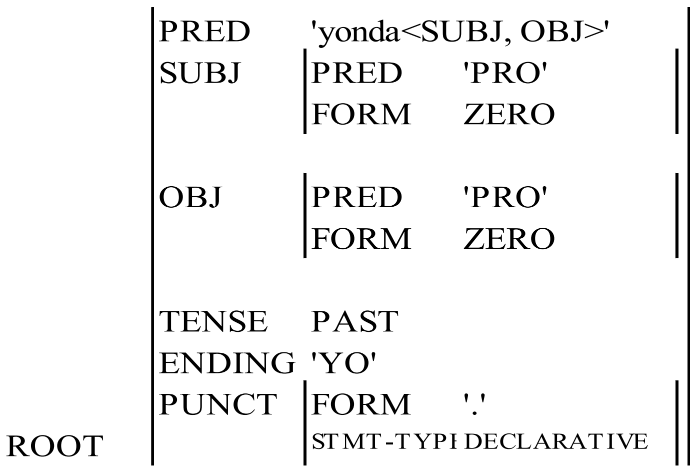


Figure 6.3. The functional structure for ‘yondayo’ with zero pronouns. (Oya 2013d, Oya 2014)

The zero pronouns are pronouns without phonological features. The antecedent of a given zero pronoun is determined by the context, inter-clausally or intra-clausally. In the sentence ‘yondayo’ in the dialogue above, the antecedent of the zero pronouns are determined inter-clausally. In addition to this, a verb with no core argument overtly depending on it can function as a one-word sentence, as far as the antecedents of its zero pronouns are determined inter-clausally.

If, on the other hand, overt core arguments depend on a verb, they function as the intra-clausal antecedents of the zero pronouns of the verb. They also have the function of making explicit the meaning of the zero pronouns. Instances of intra-clausal antecedents of zero pronouns will be illustrated in next sections.

6.4.3 Semantic types of Japanese zero pronouns

This section deals with the different semantic types of Japanese zero pronouns. Their semantic types are based on Tomioka (2003, p.324), which claims that Japanese zero pronouns (*null pronouns* in his terms) have the same semantic functions that the English overt pronouns have. (6.9) below is the list of the semantic functions:

(6.9)(= (9) in Tomioka 2003, p.324)

a. Referential

b. Bound variable

c. Unselectively bound variable

d. Pronouns with pronoun-containing antecedents

e. Indefinite pronouns

f. Property anaphora

6.4.3.1 Referential zero pronouns

Example (6.10) contains referential zero pronouns (shown as PRO) which function as either as the subject or the direct object of the verb ‘sasotta (someone invited someone)’.

(6.10)

Sarah-wa	David-wo	sasot-ta.	John-mo	sasotta[42].
Sarah-topic	*David-postp*	*invite-past*	*John-focus*	*invite-past*

‘Sarah invited David. She also invited John.’ or ‘Sarah invited David. John also invited him.’

The second sentence has two interpretations: one is that Sarah also invited John; the other is that John also invited David. This ambiguity is due to the fact that a noun with the focus marker ‘-mo’ can express either the subject of the object of a verb, and the zero pronoun in the second sentence can refer to either the subject or the object of the first sentence. The reader/listener of this sentence must judge whom this zero pronoun refers to, based on the context where it is uttered.

The typed-dependency tree for the sentence above is shown below. Notice that there is no node for ‘PRO.’ This is because this ‘PRO’ is considered to be registered in the lexical entry of the verb ‘sasotta (someone invited someone),’ as indicated in the previous section. Notice that this representation does not yield these two different interpretations mentioned above.

[42] Oya (2014) included a PRO in each of the example sentences, yet it is not included in the example sentences in this section, in order to highlight the property of a PRO that it has no phonological feature.

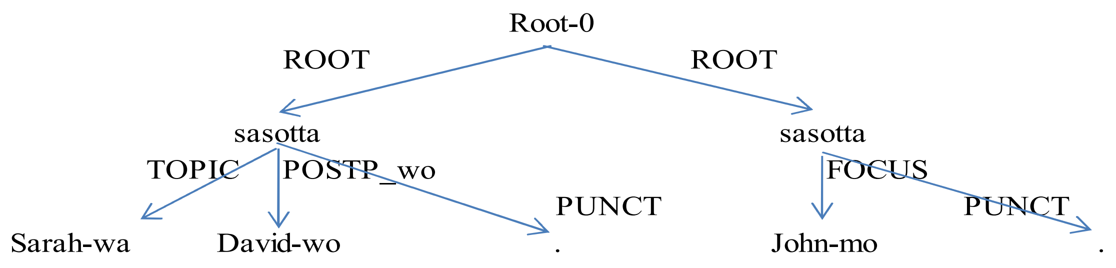


Figure 6.4. The typed-dependency tree for ‘Sarahwa Davidwo sasotta. Johnmo sasotta.’

The typed-dependency tree above is equivalent to the functional structure below, and it is this functional structure where these two interpretations above are differentiated. Notice which index is assigned to which elements in the functional structure.

$$\left[\begin{array}{ll}
\text{ROOT} & \left[\begin{array}{ll}
\text{PRED} & \text{'sasotta<SUBJ, OBJ>'}\\
\text{SUBJ} & \left[\begin{array}{ll}\text{PRED} & \text{'PRO}_i\text{'}\\ \text{FORM} & \text{ZERO}\end{array}\right]\\
\text{OBJ} & \left[\begin{array}{ll}\text{PRED} & \text{'PRO}_j\text{'}\\ \text{FORM} & \text{ZERO}\end{array}\right]\\
\text{TOPIC} & \left[\begin{array}{ll}\text{PRED} & \text{'Sarah}_i\text{'}\end{array}\right]\\
\text{POSTP\_wo} & \left[\begin{array}{ll}\text{PRED} & \text{'David}_j\text{'}\end{array}\right]\\
\text{TENSE} & \text{PAST}\\
\text{PUNCT} & \left[\begin{array}{ll}\text{FORM} & \text{'.'}\\ \text{STMT-TYPE} & \text{DECLARATIVE}\end{array}\right]
\end{array}\right]\\
\text{ROOT} & \left[\begin{array}{ll}
\text{PRED} & \text{'sasotta<SUBJ, OBJ>'}\\
\text{SUBJ} & \left[\begin{array}{ll}\text{PRED} & \text{'PRO}_i\text{'}\\ \text{FORM} & \text{ZERO}\end{array}\right]\\
\text{OBJ} & \left[\begin{array}{ll}\text{PRED} & \text{'PRO}_k\text{'}\\ \text{FORM} & \text{ZERO}\end{array}\right]\\
\text{FOCUS} & \left[\begin{array}{ll}\text{PRED} & \text{'John}_k\text{'}\end{array}\right]\\
\text{TENSE} & \text{PAST}\\
\text{PUNCT} & \left[\begin{array}{ll}\text{FORM} & \text{'.'}\\ \text{STMT-TYPE} & \text{DECLARATIVE}\end{array}\right]
\end{array}\right]
\end{array}\right]$$

Figure 6.5. The functional structure for ‘Sarah-wa David-wo sasotta. John-mo sasotta.’ (meaning: ‘Sarah invited David. She also invited John.’)

In the local functional structure for the first sentence, the index ‘i’ is assigned to the topic ‘Sarah’ and the subject zero pronoun of the verb ‘sasotta.’ In the same functional structure, the index ‘j’ is assigned to the postp_wo ‘David’ and the object zero pronoun of the same verb. This means that the subject zero pronoun of the verb ‘sasotta’ in the first sentence refers to ‘Sarah,’ and the object zero pronoun refers to ‘David.’ In the local functional structure for the second sentence, the index ‘i’ is assigned to the subject zero pronoun of the verb ‘sasotta’ and the index ‘k’ is assigned to the focus ‘John’ and the object zero pronoun of the same verb. This means that the subject zero pronoun of the verb ‘sasotta’ in the functional structure for the second sentence refers to ‘Sarah’ and the object zero pronoun of the same verb refers to ‘John’; therefore, the second sentence means that Sarah also invited John.

The following functional structure is also equivalent to the typed-dependency tree above, yet with a different interpretation. Notice that in the functional structure for the second sentence the index ‘k’ is assigned to the subject zero pronoun of the verb ‘sasotta’ and to the focus ‘John,’ and the index ‘j’ is assigned to the object zero pronoun of the ‘sasotta’ in the first sentence and to the object zero pronoun of the ‘sasotta’ in the first sentence. This means that the subject zero pronoun of the verb ‘sasotta’ in the functional structure for the second sentence refers to ‘John’ and the object zero pronoun of the same verb refers to ‘David’; therefore, the second sentence means that John also invited David.

```
      | PRED         'sasotta<SUBJ, OBJ>'                 |
      | SUBJ         | PRED     'PRO_i'           |       |
      |              | FORM     ZERO              |       |
      |                                                   |
      | OBJ          | PRED     'PRO_j'           |       |
      |              | FORM     ZERO              |       |
      |                                                   |
      | TOPIC        | PRED     'Sarah_i'         |       |
      |                                                   |
      | POSTP_wo     | PRED     'David_j'         |       |
      | TENSE          PAST                               |
      | PUNCT        | FORM     '.'               |       |
ROOT  |              | STMT-TYPE DECLARATIVE      |       |
      |                                                   |
      | PRED         'sasotta<SUBJ, OBJ>'                 |
      | SUBJ         | PRED     'PRO_k'           |       |
      |              | FORM     ZERO              |       |
      |                                                   |
      | OBJ          | PRED     'PRO_j'           |       |
      |              | FORM     ZERO              |       |
      |                                                   |
      | FOCUS        | PRED     'John_k'          |       |
      | TENSE          PAST                               |
      | PUNCT        | FORM     '.'               |       |
ROOT  |              | STMT-TYPE DECLARATIVE      |       |
```

Figure 6.6. The functional structure for 'Sarah-wa David-wo sasotta. John-mo sasotta.' (meaning: 'Sarah invited David. John also invited him.')

These examples above illustrate that the referential zero pronouns of a verb can refer something inter-clausally or intra-clausally, and that this flexibility causes ambiguous interpretations.

6.4.3.2 Zero pronouns as bound variables

Example (6.11) contains a zero pronoun as a bound variable which functions as the subject of the verb ‘okorareta (someone was scolded at by someone)’. The construction ‘dono ~mo’ means ‘every ~’ if the sentence is affirmative, or ‘no ~’ if the sentence is negative.

(6.11)

Dono	gakusei-mo	Sarah-ni	okorare-ta-to	it-ta.
Which	*student-focus*	*Sarah-postp*	*be.scolded.at-past-ccomp*	*say-past*

‘Every student said that he or she was scolded at by Sarah.’

The typed-dependency tree for the sentence above is shown below.

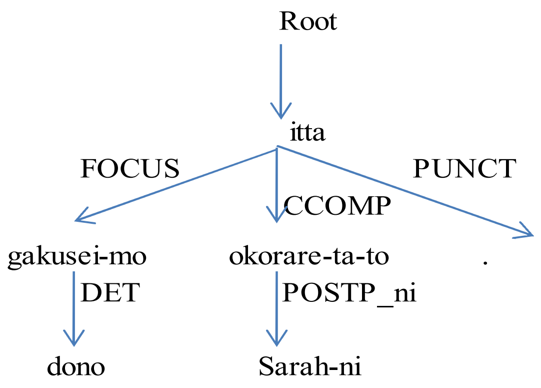


Figure 6.7. The typed-dependency tree for ‘Dono gakusei-mo Sarah-ni okorare-ta-to itta (Every student said that he or she was scolded at by Sarah).’ (Oya 2014)

The typed-dependency tree above is equivalent to the functional structure below. The index ‘i’ is assigned to the subject zero pronoun of ‘itta (someone said something),’ the subject zero pronoun of ‘okorareta (someone was scolded at by someone),’ and the local functional structure which is the value of the attribute ‘FOCUS.’ The word ‘gakusei (student)’ with the focus marker ‘-mo’ is modified by an interrogative determiner ‘dono (which),’ and this construction means ‘every student’ as mentioned before. Therefore, these zero pronouns with the index ‘i’ in the functional structure below refer to the every student.

```
|      |PRED     'itta<SUBJ, CCOMP>'                                   ||
|      |SUBJ    |PRED       'PROi'   |                                 ||
|      |        |FORM       ZERO     |                                 ||
|      |                                                               ||
|      |CCOMP   |PRED       'okorareta<SUBJ>'     |                    ||
|      |        |SUBJ      |PRED      'PROi' |    |                    ||
|      |        |          |TYPE      ZERO   |    |                    ||
|      |        |                               |                    ||
|      |        |POSTP_ni  |PRED      'Sarah'|    |                    ||
|      |        |TENSE      PAST                |                    ||
|      |        |VOICE      PASSIVE             |                    ||
|      |                                                               ||
|      |FOCUS   |PRED       'gakusei'                     |            ||
|      |        |DET       |FORM      'dono'          |   |            ||
|      |        |          |TYPE      INTERROGATIVE   |   |i           ||
|      |                                                               ||
|      |TENSE    PAST                                                  ||
|      |PUNCT   |FORM       '.'                 |                      ||
|ROOT  |        |STMT-TYPE  DECLARATIVE         |                      ||
```

Figure 6.8. The functional structure for 'Dono gakusei-mo Sarah-ni okorare-ta-to itta. (Every student said that he or she was scolded at by Sarah)' (Oya 2014)

6.4.3.3 Zero pronouns as unselectively bound variables

Zero pronouns function as unselectively bound variables when their antecedents are not referential, and they appear beyond the scope of their antecedents (Tomioka 2003, p. 322). Oya (2014) interprets this as follws: "zero pronouns of this type refer to a definite entity, while their antecedents refer to an indefinite entity." Example (6.12) contains a zero pronoun as an unselectively bound variable.

(6.12)

Sarah-wa	David-ni	atarashii	hon-wo	kat-ta-ga,
Sarah-topic	*David-postp*	*new*	*book-postp*	*buy-past-advcl,*
David-wa	suguni	nakushi-ta.		
David-topic	*immediately*	*lose-past*		

'Sarah had bought a new book for David, but he immediately lost it.'

The typed-dependency tree for the sentence above is shown in the figure below.

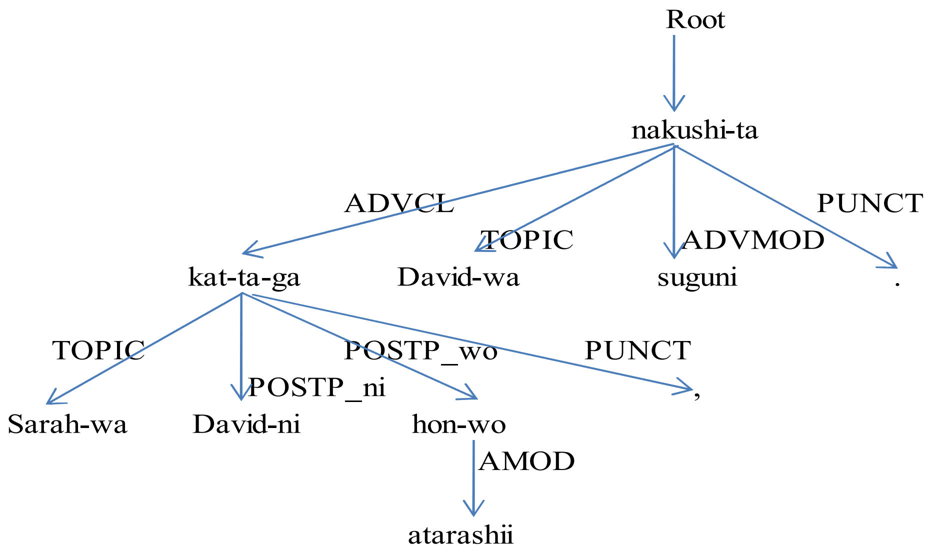


Figure 6.9. The typed-dependency tree for 'Sarah-wa David-ni atarashii hon-wo kat-ta-ga, David-wa suguni nakushita. (Sarah had bought a new book for David, but he immediately lost it.)' (Oya 2014)

The typed-dependency tree above is equivalent to the functional structure below[43]. Notice that

[43] Notice that the verb "katta" is analyzed as a transitive verb that takes the subject and the direct object, not as a ditransitive verb which takes the subject, the direct object and the indirect object. It is not an essential syntactic

the index ‘i’ is assigned to ‘Sarah’ and the subject zero pronoun of the verb ‘kattaga (someone bought something, but …).’ In addition, the index ‘j’ is assigned to the object zero pronoun of the verb ‘nakushita,’ the object zero pronoun of the verb ‘kattaga,’ and the local functional structure that is the value of the attribute POSTP_wo. Finally, the index ‘k’ is assigned to ‘David’ and the subject zero pronoun of the verb ‘nakushita (someone lost something).’

---

issue whether the “David-ni” is an argument that is subcategorized for by the verb “katta”, because the (di-)transitivity of a given verb is an issue of language use, or a matter of convention. I strongly believe that we should do away with any attempt to treat something not syntactic as syntactic.

```
|    |PRED      'nakushita<SUBJ, DOBJ>'                                    ||
|    |SUBJ     |PRED       'PRO_k'     |                                   ||
|    |         |FORM       ZERO        |                                   ||
|    |                                                                     ||
|    |DOBJ     |PRED       'PRO_j'     |                                   ||
|    |         |FORM       ZERO        |                                   ||
|    |                                                                     ||
|    |ADVCL    |PRED       'katta<SUBJ,OBJ>'                        |      ||
|    |         |SUBJ       |PRED       'PRO_i'  |                   |      ||
|    |         |           |TYPE       ZERO     |                   |      ||
|    |         |                                                    |      ||
|    |         |OBJ        |PRED       'PRO_j'  |                   |      ||
|    |         |           |TYPE       ZERO     |                   |      ||
|    |         |                                                    |      ||
|    |         |TOPIC      |PRED       'Sarah_i' |                  |      ||
|    |         |                                                    |      ||
|    |         |POSTP_ni   |PRED       'David_k' |                  |      ||
|    |         |                                                    |      ||
|    |         |POSTP_wo   |PRED       'hon'                    |   |      ||
|    |         |           |AMOD       |PRED    'atarashii' |   |_j |      ||
|    |         |                                                    |      ||
|    |         |TENSE      PAST                                     |      ||
|    |TOPIC    |PRED       'David_k'   |                                   ||
|    |                                                                     ||
|    |ADVMOD   |PRED       'suguni'    |                                   ||
|    |                                                                     ||
|    |TENSE     PAST                                                       ||
|    |                                                                     ||
|    |PUNCT    |FORM       '.'         |                                   ||
|ROOT|         |STMT-TYPE  DECLARATIVE |                                   ||
```

Figure 6.10. The functional structure for ‘Sarah-wa David-ni atarashii hon-wo kat-ta-ga, David-wa suguni nakushita. (Sarah had bought a new book for David, but he immediately lost it.)’ (Oya 2014)

6.4.3.4 Zero pronouns with a pronominal-containing antecedent.

Example (6.13) contains a zero pronoun with a pronominal-containing antecedent.

(6.13)

Sarah-wa	jibun-no	ie-wo	ut-ta.	David-mo	ut-ta.
Sarah-topic	self-postp	house-postp	sell-past	David-focus	sell-past

‘Sarah sold her own house. David did so, too.’

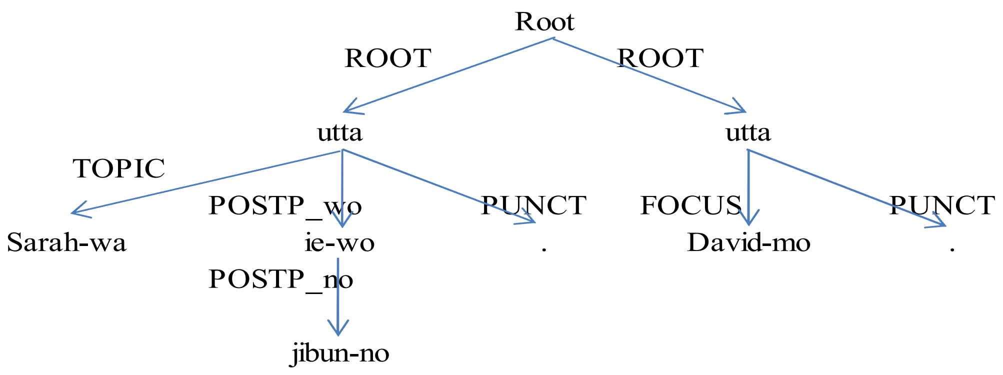
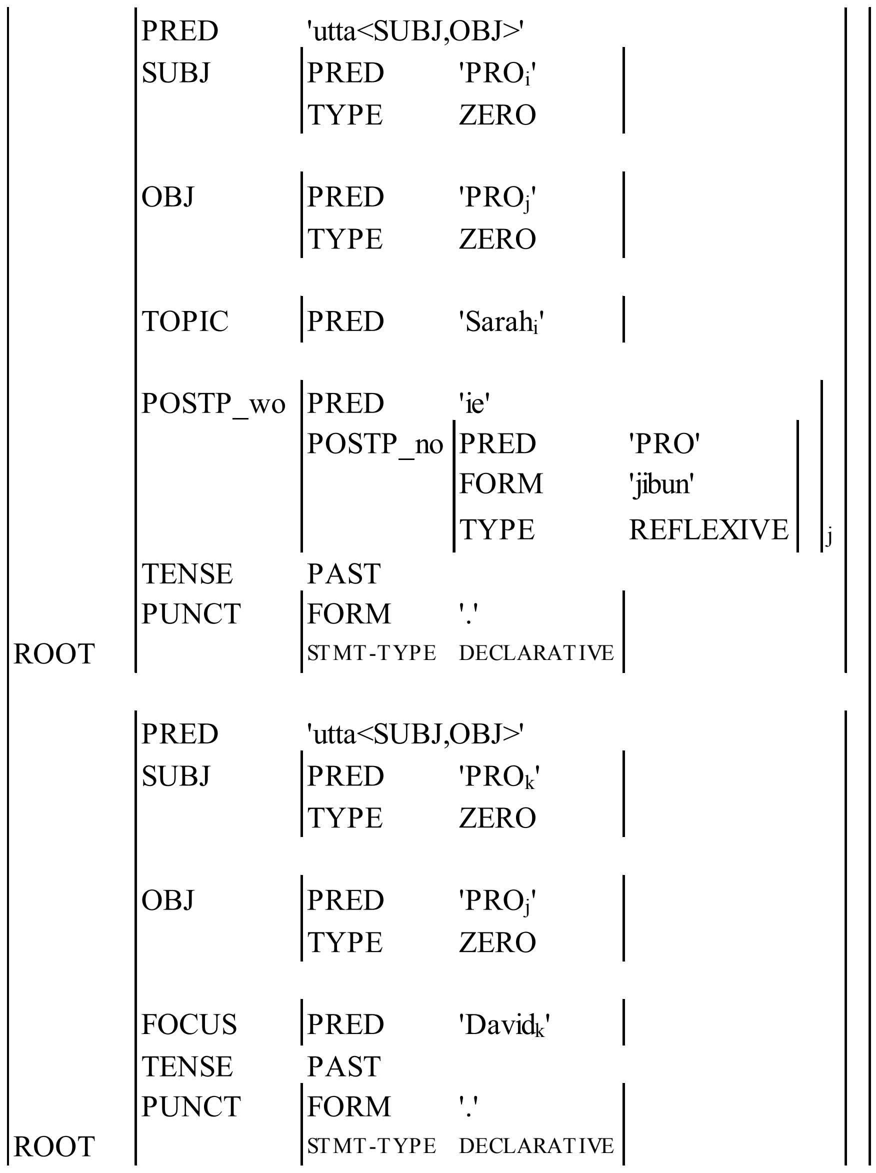


Figure 6.11. The typed-dependency tree for ‘Sarawa jibuno iewo utta. Davidmo utta (Sarah sold her own house. David did so, too).’

The typed-dependency tree above is equivalent to the functional structure below. Notice that the index ‘i’ is assigned to the subject zero pronoun of the verb ‘utta (someone sold something)’ in the first sentence, and also to the topic ‘Sarah’ in the first sentence. The subject zero pronoun in the first sentence refers to ‘Sarah’ intra-clausally.

On the other hand, the index ‘j’ is assigned to the local functional structure which is the value of the attribute POSTP_wo in the first sentence, the object zero pronoun of the verb ‘utta’ in the first sentence, and the object zero pronoun of the verb ‘utta’ in the second sentence. The object zero pronoun in the second sentence refers to the local functional structure inter-clausally. Notice that this object zero pronoun does not refer to ‘ie (house).’ If it does, then it must be interpreted as referring to an indefinite house.

Lastly, the index ‘k’ is assigned to the subject zero pronoun of the verb ‘utta’ in the second sentence, and also to the focus ‘David’ in the second sentence. The subject zero pronoun refers to ‘David’ intra-clausally.



ROOT
PRED 'utta<SUBJ,OBJ>'
SUBJ PRED 'PRO_i' TYPE ZERO
OBJ PRED 'PRO_j' TYPE ZERO
TOPIC PRED 'Sarah_i'
POSTP_wo PRED 'ie'
POSTP_no PRED 'PRO' FORM 'jibun' TYPE REFLEXIVE
j
TENSE PAST
PUNCT FORM '.' STMT-TYPE DECLARATIVE
ROOT
PRED 'utta<SUBJ,OBJ>'
SUBJ PRED 'PRO_k' TYPE ZERO
OBJ PRED 'PRO_j' TYPE ZERO
FOCUS PRED 'David_k'
TENSE PAST
PUNCT FORM '.' STMT-TYPE DECLARATIVE

Figure 6.12. The functional structure for ‘Sarawa jibuno iewo utta. Davidmo utta.’

Oya (2014) argued that the reflexive pronoun ‘jibun’ cannot be assigned a unique index. In the example functional structure above, ‘jibun’ refers to ‘Sarah’ in the first clause; hence, ‘jibun’ should be indexed “i”. On the other hand, ‘jibun’ is also contained in the local functional structure in the second clause, and this local functional structure is referred to by the object zero pronoun in the second clause. Then, ‘jibun’ comes to refer to the subject zero pronoun, which refers to ‘David’; hence, ‘jibun’ should be indexed “k.” For the binding of reflexive pronouns with respect to the notion of f-structure nucleus, see Bresnan (2001, p.215).

6.4.3.5 Zero pronouns as indefinite pronouns

Example (6.14) contains a zero pronoun as an indefinite pronoun.

(6.14)

Sarah-wa	kuruma-wo	ut-ta.	David-mo	ut-ta.
Sarah-topic	*car-postp*	*sell-past*	*David-focus*	*sell-past*

‘Sarah sold a car. David also sold one.’

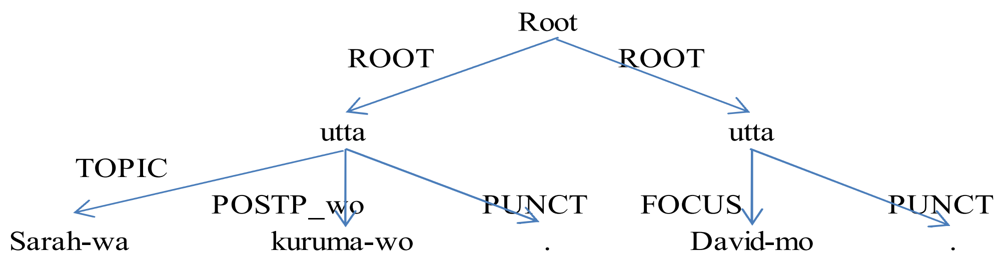


Figure 6.13. The typed-dependency tree for ‘Sarawa kurumawo utta. Davidmo utta.’

The typed-dependency tree above is equivalent to the functional structure below. Notice that no index is assigned to the object zero pronoun of the verb ‘utta’ in the second sentence. This is because the object zero pronoun of the first sentence refers to an entity (a car in the example above) which is different from what the object zero pronoun of the second sentence refers to; Sarah sold a car which is not the same car that David sold. In this case, the object zero pronoun in the second sentence functions as an indefinite pronoun referring to an entity whose identity can be interpreted through the context.

$$\left|\begin{array}{ll}
\text{ROOT} & \left|\begin{array}{ll}
\text{PRED} & \text{'utta<SUBJ,OBJ>'} \\
\text{SUBJ} & \left|\begin{array}{ll} \text{PRED} & \text{'PRO}_i\text{'} \\ \text{TYPE} & \text{ZERO} \end{array}\right| \\
\text{OBJ} & \left|\begin{array}{ll} \text{PRED} & \text{'PRO}_j\text{'} \\ \text{TYPE} & \text{ZERO} \end{array}\right| \\
\text{TOPIC} & \left|\begin{array}{ll} \text{PRED} & \text{'Sarah}_i\text{'} \end{array}\right| \\
\text{POSTP\_wo} & \left|\begin{array}{ll} \text{PRED} & \text{'kuruma}_j\text{'} \end{array}\right| \\
\text{TENSE} & \text{PAST} \\
\text{PUNCT} & \left|\begin{array}{ll} \text{FORM} & \text{'.'} \\ \text{STMT-TYPE} & \text{DECLARATIVE} \end{array}\right|
\end{array}\right| \\
\text{ROOT} & \left|\begin{array}{ll}
\text{PRED} & \text{'utta<SUBJ,OBJ>'} \\
\text{SUBJ} & \left|\begin{array}{ll} \text{PRED} & \text{'PRO}_k\text{'} \\ \text{TYPE} & \text{ZERO} \end{array}\right| \\
\text{OBJ} & \left|\begin{array}{ll} \text{PRED} & \text{'PRO'} \\ \text{TYPE} & \text{ZERO} \end{array}\right| \\
\text{FOCUS} & \left|\begin{array}{ll} \text{PRED} & \text{'David}_k\text{'} \end{array}\right| \\
\text{TENSE} & \text{PAST} \\
\text{PUNCT} & \left|\begin{array}{ll} \text{FORM} & \text{'.'} \\ \text{STMT-TYPE} & \text{DECLARATIVE} \end{array}\right|
\end{array}\right|
\end{array}\right|$$

Figure 6.14. The functional structure for ‘Sarawa kurumawo utta. Davidmo utta.’

6.4.3.6 Zero pronouns as property anaphora

Example (6.15) contains a zero pronoun as a property anaphor. Property anaphors are such that are modified by a numerical classifier. The construction '~shika V-nai' (V stands for a verb) means 'do only ~.' The expression 'issatsu-shika yoma-nai' means 'someone reads only one book.'

(6.15)

Sarah-wa	shuu-ni	san-satsu	hon-wo	yomu-ga,
Sarah-topic	*week-postp*	*three-books*	*book-postp*	*read-advcl,*
David-wa	issatsu-shika	yoma-nai.		
David-topic	*one.book-focus*	*read-neg*		

'Sarah reads three books a week, but David reads only one (a week).'

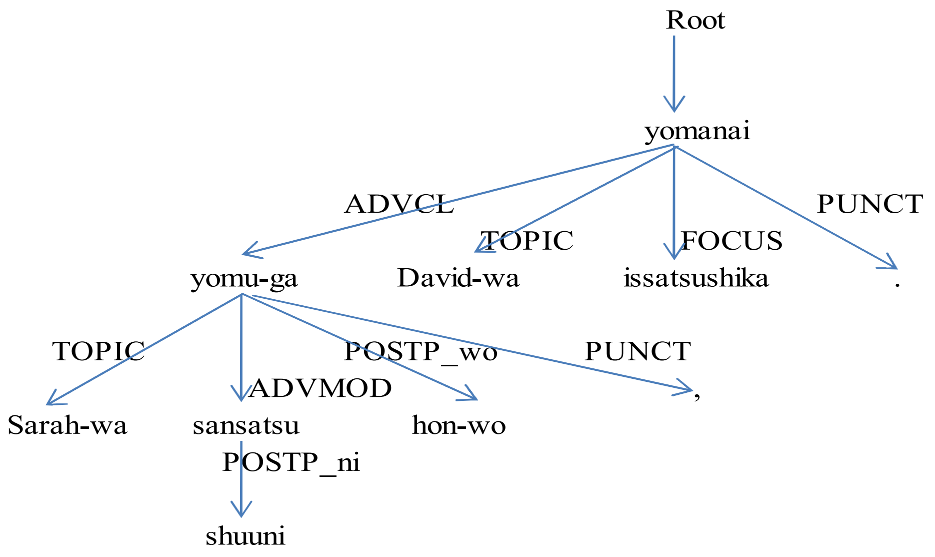


Figure 6.15. The typed-dependency tree for 'Sarawa shuuni sansatsu honwo yomuga, Davidwa issatsushika yomanai.' (Oya 2014)

The typed-dependency tree above is equivalent to the functional structure below. The index ‘j’ is assigned to the object zero pronoun of the verb ‘yomu (someone reads something)’ in the adverbial clause, and to the word ‘hon (book)’ which is the value of the POSTP_wo attribute. However, the index ‘j’ is not assigned to the object zero pronoun of the verb ‘yomanai (someone does not read something).’ This is because the focus of this sentence is to express their habit, not the identity of the books they read, and it is natural to interpret this sentence that Sarah and David read different books.

```
|       |PRED      'yomanai<SUBJ, DOBJ>'                                     ||
|       |SUBJ      |PRED       'PRO_k'   |                                   ||
|       |          |FORM       ZERO      |                                   ||
|       |                                                                    ||
|       |DOBJ      |PRED       'PRO'     |                                   ||
|       |          |FORM       ZERO      |                                   ||
|       |                                                                    ||
|       |ADVCL     |PRED       'yomu<SUBJ,OBJ>'                         |    ||
|       |          |SUBJ       |PRED       'PRO_i'   |                  |    ||
|       |          |           |TYPE       ZERO      |                  |    ||
|       |          |                                                    |    ||
|       |          |OBJ        |PRED       'PRO_j'   |                  |    ||
|       |          |           |TYPE       ZERO      |                  |    ||
|       |          |                                                    |    ||
|       |          |TOPIC      |PRED       'Sarah_i' |                  |    ||
|       |          |                                                    |    ||
|       |          |ADVMOD     |PRED       'sansatsu'               |   |    ||
|       |          |           |POSTP_ni   |PRED     'shuuni'   |   |   |    ||
|       |          |                                                    |    ||
|       |          |POSTP_wo   |PRED       'hon_j'   |                  |    ||
|       |          |TENSE      PRESENT                                  |    ||
|       |TOPIC     |PRED       'David_k' |                                   ||
|       |                                                                    ||
|       |FOCUS     |PRED       'issatsu' |                                   ||
|       |TENSE     PAST                                                      ||
|       |NEG       +                                                         ||
|       |PUNCT     |FORM       '.'           |                               ||
|ROOT   |          |STMT-TYPE  DECLARATIVE   |                               ||
```

Figure 6.16. The functional structure for ‘Sarawa shuuni sansatsu honwo yomuga, Davidwa

issatsushika yomanai.' (Oya 2014)

6.4.4 Zero pronouns and topic

Oya (2010a) argued that the determination of the antecedent of a given zero pronoun is relevant to the ambiguity of the topic marker '-wa.' Consider the following tree for an example sentence 'Bokuwa unagida.'

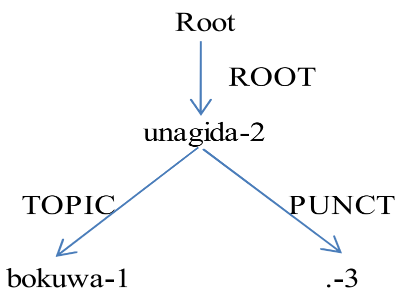


Figure 6.17. The typed-dependency tree for 'Bokuwa unagida.'

Kuno (1972) pointed out that the Japanese particle '-wa' has a *thematic* function and a *contrastive* function because a noun with '-wa' can be interpreted either thematically or contrastively. The essence of Oya's (2010a) argument is that a thematic '-wa' noun is the antecedent of the zero-pronoun subject, while a contrastive '-wa' noun is not the antecedent of the zero-pronoun subject; in the example above, the subject zero pronoun and the personal pronoun 'boku' refer to the same person. Therefore, the Japanese sentence 'Bokuwa' unagida.' with thematic interpretation of 'wa' means 'I am an eel' in English. This is indicated by the same index 'i' on the 'PRO' in SUBJ and on the 'PRO' in TOPIC.

	PRED	'unagi<SUBJ>'	
	SUBJ	PRED	'PROi'
		FORM	ZERO
	TOPIC	PRED	'PROi'
		FORM	'boku'
		PERSON	1ST
		NUMBER	SINGULAR
		STYLE	INFORMAL
	TENSE	PRESENT	
	PUNCT	FORM	'.'
ROOT		STMT-TYPE	DECLARATIVE

Figure 6.18. The functional structure for 'Bokuwa unagida.' with a thematic '-wa' noun

On the other hand, Oya (2010a) argued that a contrastive '-wa' noun is not the antecedent of the zero-pronoun subject. This means that the subject zero pronoun and the personal pronoun 'boku' refer to different entities. The 'PRO' in TOPIC refers to the speaker, while the 'PRO' in SUBJ refers to something other than the speaker. Therefore, the Japanese sentence 'Bokuwa unagida.' with a contrastive interpretation of 'wa' means 'As for me, it is an eel' in English. This is indicated by the different indices 'i' on the subject zero pronoun and 'j'on the 'PRO' in TOPIC.

	PRED	'unagi<SUBJ>'	
	SUBJ	PRED	'PROi'
		FORM	ZERO
	TOPIC	PRED	'PROj'
		FORM	'boku'
		PERSON	1ST
		NUMBER	SINGULAR
		STYLE	INFORMAL
	TENSE	PRESENT	
	PUNCT	FORM	'.'
ROOT		STMT-TYPE	DECLARATIVE

Figure 6.19. The functional structure for ‘Bokuwa unagida’ with a contrastive ‘-wa’ noun

In addition to the thematic ‘-wa’ noun, other postpositional nouns can also function as the inter-clausal antecedent for a core-argument zero pronoun. For example, consider the sentence ‘Kono honwa bokuga yonda (As for this book, I have read it).’ The typed-dependency tree for this sentence is shown below.

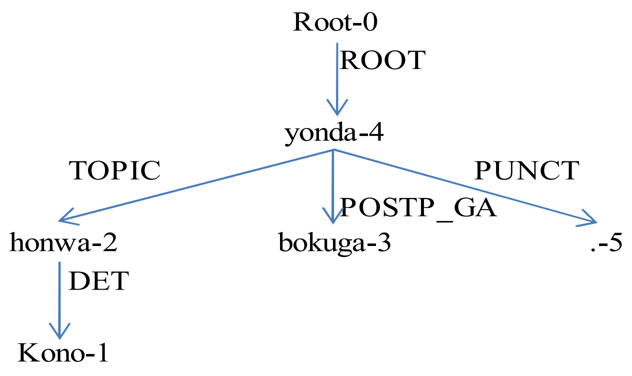


Figure 5.20. The typed-dependency tree for ‘Kono honwa bokuga yonda.’

The typed-dependency tree above is equivalent to the functional structure below. The noun ‘honwa’ is the antecedent for the object zero pronoun of the verb ‘yonda,’ and the index ‘j’ indicates this anaphoric relation. In addition, the postpositional phrase ‘bokuga’ is the antecedent for the subject zero pronoun of the verb ‘yonda,’ and the index ‘i’ indicates this anaphoric relation.

```
      PRED      'yonda<SUBJ, OBJ>'
      SUBJ      PRED      'PRO_i'
                FORM      ZERO

      OBJ       PRED      'PRO_j'
                FORM      ZERO

      TOPIC     PRED      'hon'
                DET       PRED ''kono'     j

      POSTP_GA  PRED      'PRO_i'
                FORM      'boku'
                PERSON    1ST
                NUMBER    SINGULAR
                STYLE     INFORMAL
      TENSE     PAST
      PUNCT     FORM      '.'
ROOT            STMT-TYPE DECLARATIVE
```

Figure 5.21. The functional structure for 'Kono honwa bokuga yonda (As for this book, I have read it).'

Nouns without postposition can be the antecedent for a zero pronoun. For example, consider the sentence 'Kono hon yonda?' in the dialogue (6.31). The typed-dependency tree for this sentence is shown below. The phrase 'hon' depends on the verb 'yonda' with the dependency type "advmod" because the phrase has no particle.

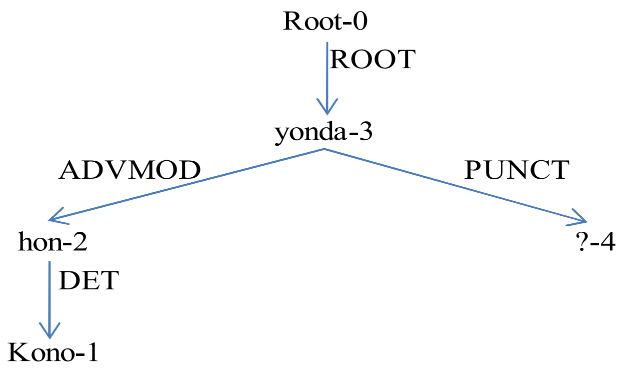


Figure 5.22. The typed-dependency tree for ‘Kono hon yonda?’

The noun ‘hon’ does not have a postposition in this sentence. It is also the antecedent for the object zero pronoun. Contextual judgment prevents the hearer from interpreting ‘hon’ as the antecedent for the *subject* zero pronoun because books cannot read anything.

```
     | PRED     'yonda<SUBJ, OBJ>'                      | |
     | SUBJ     | PRED      'PRO_i'           |         | |
     |          | FORM      ZERO              |         | |
     |                                                  | |
     | OBJ      | PRED      'PRO_j'           |         | |
     |          | FORM      ZERO              |         | |
     |                                                  | |
     | ADVMOD   | PRED      'hon'               |       | |
     |          | DET       |PRED ''kono'  |  |j      | |
     | TENSE    PAST                                    | |
     | PUNCT    | FORM      '?'               |         | |
ROOT |          | STMT-TYPE INTERROGATIVE               | |
```

Figure 5.23. The functional structure for ‘Kono hon yonda?’

The existence of zero pronouns is also relevant for cases in which the postposition ‘-ga’ and ‘-wo’ are interchangeable in a sentence. For example, Masuoka & Takubo (1997, p.75) state that nouns with ‘-ga’ can express the theme of an action the speaker is able to do (e.g., ‘Watashiwa eigoga hanaseru,’ which means ‘I can speak English’), or the theme of an action the speaker wants to do (e.g., ‘Watashiwa eigoga hanashitai,’ which means ‘I want to speak in English’). The ‘-ga’ can be replaced by ‘-wo.’ For example, it is possible to say either ‘Watashiwa eigo*ga* hanaseru’ or ‘Watashiwa eigo*wo* hanaseru.’ Different particles imply different meanings; the use of ‘-ga’ in such constructions implies that the speaker is emphasizing the noun with ‘-ga.’ Therefore, the postposition ‘-ga’ does not have the function to indicate the subject of a verb; rather, its function is determined by the context in which the sentence containing it is used.[44]

(6.16)

a. Watashiwa eigoga hanaseru.

b. Watashiwa eigowo hanaseru.

The lexical information for ‘hanaseru’ is as follows. The subject and object are lexically specified as zero pronouns.[45]

(6.17)

(↑PRED)= ‘hanaseru<SUBJ,OBJ>’

(↑SUBJ PRED)= ‘PRO’

(↑SUBJ FORM)=ZERO

(↑OBJ PRED)= ‘PRO’

(↑OBJ FORM)=ZERO

(↑TENSE)=PRESENT

(↑MODALITY)=POSSIBLE

---

44 Oya (2004) argued that the grammatical function of the particle “-ga” is represented in the constituent structure of Japanese. This study, however, does not take this stance, because this study does not presuppose the existence of the constituent structure of Japanese.

45 This study treats complex predicates as monoclausal. See Section 4.3.1.

The typed-dependency tree for 'Watashiwa eigoga hanaseru' is shown below. The noun 'eigoga' depends on the verb 'hanaseru' with the dependency type "postp_ga."

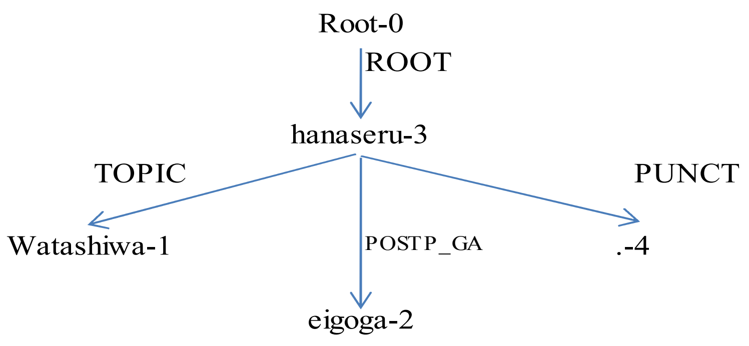


Figure 5.24. The typed-dependency tree for 'Watashiwa eigoga hanaseru.'

The typed-dependency tree for 'Watashiwa eigowo hanaseru' is shown below. The noun 'eigowo' depends on the verb 'hanaseru' with the dependency type "postp_wo."

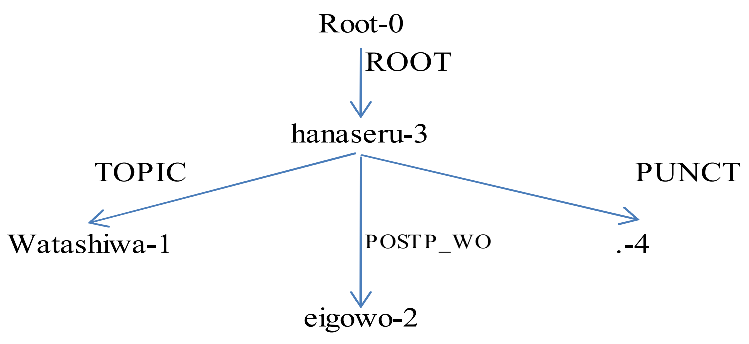


Figure 5.25. The typed-dependency tree for 'Watashiwa eigowo hanaseru.'

The typed-dependency tree for 'Watashiwa eigoga hanaseru' is equivalent to the functional structure shown below. The indices indicate that the topic ‘watashiwa’ is the antecedent of the subject zero pronoun, and the noun with the dependency type “postp_ga” is the antecedent of the object zero pronoun.

```
     | PRED       'hanaseru<SUBJ, OBJ>'                 | |
     |                                                  | |
     | SUBJ       | PRED      'PROi'            |       | |
     |            | FORM      ZERO              |       | |
     |                                                  | |
     | OBJ        | PRED      'PROj'            |       | |
     |            | FORM      ZERO              |       | |
     |                                                  | |
     | TOPIC      | PRED      'PROi'            |       | |
     |            | FORM      'watashi'         |       | |
     |            | PERSON    1st               |       | |
     |            | NUMBER    SINGULAR          |       | |
     |                                                  | |
     | POSTP_ga   | PRED      'eigoj'           |       | |
     | TENSE        PRESENT                             | |
     | MOOD         POSSIBLE                            | |
     | PUNCT      | FORM      '.'               |       | |
|ROOT|            | STMT-TYPE DECLARATIVE       |       | |
```

Figure 5.26. The functional structure for ‘Watashiwa eigoga hanaseru.’

The functional structure for ‘Watashiwa eigowo hanaseru’ is shown below. The indices indicate that the topic ‘watashiwa’ is the antecedent of the subject zero pronoun, and the noun with the dependency type “postp_wo” is the antecedent of the object zero pronoun.

```
       |PRED        'hanaseru<SUBJ, OBJ>'              | |
       |                                               | |
       |SUBJ       |PRED       'PROi'         |        | |
       |           |FORM       ZERO           |        | |
       |                                               | |
       |OBJ        |PRED       'PROj'         |        | |
       |           |FORM       ZERO           |        | |
       |                                               | |
       |TOPIC      |PRED       'PROi'         |        | |
       |           |FORM       'watashi'      |        | |
       |           |PERSON     1st            |        | |
       |           |NUMBER     SINGULAR       |        | |
       |                                               | |
       |POSTP_wo   |PRED       'eigoj'        |        | |
       |TENSE       PRESENT                            | |
       |MOOD        POSSIBLE                           | |
       |PUNCT      |FORM       '.'            |        | |
ROOT   |           |STMT-TYPE  DECLARATIVE    |        | |
```

Figure 5.27. The functional structure for ‘Watashiwa eigowo hanaseru.’

The functional structures in this section show that the verbal predicates of Japanese language contain the core arguments as zero pronouns in their lexicon. These zero pronouns refer to something or someone either intra-clausally or inter-clausally. These zero pronouns ensure that elliptic sentences that are often used in Japanese language do not violate the completeness constraint of functional structure; in other words, these “elliptic” sentences actually lack nothing.

From the standpoint which has been explained in this section, we must explore the condition on which a verb needs overt arguments. Intuitively speaking, such conditions must take the speaker’s/writer’s intention into consideration. For example, it can be claimed that a verb needs an overt noun phrase with the postposition ‘-ga’ when the speaker/writer intends to have the listener/hearer pay attention to one of the arguments of the verb. A detailed account of possible conditions on overt arguments goes beyond the scope of this dissertation, so I will leave it to future research.

6.5 The Definition of Each Dependency Type for Japanese

This section introduces the dependency types for Japanese sentences, and provides definitions to each of them with example sentences. The different dependency types in Japanese are presented in Table 6.1 along with their corresponding categories according to Mel’čuk’s syntactic relations framework (see Section 2.4.2 for the definition of Dsynt-Rels, and see Section 2.4.3 for the criteria for Ssynt-Rels).

Table 6.1. Japanese dependency types categorized into the inventory of Dsynt-Rels in Mel’čuk (2011, p.6)

coordinate DSyntRels		subordinate DSyntRels									
		weak subordinate DSyntRel	strong subordinate DSyntRels								
			modification		complementation						
COORD	QUASI-COORD	APPEND	ATTR	ATTRdescr	I	II	III	IV	V	VI	IIdir-sp
postp		advmod	advmod, amod, focus, det, postp, advcl, topic	rcmod	postp						ccomp

Similar to the discussion of English dependency types presented earlier in Section 5.3, this section defines each dependency type in Japanese with respect to the criteria for surface syntactic relations (SsyntRels) by Mel’čuk (2009, 2011). This discussion defines each Japanese dependency type in terms of Criterion A (i.e., the presence of the syntactic dependency between two words or elements), Criterion B (i.e., the orientation of the syntactic dependency), and

Criterion C (i.e., the type of the syntactic dependency). Also similar to the discussion of English dependency types, typed-dependency trees and functional-structure representations for the example Japanese sentences are illustrated, so that we can clarify the equivalence between typed-dependency trees and functional-structure representations (see Section 3.3).

6.5.1 Postp

Cases where a verb or a noun is the dependency head and a noun with a postposition (or a *case particle*) is the dependency tail are typed as "postp" (POSTPositional elements).

The dependency type "postp" has a number of subtypes according to the postposition particles used. For example, the dependency between a noun with the postposition '-ga' and a verb is subtyped as "postp_ga."[46] This subtyping is necessary because postpositions are not interchangeable with each other. In addition, the different subtypes allow us to derive dependency types that follow the revised Criterion C for SsyntRel, i.e., that each dependency type implies a specific semantic relationship between two words that is different from what is implied by any other dependency type (see Section 2.4.3.3).

Each postposition has more than one meaning, and some postpositions have ranges of meaning. For example, the case particle '-kara' can have the following meanings: the starting point of an event of moving, source of information, evidence, the date on which an event starts, or the material of a product (Masuoka & Takubo 1992, p.77-78). These different meanings are shown below.

The sentence (6.18) is an example in which the postposition '-kara' indicates the starting

[46] This study does not assume that a noun with the particle "-ga" is a subject, because there are instances where a noun with "-ga" does not function as a subject (Matsuoka & Takubo 1997, p.75).

point of an event of moving.

(6.18)

Watashi-wa	eki-kara	ie-made	arui-ta.
I-topic	*station-postp*	*house-postp*	*walk-pas*t

‘I walked from the station to my house.’

The dependency relationships among the syntactic units in the sentence above are represented in the following typed-dependency tree. The nominal syntactic unit[47] ‘ekikara (from the station)’ depends on the verbal syntactic unit ‘aruita (someone walked),’ and this dependency is typed as “postp_kara.”

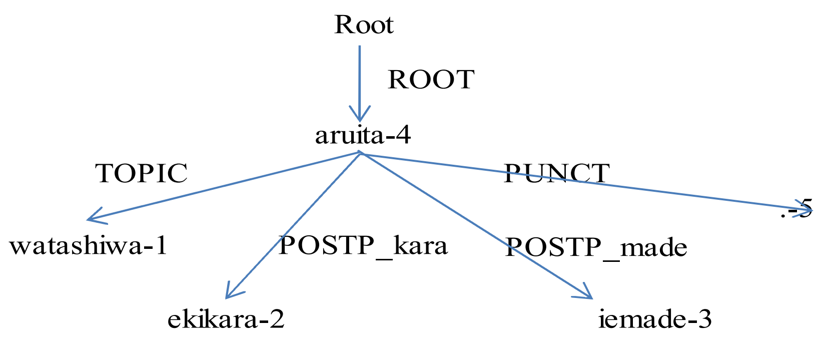


Figure 5.28. The typed-dependency tree for ‘Watashiwa ekikara iemade aruita (I walked from the station to my house).’

The typed-dependency tree above is equivalent to the functional structure shown below (see Section 6.4 on the treatment of zero pronouns).

[47] In this study, a nominal syntactic unit means a unit that contains one noun.

```
     | PRED          'aruita<SUBJ>'                  | |
     | SUBJ         | PRED      'PRO_i'            | | |
     |              | TYPE      ZERO               | | |
     |                                               | |
     | TOPIC        | PRED      'PRO_i'            | | |
     |              | PERSON    1ST                | | |
     |              | NUMBER    SINGULAR           | | |
     |              | FORM      'watashi'          | | |
     |                                               | |
     | POSTP_kara   | PRED      'eki'              | | |
     |                                               | |
     | POSTP_made   | PRED      'ie'               | | |
     | TENSE          PAST                           | |
     | PUNCT        | FORM      '.'                | | |
ROOT |              | STMT-TYPE DECLARATIVE        | | |
```

Figure 5.29. The functional structure for ‘Watashiwa ekikara iemade aruita (I walked from the station to my house).’

The sentence (6.19) is an example in which the postposition ‘-kara’ indicates the source of information.

(6.19)

Kono	hanashi-wa	ani-kara	kii-ta.
this	*story-topic*	*brother-postp*	*hear-pas*t

‘I heard this story from my elder brother.’

The dependency relationships among the syntactic units in the sentence above are represented in the following typed-dependency tree. The nominal syntactic unit ‘anikara (from my elder brother)’ depends on ‘kiita (someone heard something).’

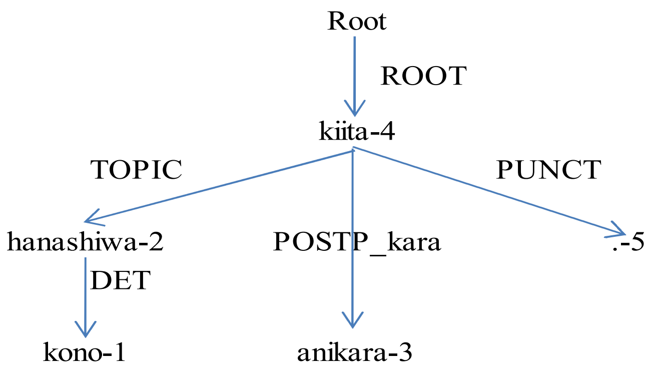


Figure 5.30. The typed-dependency tree for ‘Kono hanashiwa anikara kiita (I heard this story from my elder brother).’

The typed-dependency tree above is equivalent to the functional structure below. Notice that the subject zero pronoun of the verb ‘kiita’ is not assigned any index. This indicates that this zero pronoun does not refer to anything in the sentence, but to someone or something inter-clausally (it is naturally interpreted that the speaker of this sentence is also the subject of the verb ‘kiita’).

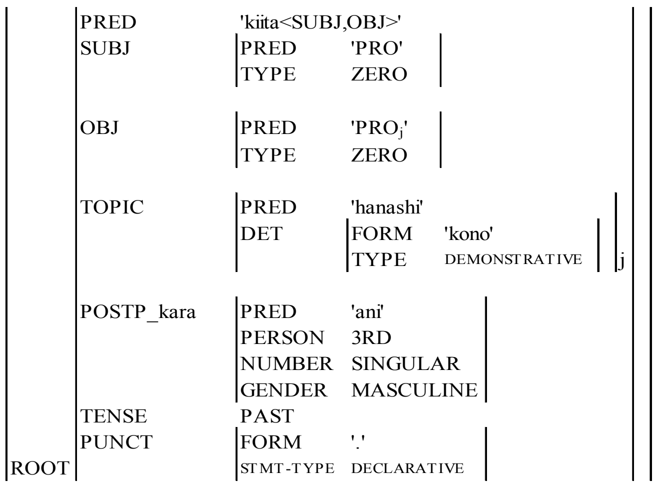

PRED 'kiita<SUBJ,OBJ>'
SUBJ PRED 'PRO' TYPE ZERO
OBJ PRED 'PROj' TYPE ZERO
TOPIC PRED 'hanashi' DET FORM 'kono' TYPE DEMONSTRATIVE j
POSTP_kara PRED 'ani' PERSON 3RD NUMBER SINGULAR GENDER MASCULINE
TENSE PAST
PUNCT FORM '.' STMT-TYPE DECLARATIVE
ROOT

Figure 5.31. The functional structure for ‘Kono hanashiwa anikara kiita.’

The sentence (6.20) is an example in which the postposition '-kara' indicates evidence.

(6.20)

Samazama-na	jijitu-kara	hitotsu-no	ketsuron-ga
various-adj	*fact-postp*	*one-postp*	*conclusion-postp*
michibikidas-are-ta.			
induce-passive-past.			

'One conclusion was induced from various facts.'

The dependency relationships among the syntactic units in the sentence above are represented in the following typed-dependency tree. The nominal syntactic unit 'jijitsukara (from facts)' depends on 'michibikidasareta (something was induced).'

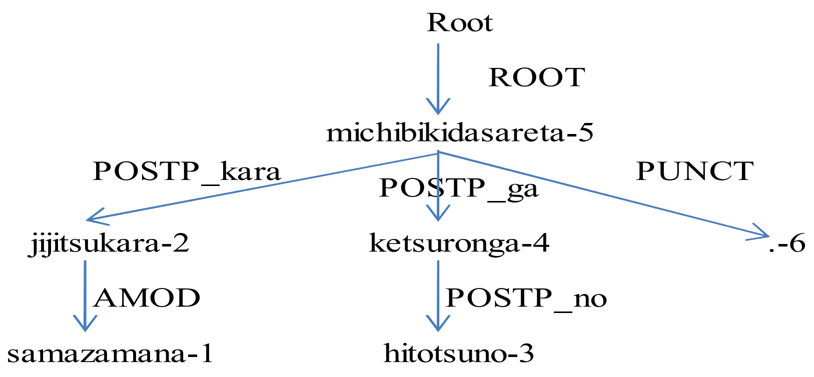


Figure 5.32. The typed-dependency tree for 'Samazamana jijitsukara hitotsuno ketsuronga michibikidasareta (One conclusion is induced from various facts).'

The typed-dependency tree above is equivalent to the functional structure below. Notice that the verb 'michibikidasareta' is a passive form of the verb 'michibikidasu,' and the attribute

VOICE has the value PASSIVE.[48]

```
     | PRED          'michibikidasareta<SUBJ>'                        | |
     | SUBJ          |PRED        'PROi'       |                      | |
     |               |TYPE        ZERO         |                      | |
     |                                                                | |
     | POSTP_kara    |PRED        'jijitsu'                   |       | |
     |               |ADJ         |PRED    'samazamana' |    |       | |
     |                                                                | |
     | POSTP_ga      |PRED        'ketsuron'           |              | |
     |               |POSTP_no    |PRED    'hitotsu' | |i             | |
     | TENSE         PAST                                             | |
     | VOICE         PASSIVE                                          | |
     | PUNCT         |FORM        '.'                 |               | |
ROOT |               |STMT-TYPE   DECLARATIVE         |               | |
```

Figure 5.33. The functional structure for ‘Samazamana jijitsukara hitotsuno ketsuronga michibikidasareta (One conclusion is induced from various facts).’

The sentence (6.21) is an example in which the postposition ‘-kara’ indicates the date on which an event starts.

(6.21)

Ashita-kara	toukikyuuka-ga	hajimar-u.
tomorrow-postp	*winter.holidays-postp*	*start-present*

‘Winter holidays start tomorrow.’

The dependency relationships among the syntactic units in the sentence above are represented in

[48] This study analyses complex predicate monoclausally both in typed-dependency trees and in functional-structure representations. See Section 4.3.1.

the following typed-dependency tree. The syntactic unit ‘ashitakara (from tomorrow)’ depends on the syntactic unit ‘hajimaru (something starts).’

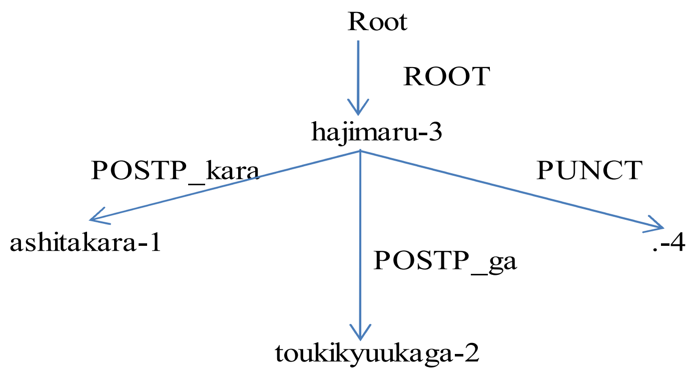


Figure 5.34. The typed-dependency tree for ‘Ashitakara toukikyuukaga hajimaru’ (Winter holidays start tomorrow).

The typed-dependency tree above is equivalent to the functional structure below.

```
|     |PRED        'hajimaru<SUBJ>'                   | |
|     |SUBJ        |PRED        'PRO_i'      |        | |
|     |            |TYPE        ZERO         |        | |
|     |                                               | |
|     |POSTP_ga    |PRED        'toukikyuuka_i'|      | |
|     |                                               | |
|     |POSTP_kara  |PRED        'ashita'     |        | |
|     |TENSE       PRESENT                            | |
|     |PUNCT       |FORM        '.'               |   | |
|ROOT |            |STMT-TYPE   DECLARATIVE       |   | |
```

Figure 5.35. The functional structure for ‘Ashitakara toukikyuukaga hajimaru’ (Winter holidays start tomorrow).

The sentence (6.22) is an example in which the postposition ‘-kara’ indicates the material of a product.

(6.22)

Daizu-kara	toufu-ga	tsukur-are-ru.
soybeans-postp	*tofu-postp*	*make-passive-present*

‘Tofu is made from soybeans.’

The dependency relationships among the syntactic units in the sentence above are represented in the following typed-dependency tree. The syntactic unit ‘daizukara (from soybeans)’ depends on the syntactic unit ‘tsukurareru (something is made).’

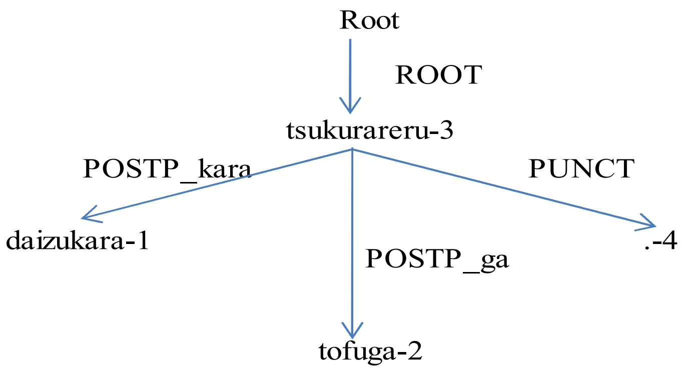


Figure 5.36. The typed-dependency tree for ‘Daizukara tofuga tsukurareru (Tofu is made from soybeans).’

The typed-dependency tree above is equivalent to the functional structure below. Notice that the verb ‘tsukurareru’ is a passive form of the verb ‘tsukuru,’ and the attribute VOICE has the value PASSIVE.

```
     PRED          'tsukurareru<SUBJ>'
     SUBJ          PRED        'PROi'
                   TYPE        ZERO

     POSTP_kara    PRED        'daizu'

     POSTP_ga      PRED        'toufui'
     TENSE         PRESENT
     VOICE         PASSIVE
     PUNCT         FORM        '.'
ROOT               STMT-TYPE   DECLARATIVE
```

Figure 5.37. The functional structure for ‘Daizukara tofuga tsukurareru (Tofu is made from soybeans).’

The prototypical meaning of a case particle is the underlying sense from which the different meanings emerge. In the case of ‘-kara,’ the different meanings seem to convey the sense of something coming into existence. Further exploration of these meaning relationships is beyond the scope of this dissertation. Nevertheless, the underlying meaning of a case particle cannot be expressed by another case particle, and in this way, particles are not usually interchangeable. Thus, we must differentiate the dependency type “postp” in terms of the case particle used, i.e., “padj_kara” in the examples above, which is similar to the output format of the Stanford Parser for collapsed dependencies (see Section 5.4.2).

The dependency type “postp” follows Mel’čuk’s criteria for SsyntRel. First, a verbal predicate and the noun with a postposition constitute a prosodic unit, and they have a fixed linear order in a sentence, e.g., the noun with a postposition precedes the predicate on which it depends. They also follow the revised Criterion A (see Section 2.4.3.1) because it constitutes a semantic unit, as we have seen in the functutional structure representations above. Second, the predicate determines the passive valence of the phrase. Therefore, the predicate and the noun with a case particle follow Mel’čuk’s Criterion B1. This dependency type also follows the revised

Criterion C proposed in Section 2.4.3.3, because it implies a unique semantic relationship between the noun with the '-ni' case and a verb that cannot be expressed by any other dependency type.

In this study, the subject, direct object, and indirect object in Japanese sentences are included in the "postp" subtypes. For example, a noun with the postposition '-ga,' which is usually considered as the subject of a verb, depends on a verb with the dependency type "postp_ga;" a noun with the postposition '-wo' depends on a verb with the dependency type "postp_wo;" and a noun with the postposition '-ni' depends on a verb with the dependency type "postp_ni."

Postp_ga

The dependency type "postp_ga" implies several unique semantic relationships not represented by other dependency types, and these semantic relationships are not restricted to the subject of a verbal predicate. For example, Masuoka & Takubo (1992, p.75) state that this type can express the agent of an action (an example of this is shown above), the theme of an action the speaker is able to do (e.g., 'Watashiwa eigoga hanaseru,' which means '*I can speak English.*' See Section 6.4.4 for the typed-dependency tree and the functional structure for the example sentence), or the theme of an action the speaker wants to do (e.g., 'Watashiwa eigoga hanashitai,' which means '*I want to speak in English.*' See Section 6.4.4 for the typed-dependency tree and the functional structure for the example sentence).

The sentence (6.23) is an example in which the postposition '-ga' is used.

(6.23)

Watashi-ga	kono	hon-wo	yon-da.
I-postp	*this*	*book-postp*	*read-past*

‘I have read this book.’[49]

The dependency relationships among the syntactic units in the sentence above are represented in the following typed-dependency tree. The syntactic unit ‘watashiga (I)’ depends on ‘yonda (someone has read something).’

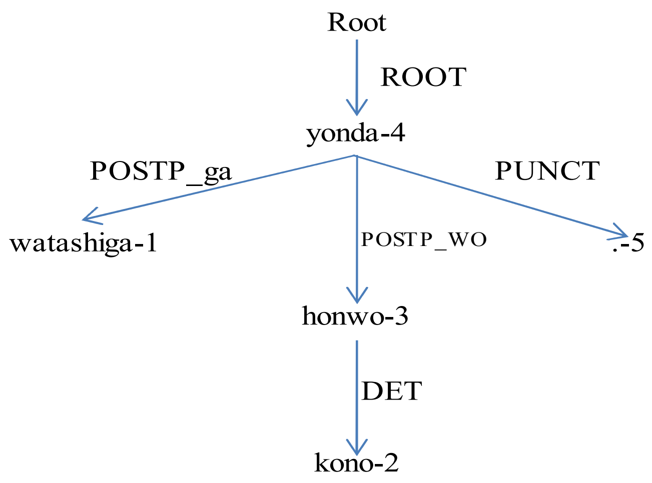


Figure 5.38. The typed-dependency tree for ‘Watashiga kono honwo yonda’ (I have read this book).

The typed-dependency tree above is equivalent to the functional structure below. The subject zero pronoun of the verb ‘yonda (someone has read something)’ refers to the syntactic unit ‘watashiga (I).’ The object zero pronoun of the same verb refers to the phrase ‘kono honwo (this book).’

[49] Japanese past-tense verbs can express the perfect aspect.

```
     PRED        'yonda<SUBJ,OBJ>'
     SUBJ        [PRED     'PRO_i'    ]
                 [TYPE     ZERO       ]

     OBJ         [PRED     'PRO_j'    ]
                 [TYPE     ZERO       ]

     POSTP_ga    [PRED     'PRO_i'    ]
                 [PERSON   1ST        ]
                 [NUMBER   SINGULAR   ]
                 [FORM     'watashi'  ]

     POSTP_wo    [PRED     'hon'                            ]
                 [DET      [FORM     'kono'          ]      ]
                 [         [TYPE     DEMONSTRATIVE   ]      ]_j
     TENSE       PRESENT
     ASPECT      PERFECT
     PUNCT       [FORM       '.'          ]
ROOT             [STMT-TYPE  DECLARATIVE  ]
```

Figure 5.39. The functional structure for 'Watashiga kono honwo yonda' (I have read this book).

Postp_wo

The dependency type "postp_wo" implies several unique semantic relationships not represented by other dependency types. Masuoka & Takubo (1992, p.75) state that there are three different semantic relationships expressed by "postp_wo": the object of an action or emotion (e.g., 'Watashi-ga kono hon-wo yon-da,' which means '*I have read this book.*'); the place where something or someone moves (e.g., 'Hakucho-wa mizuumi-no ue-wo susunde-ita,' which means '*Swans were moving on the surface of the lake.*'); or the place where something or someone starts moving (e.g., 'Watashi-wa daigaku-wo daitai gogorokuji-ni de-ta,' which means '*I left university around 6 P.M.*').

The following sentence (6.24) is an example in which '-wo' indicates the place where something or someone moves.

(6.24)

Hakucho-wa mizuumi-no ue-wo susunde-ita

swan-topic *lake-postp* *surface-postp* *move-past.progressive*

'Swans were moving on the surface of the lake.'

The dependency relationships among the syntactic units in the sentence above are represented in the following typed-dependency tree. The syntactic unit 'ue-wo (on the surface)' depends on the syntactic unit 'susunde-ita (something was moving),' expressing the place where the swans were moving.

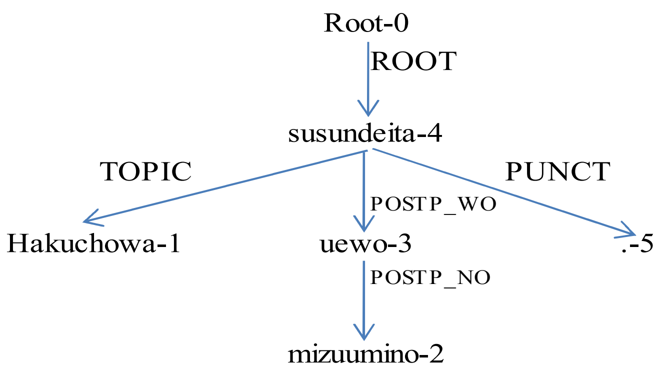


Figure 5.40. The typed-dependency tree for 'Hakuchowa mizuumino uewo susundeita. (Swans were moving on the surface of the lake.)'

The typed-dependency tree above is equivalent to the functional structure below. The verb 'susundeita (was moving)' is an intransitive verb; therefore, it does not have its object zero pronoun, and the syntactic unit 'uewo' is *not* the object of the verb 'susundeita.'

```
     |    |PRED       'susundeita<SUBJ>'                     | |
     |    |SUBJ       |PRED       'PRO_i'   |                | |
     |    |           |TYPE       ZERO      |                | |
     |    |                                                  | |
     |    |TOPIC      |PRED       'hakucho_i'  |             | |
     |    |                                                  | |
     |    |POSTP_wo   |PRED       'ue'                   |   | |
     |    |           |POSTP_no   |PRED       'mizuumi'| |   | |
     |    |TENSE      PAST                                   | |
     |    |ASPECT     PROGRESSIVE                            | |
     |    |PUNCT      |FORM       '.'          |             | |
     |ROOT|           |STMT-TYPE  DECLARATIVE  |             | |
```

Figure 5.41. The functional structure for ‘Hakuchowa mizuumino uewo susundeita. (Swans were moving on the surface of the lake.)’

The following sentence (6.25) is an example in which ‘-wo’ indicates the place where something or someone starts moving.

(6.25)

Watashi-wa	daigaku-wo	daitai	gogorokuji-ni	de-ta
I-topic	*university-postp*	*around*	*6P.M.-postp*	*leave-past*

‘I left university around 6 P.M.’

The dependency relationships among the syntactic units in the sentence above are represented in the following typed-dependency tree. The syntactic unit ‘daigaku-wo (university)’ depends on the syntactic unit ‘deta (someone left),’ expressing the place where the person which the pronoun ‘watashi’ refers to started the action of leaving.

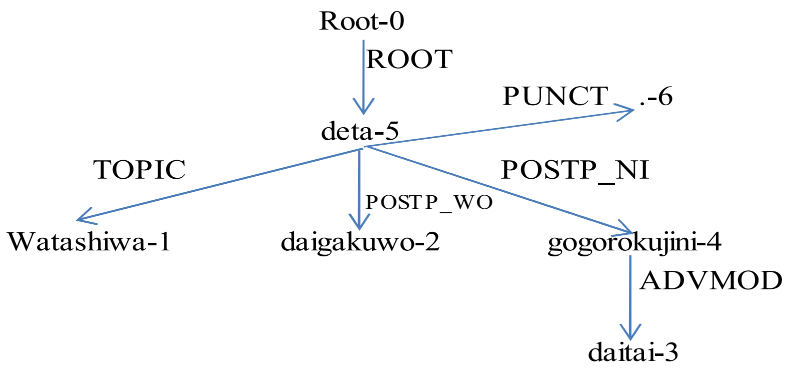


Figure 5.42. The typed-dependency tree for ‘Watashiwa daigakuwo daitai gogorokujini deta. (I left university around 6 P.M.)’

The typed-dependency tree above is equivalent to the functional structure below. The verb ‘deta (someone left)’ is an intransitive verb; therefore, it does not have an object zero pronoun, and the syntactic unit ‘daigakuwo’ is *not* the object of the verb.

```
|        |PRED        'deta<SUBJ>'                       ||
|        |SUBJ        |PRED       'PRO_i'      |         ||
|        |            |TYPE       ZERO         |         ||
|        |                                               ||
|        |TOPIC       |PRED       'PRO_i'          |     ||
|        |            |PERSON     1ST              |     ||
|        |            |NUMBER     SINGULAR         |     ||
|        |            |FORM       'watashi'        |     ||
|        |                                               ||
|        |POSTP_wo    |PRED       'daigaku'    |         ||
|        |                                               ||
|        |POSTP_ni    |PRED       'gogokuji'   |         ||
|        |                                               ||
|        |TENSE       PAST                               ||
|        |PUNCT       |FORM       '.'              |     ||
|ROOT    |            |STMT-TYPE  DECLARATIVE      |     ||
```

Figure 5.43. The functional structure for ‘Watashiwa daigakuwo daitai gogorokujini deta. (I left university around 6 P.M.)’

Postp_ni

The dependency type "postp_ni" represents dependency between a verb and a noun with the postposition '-ni.' This dependency type can imply the following semantic relationships: the receiver in a giving event, the goal of a moving event, the agent of a passive sentence, the agent of an adversative passive sentence, or the causee of a causative sentence. These different meanings are illustrated in the examples below.

The following sentence (6.26) is an example in which the postposition '-ni' indicates the receiver in a giving event.

(6.26)

Ken-wa	Naomi-ni	sono	hon-wo	kashi-ta.
Ken-topic	*Naomi-postp*	*the*	*book-postp*	*lend-past*

'Ken lent Naomi the book.'

The dependency relationships among the syntactic units in the sentence above are represented in the following typed-dependency tree. The syntactic unit 'Naomini' depends on the verb 'kashita (someone lent something to someone),' indicating the receiver in the event expressed by the verb 'kashita.'

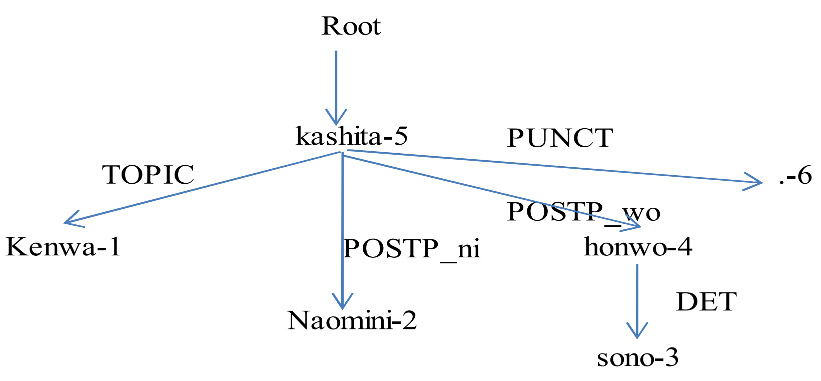


Figure 5.44. The typed-dependency tree for ‘Kenwa Naomini sono honwo kashita (Ken lent Naomi the book).’

The typed-dependency tree above is equivalent to the functional structure below. The index ‘k’ is assigned to the second-object (OBJ2) zero pronoun of the verb ‘kashita (lent)’ and to ‘Naomi,’ indicating that this second-object zero pronoun refers to ‘Naomi.’ Therefore, she is the receiver of the action expressed by the verb ‘kashita’ in this functional structure.

```
      | PRED       'kashita<SUBJ,OBJ,OBJ2>'                          | |
      | SUBJ      |PRED      'PROᵢ'       |                          | |
      |           |TYPE      ZERO         |                          | |
      |                                                              | |
      | OBJ       |PRED      'PROⱼ'       |                          | |
      |           |TYPE      ZERO         |                          | |
      |                                                              | |
      | OBJ2      |PRED      'PROₖ'       |                          | |
      |           |TYPE      ZERO         |                          | |
      |                                                              | |
      | TOPIC     |PRED      'Kenᵢ'       |                          | |
      |                                                              | |
      | POSTP_wo  |PRED      'hon'                           |       | |
      |           |DET      |FORM      'sono'            |   |       | |
      |           |         |TYPE      DEMONSTRATIVE     |   |j      | |
      |                                                              | |
      | POSTP_ni  |PRED      'Naomiₖ'     |                          | |
      |                                                              | |
      | TENSE      PAST                                              | |
      | PUNCT     |FORM      '.'              |                      | |
|ROOT |           |STMT-TYPE DECLARATIVE      |                      | |
```

Figure 5.45. The functional structure for ‘Kenwa Naomini sono honwo kashita (Ken lent Naomi the book).’

The sentence (6.27) is an example in which the postposition ‘-ni’ indicates the goal of a motion event.

(6.27)

Watashi-wa	Kinou	Ueno-ni	it-ta.
I-postp	*Yesterday*	*Ueno-postp*	*go-past*

“I went to Ueno yesterday.”

The dependency relationships among the syntactic units in the example sentence above are the typed-dependency tree below. The dependency type “postp_ni” is used for the dependency between the syntactic unit ‘uenoni’ and ‘itta.’

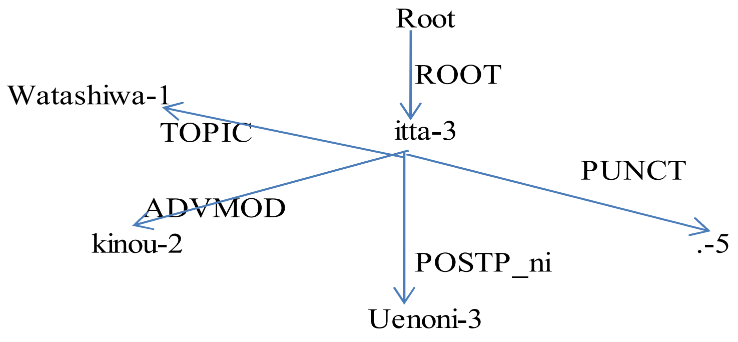


Figure 5.46. The typed-dependency tree for ‘Watashiwa kinou Uenoni itta (I went to Ueno yesterday).’

The typed-dependency tree above is equivalent to the functional structure below. The syntactic unit ‘Uenoni’ modifies the clause with the meaning of the event’s goal.

$$\left[\begin{array}{ll} \text{ROOT} & \left[\begin{array}{ll} \text{PRED} & \text{'itta<SUBJ>'} \\ \text{SUBJ} & \left[\begin{array}{ll} \text{PRED} & \text{'PRO}_i\text{'} \\ \text{TYPE} & \text{ZERO} \end{array}\right] \\ & \\ \text{TOPIC} & \left[\begin{array}{ll} \text{PRED} & \text{'PRO}_i\text{'} \\ \text{PERSON} & \text{1ST} \\ \text{NUMBER} & \text{SINGULAR} \\ \text{FORM} & \text{'watashi'} \end{array}\right] \\ & \\ \text{POSTP\_ni} & \left[\begin{array}{ll} \text{PRED} & \text{'Ueno'} \end{array}\right] \\ & \\ \text{ADVMOD} & \left[\begin{array}{ll} \text{PRED} & \text{'kinou'} \end{array}\right] \\ & \\ \text{TENSE} & \text{PAST} \\ \text{PUNCT} & \left[\begin{array}{ll} \text{FORM} & \text{'.'} \\ \text{STMT-TYPE} & \text{DECLARATIVE} \end{array}\right] \end{array}\right] \end{array}\right]$$

Figure 5.47. The functional structure for ‘Watashiwa kinou Uenoni itta (I went to Ueno yesterday).’

The following sentence (6.28) is an example in which the postposition “-ni” indicates the agent of a passive sentence.

(6.28)

Watashi-no	nikki-ga	ane-ni	yom-are-ta.
I-postp	*diary-postp*	*elder.sister-postp*	*read-passive-past*

‘My diary was read by my elder sister.’

The dependency relationships among the syntactic units in the example sentence above are shown in the typed-dependency tree below. The dependency type “postp_ni” is used for the dependency between the syntactic unit ‘aneni (by my sister)’ and ‘yomareta (something was read).’

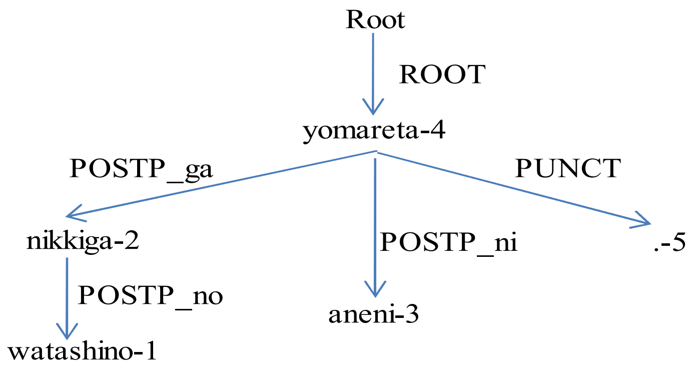


Figure 5.48. The typed-dependency tree for ‘Watashino nikkiga aneni yomareta (My diary was read by my elder sister).’

The typed-dependency tree above is equivalent to the functional structure below. The subject zero pronoun of the verb ‘yomareta (something was read)’ refers to ‘watashino nikkiga (my diary),’ which corresponds to the local functional structure as the value of the POSTP_ga attribute. The passivized verb ‘yomareta’ does not syntactically require the “postp_ni” syntactic unit; rather, the “postp_ni” syntactic unit modifies the clause with the meaning of the agent of the event.

```
|      |PRED       'yomareta<SUBJ>'                                 ||
|      |SUBJ      |PRED       'PRO_i'  |                            ||
|      |          |TYPE       ZERO     |                            ||
|      |                                                            ||
|      |POSTP_ga  |PRED       'nikki'                          |    ||
|      |          |POSTP_no  |PRED      'PRO'          |      |    ||
|      |          |          |PERSON    1ST            |      |    ||
|      |          |          |NUMBER    SINGULAR       |      |    ||
|      |          |          |FORM      watashi'       |      |i   ||
|      |                                                            ||
|      |POSTP_ni  |PRED       'ane'      |                          ||
|      |          |PERSON     3RD        |                          ||
|      |          |NUMBER     SINGULAR   |                          ||
|      |          |GENDER     FEMININE   |                          ||
|      |VOICE      PASSIVE                                          ||
|      |TENSE      PAST                                             ||
|      |PUNCT     |FORM       '.'            |                      ||
|ROOT  |          |STMT-TYPE  DECLARATIVE    |                      ||
```

Figure 5.49. The functional structure for ‘Watashino nikkiga aneni yomareta (My diary was read by my elder sister).’

The next sentence (6.29) is an example in which the postposition “-ni” indicates the agent of an adversative passive sentence.[50]

(6.29)

Watashi-wa	jibun-no	nikki-wo	ane-ni	yom-are-ta.
I-topic	*self-postp*	*diary-postp*	*elder.sister-postp*	*read-passive-past*

‘I had my diary read by my elder sister.’

[50] The subject of an adversative passive sentence is “adversely affected by the event denoted by the rest of the sentence” (Tsujimura 2007). The example sentence (6.29) means that the subject “I” suffers from the fact that his or her diary was read by his or her elder sister. The example sentence (6.28), on the other hand, is an objective description of an event, and the subject’s suffering is not evident in its sentential meaning without context. In a certain context, the example sentence (6.28) can imply that the speaker suffers from the event; the example sentence (6.29), on the other hand, explicitly means that the subject “I” suffers from the event.

The dependency relationships among the syntactic units in the example sentence above are shown in the typed-dependency tree below. The dependency type "postp_ni" is used for the dependency between the syntactic unit 'aneni (by my elder sister)' and 'yomareta (something was read).'

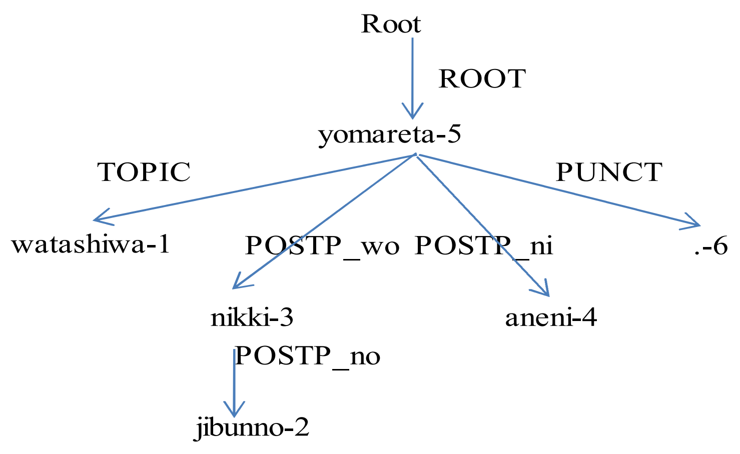


Figure 5.50. The typed-dependency tree for 'Watashiwa jibunno nikkiwo aneni yomareta (I had my diary read by my elder sister).'

The typed-dependency tree above is equivalent to the functional structure below. The topic is not required by the main verb of this sentence; rather, it modifies the verb with the meaning of the person who suffers from the event which the main verb of the sentence describes. The subject zero pronoun of the verb 'yomareta (something was read)' refers to 'watashino nikkiwo (my diary),' which corresponds to the local functional structure as the value of the POSTP_wo attribute.[51]

[51] In this study, we assume that the selection of the postposition "wo" for the theme argument of the adversative-passivized transitive verb is a matter of convention, probably to indicate the difference between normal passives and adversative passives.

```
|         |PRED       'yomareta<SUBJ>'                                   ||
|         |SUBJ      |PRED      'PRO_i'    |                            ||
|         |          |TYPE      ZERO       |                            ||
|         |                                                             ||
|         |TOPIC     |PRED      'PRO_j'    |                            ||
|         |          |PERSON    1ST        |                            ||
|         |          |NUMBER    SINGULAR   |                            ||
|         |          |FORM      'watashi'  |                            ||
|         |                                                             ||
|         |POSTP_wo  |PRED      'nikki'                        |  |     ||
|         |          |POSTP_no |PRED      'PRO_j'            |  |     ||
|         |          |         |TYPE      REFLEXIVE          |  |     ||
|         |          |         |FORM      jibun'             |  | i   ||
|         |                                                             ||
|         |POSTP_ni  |PRED      'ane'      |                            ||
|         |          |PERSON    3RD        |                            ||
|         |          |NUMBER    SINGULAR   |                            ||
|         |          |GENDER    FEMININE   |                            ||
|         |                                                             ||
|         |TENSE      PAST                                              ||
|         |PUNCT     |FORM      '.'          |                          ||
|ROOT     |          |STMT-TYPE DECLARATIVE  |                          ||
```

Figure 5.51. The typed-dependency tree for ‘Watashiwa jibunno nikkiwo aneni yomareta (I had my diary read by my elder sister).’

The sentence (6.30) is an example in which the postposition ‘-ni’ indicates the causee of a causative sentence:

(6.30)

Watashi-wa	ototo-ni	kono	hon-wo	yoma-se-ta.
I-topic	*younger.brother-postp*	*this*	*book-postp*	*read-cause-past*

‘I had my younger brother read this book.’

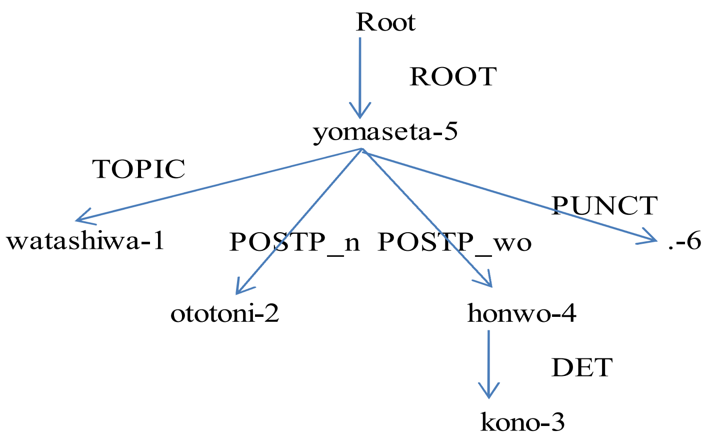
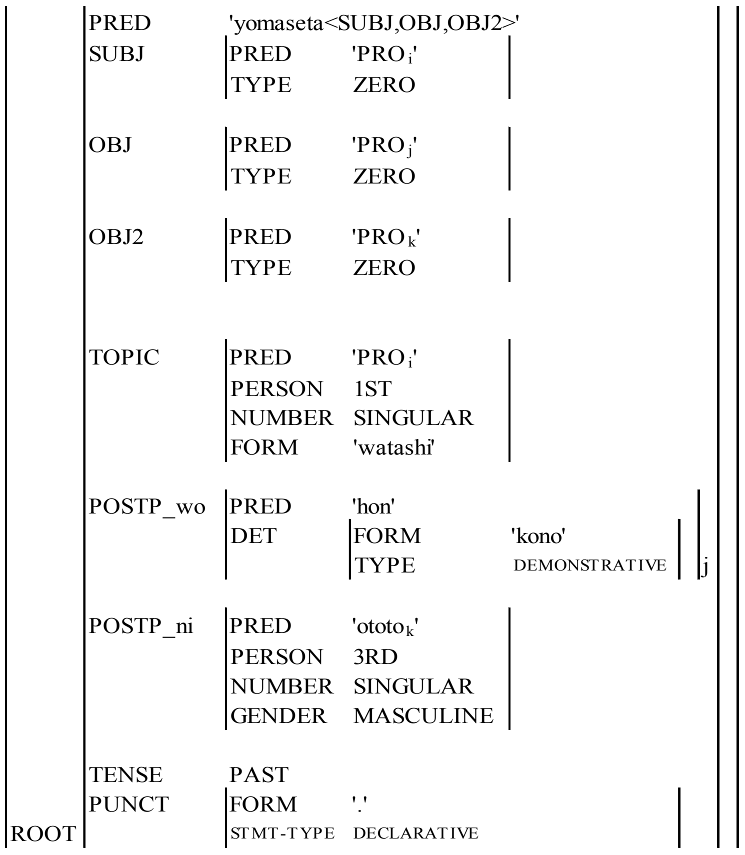


Figure 5.52. The typed-dependency tree for ‘Watashiwa ototoni kono honwo yomaseta (I had my younger brother read this book).’



PRED 'yomaseta<SUBJ,OBJ,OBJ2>'
SUBJ PRED 'PRO_i' TYPE ZERO
OBJ PRED 'PRO_j' TYPE ZERO
OBJ2 PRED 'PRO_k' TYPE ZERO
TOPIC PRED 'PRO_i' PERSON 1ST NUMBER SINGULAR FORM 'watashi'
POSTP_wo PRED 'hon' DET FORM 'kono' TYPE DEMONSTRATIVE j
POSTP_ni PRED 'ototo_k' PERSON 3RD NUMBER SINGULAR GENDER MASCULINE
TENSE PAST
PUNCT FORM '.' STMT-TYPE DECLARATIVE
ROOT

Figure 5.53. The functional structure for ‘Watashiwa ototoni kono honwo yomaseta (I had my younger brother read this book).’

6.5.2 Topic

The dependency between a verb and a noun with the postposition '-wa' is typed as "TOPIC." The noun with the postposition '-wa' introduces the topic of the sentence.

(6.31)

Kono	eki-wa	ooku-no	hitobito-ga	mainichi
This	*station-topic*	*many-post*	*people-postp*	*every.day*

riyousi-tei-ru.

use-progressive-present

'Many people use this station every day.'

The dependency relationships among the syntactic units in the example sentence above are represented by the typed-dependency tree below. The dependency type "topic" is used for the dependency relationship between 'ekiwa (station)' and 'riyoushiteiru (be using).'

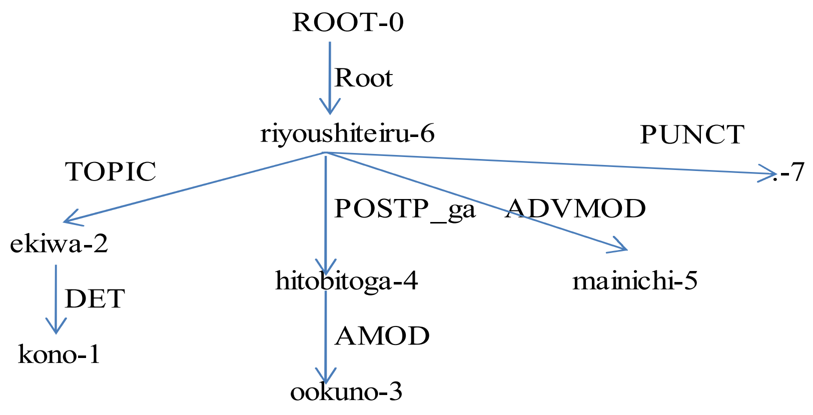


Figure 5.54. The typed-dependency tree for ‘Kono ekiwa ookuno hitobitoga mainichi riyousiteiru (Many people use this station every day).’

The typed-dependency tree above is equivalent to the functional structure below. Notice that the index ‘i’ is assigned to the subject zero pronoun of the verb ‘riyoushiteiru’ (be using) and the local functional structure as the value of the POSTP_ga attribute.

$$\left[\begin{array}{ll} & \left[\begin{array}{ll} \text{PRED} & \text{'riyoushiteiru<SUBJ,OBJ>'} \\ \text{SUBJ} & \left[\begin{array}{ll} \text{PRED} & \text{'PRO}_i\text{'} \\ \text{TYPE} & \text{ZERO} \end{array}\right] \\ \text{OBJ} & \left[\begin{array}{ll} \text{PRED} & \text{'PRO}_j\text{'} \\ \text{TYPE} & \text{ZERO} \end{array}\right] \\ \text{TOPIC} & \left[\begin{array}{ll} \text{PRED} & \text{'eki'} \\ \text{DET} & \left[\begin{array}{ll} \text{FORM} & \text{'kono'} \\ \text{TYPE} & \text{DEMONSTRATIVE} \end{array}\right] \end{array}\right]_j \\ \text{POSTP\_ga} & \left[\begin{array}{ll} \text{PRED} & \text{'hitobito'} \\ \text{NUM} & \text{PLURAL} \\ \text{AMOD} & \left[\begin{array}{ll} \text{PRED} & \text{'ookuno'} \end{array}\right] \end{array}\right]_i \\ \text{TENSE} & \text{PRESENT} \\ \text{PUNCT} & \left[\begin{array}{ll} \text{FORM} & \text{'.'} \\ \text{STMT-TYPE} & \text{DECLARATIVE} \end{array}\right] \end{array}\right] \\ \text{ROOT} & \end{array}\right]$$

Figure 5.55. The functional structure for 'Kono ekiwa ookuno hitobitoga mainichi riyousiteiru (Many people use this station every day).'

The postposition '-wa' can co-occur with another postposition. The sentence (6.32) is an example in which '-wa' co-occurs with '-ni.' The English translation indicates that the phrase 'to my sister' is put at the beginning of the sentence and thus topicalized.

(6.32)

Imoto-ni-wa	kono	hon-wo	age-ta.
younger.sister-postp-postp	*this*	*book-postp*	*give-past*

'To my sister, I gave this book.'

The dependency relationships among the syntactic units in the example sentence above are represented in the typed-dependency tree below. The dependency type "topic" is used for the dependency between 'imotoniwa (to my sister)' and 'ageta (gave).'

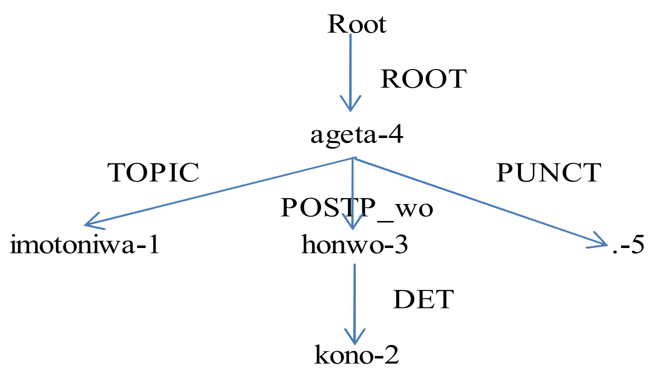


Figure 5.56. The typed-dependency tree for 'Imotoniwa kono honwo ageta (To my sister, I gave

this book).'

The typed-dependency tree above is equivalent to the functional structure below.

```
|  |PRED       'ageta<SUBJ,OBJ,OBJ2>'                          ||
|  |SUBJ      |PRED     'PRO_i'     |                          ||
|  |          |TYPE     ZERO        |                          ||
|  |                                                           ||
|  |OBJ       |PRED     'PRO_j'     |                          ||
|  |          |TYPE     ZERO        |                          ||
|  |                                                           ||
|  |OBJ2      |PRED     'PRO_k'     |                          ||
|  |          |TYPE     ZERO        |                          ||
|  |                                                           ||
|  |TOPIC     |PRED     'imoto'     |                          ||
|  |          |PERSON   3RD         |                          ||
|  |          |GENDER   FEMININE    |                          ||
|  |          |POSTP    'ni'        |k                         ||
|  |                                                           ||
|  |POSTP_wo  |PRED     'hon'                          |       ||
|  |          |DET     |FORM      'kono'          |   |       ||
|  |          |        |TYPE      DEMONSTRATIVE   |   |j      ||
|  |TENSE      PAST                                            ||
|  |PUNCT     |FORM     '.'         |                          ||
```

Figure 5.57. The functional structure for 'Imotoniwa kono honwo ageta (To my sister, I gave this book).'

The sentence (6.33) is an example in which '-wa' co-occurs with '-kara.' The English translation indicates that the phrase 'from this port' is put at the beginning of the sentence and thus topicalized.

(6.33)

Kono	minato-kara-wa	mainichi	ooku-no	fune-ga

This port-postp-postp every.day many-postp ship-postp

De-te-iru

depart-progressive-present

'From this port, many ships depart every day.'

The dependency relationships among the syntactic units in the example sentence above are represented in the typed-dependency tree below. The dependency type "topic" is used for the dependency between 'minatokarawa' and 'deteiru.'

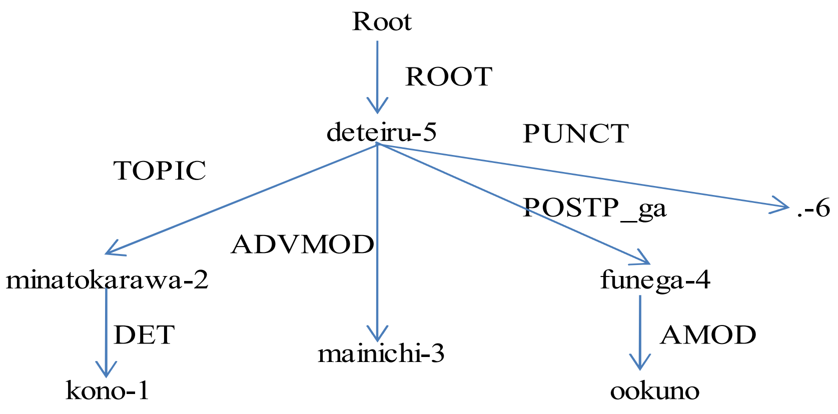


Figure 5.58. The typed-dependency tree for 'Kono minatokarawa mainichi ookuno funega deteiru (From this port, many ships depart every day).'

The typed-dependency tree above is equivalent to the functional structure below.

```
|    |PRED       'deteiru<SUBJ>'                                      ||
|    |SUBJ      |PRED    'PRO_i'       |                              ||
|    |          |TYPE    ZERO          |                              ||
|    |                                                                ||
|    |TOPIC     |PRED    'minato'                          |          ||
|    |          |DET    |FORM     'kono'         |         |          ||
|    |          |       |TYPE     DEMONSTRATIVE  |         |          ||
|    |          |POSTP   'kara'                            |          ||
|    |                                                                ||
|    |POSTP_ga  |PRED    'fune'                            |          ||
|    |          |AMOD   |PRED     'ookuno'       |         |i         ||
|    |TENSE      PRESENT                                              ||
|    |ASPECT     PROGRESSIVE                                          ||
|    |PUNCT     |FORM    '.'           |                              ||
|ROOT|          |STMT-TYPE DECLARATIVE |                              ||
```

Figure 5.59. The functional structure for ‘Kono minatokarawa mainichi ookuno funega deteiru (From this port, many ships depart every day).’

The dependency type “topic” follows Mel’čuk’s criteria for SsyntRel. First, a verbal predicate and a topic constitute a prosodic unit, and they have a fixed linear order in a sentence, e.g., the topic precedes the predicate. They also follow the revised Criterion A (see Section 2.4.3.1) because they constitute a semantic unit, as we have seen in the functional structure representations above.

Second, the predicate, not the topic, determines the passive valence of the phrase. In the example above, the topic ‘bokuwa’ depends on the predicate ‘unagida,’ not vice versa, because the predicate depends on another element in the sentence (i.e., Root). Therefore, the predicate and the topic follow Mel’čuk’s Criterion B1.

This dependency type also follows the revised Criterion C proposed in Section 2.4.3.3, because it implies a certain kind of semantic relationship between the topic and a verb that cannot be expressed by any other dependency type.

6.5.3 Focus

The dependency between a verb and a noun with a postposition other than '-ga,' '-ni,' '-wo,' or '-wa' is typed as "focus." Different postpositions have different discourse functions. For example, the postposition '-mo' functions like 'too' in English.

The following sentence (6.34) is an example in which the postposition '-mo' indicates that the subject 'watashi' is focused. The English translation indicates that the adverb 'too' modifies the pronoun 'I.'

(6.34)

Watashi-mo	kono	hon-wo	yon-da.
I-postp	*this*	*book-postp*	*read-past*

'I, too, have read this book.'

The dependency relationships among the syntactic units in the example sentence above are represented in the typed-dependency tree below. The dependency type "focus" is used for the dependency between 'watashimo (I, too)' and 'yonda (read).'

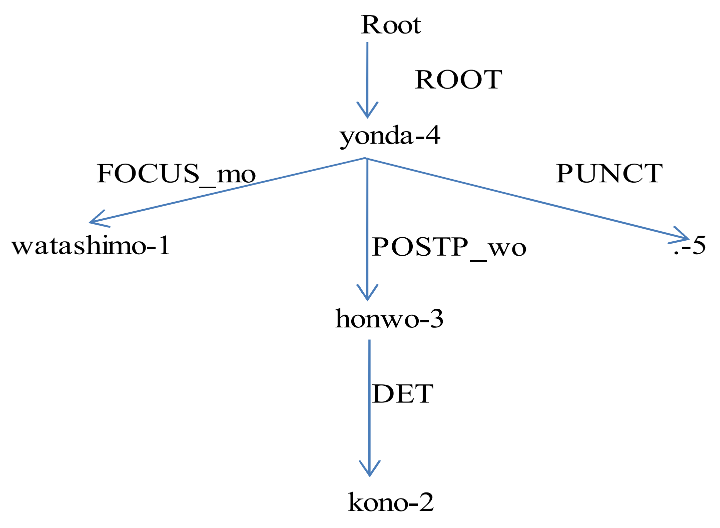


Figure 5.60. The typed-dependency tree for ‘Watashimo kono honwo yonda (I, too, have read this book).’

The typed-dependency tree above is equivalent to the functional structure below.

```
|      |PRED       'yonda<SUBJ, OBJ>'                  |  |
|      |SUBJ       |PRED       'PRO_i'        |        |  |
|      |           |FORM       ZERO           |        |  |
|      |                                               |  |
|      |OBJ        |PRED       'PRO_j'        |        |  |
|      |           |FORM       ZERO           |        |  |
|      |                                               |  |
|      |FOCUS_mo   |PRED       'PRO_i'        |        |  |
|      |           |FORM       'watashi'      |        |  |
|      |           |PERSON     1ST            |        |  |
|      |           |NUMBER     SINGULAR       |        |  |
|      |                                               |  |
|      |POSTP_wo   |PRED       'hon'          ||       |  |
|      |           |DET        |PRED ''kono' | |j      |  |
|      |TENSE      PAST                                |  |
|      |PUNCT      |FORM       '.'            |        |  |
|ROOT  |           |STMT-TYPE  DECLARATIVE    |        |  |
```

Figure 6.61. The functional structure for ‘Watashimo kono honwo yonda (I, too, have read this book).’

The sentence (6.35) is an example in which the postposition '-mo' indicates that the direct object 'hon' is focused. The English translation indicates that the adverb 'too' modifies the noun 'book':

(6.35)

Watashi-wa	kono	hon-mo	yon-da.
I-postp	*this*	*book-postp*	*read-past*

'I have read this book, too.'

The dependency relationships among the syntactic units in the example sentence above are represented in the typed-dependency tree below. The dependency type "focus" is used for the dependency between 'watashiwa (I)' and 'yonda (read).'

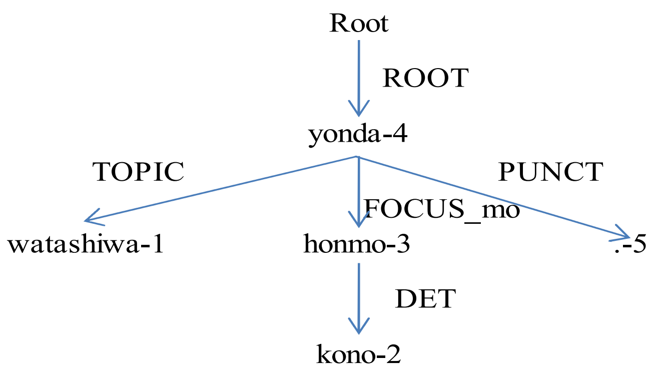


Figure 6.62. The typed-dependency tree for 'Watashiwa kono honmo yonda (I have read this book, too).'

The typed-dependency tree above is equivalent to the functional structure below.

```
|      |PRED       'yonda<SUBJ, OBJ>'                          |  |
|      |SUBJ      |PRED       'PRO$_i$'             |          |  |
|      |          |FORM       ZERO                  |          |  |
|      |                                                       |  |
|      |OBJ       |PRED       'PRO$_j$'             |          |  |
|      |          |FORM       ZERO                  |          |  |
|      |                                                       |  |
|      |TOPIC     |PRED       'PRO$_i$'             |          |  |
|      |          |FORM       'watashi'             |          |  |
|      |          |PERSON     1ST                   |          |  |
|      |          |NUMBER     SINGULAR              |          |  |
|      |                                                       |  |
|      |FOCUS_mo  |PRED       'hon'                   |        |  |
|      |          |DET       |PRED "kono'  |          | j      |  |
|      |TENSE      PAST                                        |  |
|      |PUNCT     |FORM       '.'                   |          |  |
|ROOT  |          |STMT-TYPE  DECLARATIVE           |          |  |
```

Figure 6.63. The functional structure for ‘Watashiwa kono honmo yonda (I have read this book, too).’

If a sentence contains two nouns with the postposition ‘-wa,’ as in the sentence (6.36), the second ‘wa’ functions like ‘such’ in English.

(6.36)

Watashiwa	konna	hon-wa	yoma-nai.
I-postp	*such*	*book-postp*	*read-neg*

‘I will not read such a book.’

The dependency relationships among the syntactic units in the example sentence above are represented in the typed-dependency tree below. The dependency type “focus” is used for the dependency between ‘honwa (book)’ and ‘yomanai (do not read).’

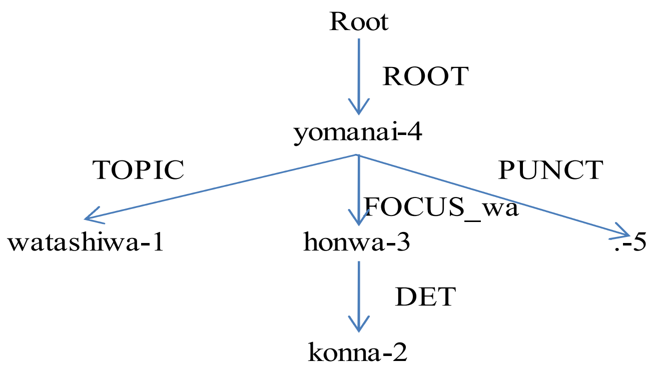


Figure 6.64. The typed-dependency tree for ‘Watashiwa konna honwa yomanai (I will not read such a book).’

The typed-dependency tree above is equivalent to the functional structure below.

```
|        |PRED       'yomanai<SUBJ, OBJ>'             |  |
|        |SUBJ       |PRED       'PRO_i'      |        |  |
|        |           |FORM       ZERO         |        |  |
|        |                                             |  |
|        |OBJ        |PRED       'PRO_j'      |        |  |
|        |           |FORM       ZERO         |        |  |
|        |                                             |  |
|        |TOPIC      |PRED       'PRO_i'      |        |  |
|        |           |FORM       'watashi'    |        |  |
|        |           |PERSON     1ST          |        |  |
|        |           |NUMBER     SINGULAR     |        |  |
|        |                                             |  |
|        |FOCUS_wa   |PRED       'hon'           |     |  |
|        |           |DET        |PRED "konna'| |j    |  |
|        |TENSE      PAST                              |  |
|        |NEG        +                                 |  |
|        |PUNCT      |FORM       '.'          |        |  |
|ROOT    |           |STMT-TYPE  DECLARATIVE  |        |  |
```

Figure 6.65. The functional structure for ‘Watashiwa konna honwa yomanai (I will not read such a book).’

This dependency type follows Mel'čuk's criteria for SsyntRel. First, a verbal predicate and the focused element constitute a prosodic unit, and they have a fixed linear order in a sentence, e.g., the focused element precedes the predicate. They also follow the revised Criterion A (see Section 2.4.3.1) because they constitute a semantic unit, as we have seen in the functional structure representations in this subsection.

Second, the predicate determines the passive valence of the phrase. Therefore, the predicate and the focused element follow Mel'čuk's Criterion B1.

This dependency type also follows the revised Criterion C proposed in Section 2.4.3.3, because it implies a certain kind of semantic relationship between the focused element and the verb that cannot be expressed by any other dependency type.

6.5.4 Advmod

This dependency type describes the dependency between a verb and an adverb, an adjunct in the adverbial form,[52] or a noun without a particle. Examples of this type are presented below. The sentence (6.37) is an example in which an adverb is used.

(6.37)

Watashi-wa	jibun-no	shorai-nitsuite	itsumo	kangae-te-iru.
I-postp	*myself-postp*	*future-postp*	*always*	*think-progressive-present*

'I am always thinking about my future.'

[52] Japanese adjectives inflect to function as adverbs. For example, the adjective 'osoi', which means 'late' in English, is inflected to 'osoku', which means 'lately.'

The dependency relationships among the syntactic units in the example sentence above are represented in the typed-dependency tree below. The dependency type "advmod" is used for the dependency between 'itsumo (always)' and 'kangaeteiru (be thinking about).'

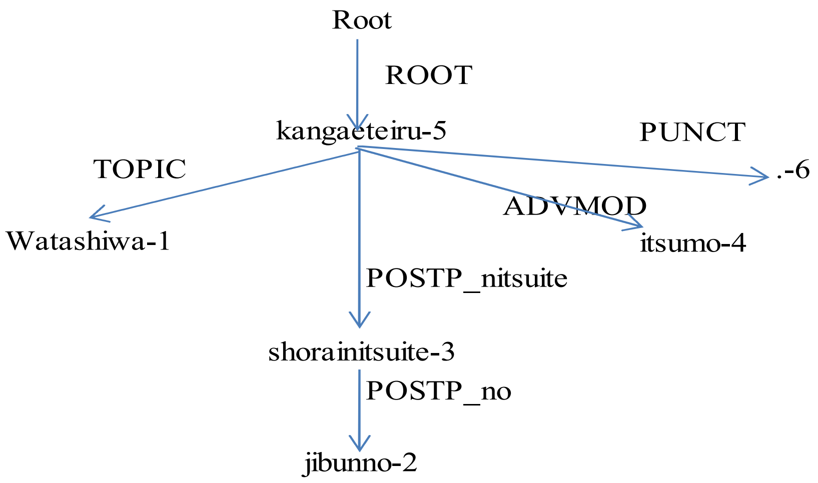


Figure 6.66. The typed-dependency tree for 'Watashiwa jibunno shorainitsuite itsumo kangaeteiru (I am always thinking about my future).'

The typed-dependency tree above is equivalent to the functional structure below. The reflexive pronoun 'jibun' refers to 'watashi.'

```
     | PRED       'kangaeteiru<SUBJ, OBJ>'                          | |
     | SUBJ      |PRED        'PROi'      |                         | |
     |           |FORM        ZERO        |                         | |
     |                                                              | |
     | OBJ       |PRED        'PROj'      |                         | |
     |           |FORM        ZERO        |                         | |
     |                                                              | |
     | TOPIC     |PRED        'PROi'      |                         | |
     |           |FORM        'watashi'   |                         | |
     |           |PERSON      1ST         |                         | |
     |           |NUMBER      SINGULAR    |                         | |
     |                                                              | |
     | POSTP_    |PRED        'shorai'                     |        | |
     | nitsuite  |POSTP_no   |PRED ''PROi'        |        |        | |
     |           |           |TYPE REFLEXIVE      |        |        | |
     |           |           |FORM 'jibun'        |        |j       | |
     |                                                              | |
     | ADVMOD    |PRED        'itsumo'    |                         | |
     | TENSE      PRESENT                                           | |
     | ASPECT     PROGRESSIVE                                       | |
     | PUNCT     |FORM        '.'         |                         | |
ROOT |           |STMT-TYPE   DECLARATIVE |                         | |
```

Figure 6.67. The functional structure for 'Watashiwa jibunno shorainitsuite itsumo kangaeteiru (I am always thinking about my future).'

The sentence (6.34) is an example in which an adjective in the adverbial form is used. The adverb 'yoku' is the adverbial form of an adjective 'yoi (good).'

(6.34)

Watashi-wa	jibun-no	shorai-nitsite	yoku	kangae-te-iru.
I-postp	*myself-postp*	*future-postp*	*often*	*think-progressive-present*

'I am often thinking about my future.'

The dependency relationships among the words in the sentence above are represented in the following typed-dependency tree. The dependency type "advmod" is used for the dependency between 'yoku (often)' and 'kangaeteiru (be thinking about).'

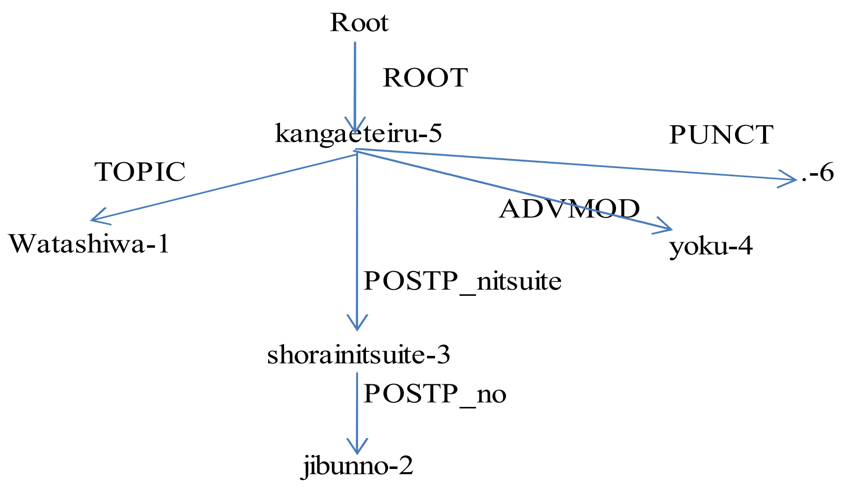


Figure 6.68. The typed-dependency tree for ‘Watashiwa jibunno shorainitsuite yoku kangaeteiru (I am often thinking about my future).’

The typed-dependency tree above is equivalent to the functional structure below.

```
|      |PRED      'kangaeteiru<SUBJ, OBJ>'                 |  |
|      |SUBJ      |PRED      'PRO_i'       |               |  |
|      |          |FORM      ZERO          |               |  |
|      |                                                   |  |
|      |OBJ       |PRED      'PRO_j'       |               |  |
|      |          |FORM      ZERO          |               |  |
|      |                                                   |  |
|      |TOPIC     |PRED      'PRO_i'       |               |  |
|      |          |FORM      'watashi'     |               |  |
|      |          |PERSON    1ST           |               |  |
|      |          |NUMBER    SINGULAR      |               |  |
|      |                                                   |  |
|      |POSTP_    |PRED      'shorai'                  |   |  |
|      |nitsuite  |POSTP_no  |PRED ''PRO_i'      |     |   |  |
|      |          |          |TYPE REFLEXIVE     |     |   |  |
|      |          |          |FORM 'jibun'       |     |j  |  |
|      |                                                   |  |
|      |ADVMOD    |PRED      'yoku'        |               |  |
|      |TENSE     PRESENT                                  |  |
|      |ASPECT    PROGRESSIVE                              |  |
|      |PUNCT     |FORM      '.'           |               |  |
|ROOT  |          |STMT-TYPE DECLARATIVE   |               |  |
```

Figure 6.69. The functional structure for ‘Watashiwa jibunno shorainitsuite yoku kangaeteiru (I am often thinking about my future).’

The sentence (6.35) is an example in which a noun without a particle is used.

(6.35)

Watashi-wa	jibunno	shorai-nituite	mainichi
I-postp	*myself-postp*	*future-postp*	*every.day*

kangaeteiru.

think-progressive-present

‘I am thinking about my future every day.’

The dependency relationships among the syntactic units in the example sentence above are represented in the typed-dependency tree below. The dependency type “advmod” is used for the dependency between ‘mainichi (every day)’ and ‘kangaeteiru (be thinking about).’

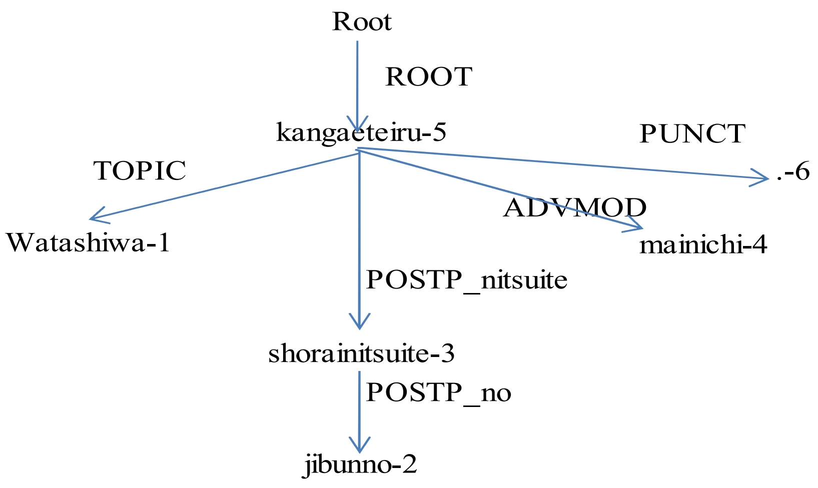


Figure 6.70. The typed-dependency tree for ‘Watashiwa jibunno shorainitsuite mainichi

kangaeteiru (I am thinking about my future every day).’

The typed-dependency tree above is equivalent to the functional structure below.

```
|          |PRED       'kangaeteiru<SUBJ, OBJ>'                      |  |
|          |SUBJ       |PRED        'PRO_i'       |                  |  |
|          |           |FORM        ZERO          |                  |  |
|          |                                                         |  |
|          |OBJ        |PRED        'PRO_j'       |                  |  |
|          |           |FORM        ZERO          |                  |  |
|          |                                                         |  |
|          |TOPIC      |PRED        'PRO_i'       |                  |  |
|          |           |FORM        'watashi'     |                  |  |
|          |           |PERSON      1ST           |                  |  |
|          |           |NUMBER      SINGULAR      |                  |  |
|          |                                                         |  |
|          |POSTP_     |PRED        'shorai'                    |    |  |
|          |nitsuite   |POSTP_no   |PRED ''PRO_i'     |          |    |  |
|          |           |           |TYPE REFLEXIVE    |          |    |  |
|          |           |           |FORM 'jibun'      |          |j   |  |
|          |                                                         |  |
|          |ADVMOD     |PRED        'mainichi'    |                  |  |
|          |TENSE       PRESENT                                      |  |
|          |ASPECT      PROGRESSIVE                                  |  |
|          |PUNCT      |FORM        '.'           |                  |  |
|ROOT      |           |STMT-TYPE   DECLARATIVE   |                  |  |
```

Figure 6.71. The functional structure for ‘Watashiwa jibunno shorainitsuite mainichi kangaeteiru (I am thinking about my future every day).’

The dependency type “advmod” in Japanese follows Mel’čuk’s Criterion A for SsyntRel because a verb and its adverb constitute a prosodic unit and have a fixed linear order in a sentence whereby the adverb precedes the verb. This dependency type also follows the revised Criterion A because a verb and its adverb constitute a semantic unit.

This dependency type follows Mel’čuk’s Criterion B because the verb determines the passive valence of the phrase. For example, the dependency relation between ‘kangaeteiru (…be thinking …)’ and ‘shorainitsuite (about future)’ in the sentence above shows that ‘shorainitsuite’ depends on the verb ‘kangaeteiru,’ which can be subordinated to the root node.

This dependency type also follows the revised Criterion C, because it implies a certain kind of semantic relationship between a verb and an adverb that cannot be expressed by any other dependency type. Moreover, the prototypical dependent of this dependency type is an adverb. However, this is not necessarily the case with adjectives in the adverbial form and particle-less nouns. Therefore, this dependency type follows the revised Criteria C1 and C2 proposed in Section 2.4.3.3.

6.5.5 Amod

The dependency type “amod” represents cases where the dependent is an adjective and the head is a noun. The following example illustrates this dependency type in Japanese.

(6.36)

Watashi-wa	huruhonya-ni	tsumaranai	hon-wo
I-postp	*second-hand.bookshop-postp*	*uninteresting*	*book-postp*
ut-ta.			
sell-past			

‘I sold uninteresting books to a second-hand bookshop.’

The dependency relationships among the syntactic units in the example sentence above are represented in the typed-dependency tree below. The dependency type “amod” is used for the dependency between the syntactic unit ‘tsumaranai (uninteresting)’ and ‘honwo (book).’

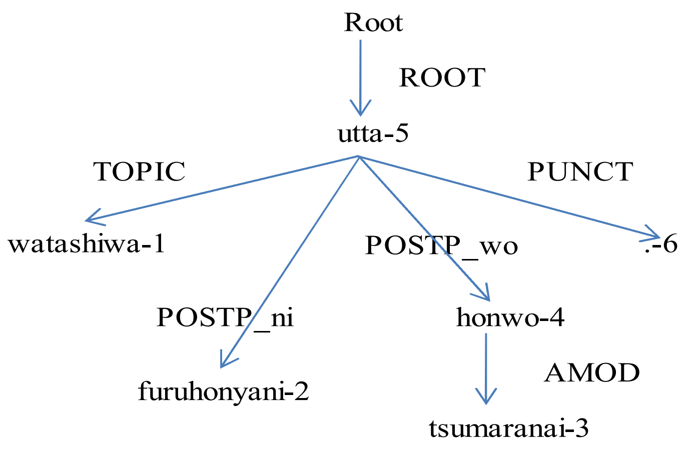


Figure 6.72. The typed-dependency tree for ‘Watashiwa furuhonyani tsumaranai honwo utta (I sold uninteresting books to a second-hand bookshop).’

The typed-dependency tree above is equivalent to the functional structure below.

```
     |  PRED        'utta<SUBJ,OBJ,OBJ2>'                                ||
     |  SUBJ       |PRED       'PRO_i'    |                              ||
     |             |TYPE       ZERO       |                              ||
     |                                                                   ||
     |  OBJ        |PRED       'PRO_j'    |                              ||
     |             |TYPE       ZERO       |                              ||
     |                                                                   ||
     |  OBJ2       |PRED       'PRO_k'    |                              ||
     |             |TYPE       ZERO       |                              ||
     |                                                                   ||
     |  TOPIC      |PRED       'PRO_i'    |                              ||
     |             |FORM       'watashi'  |                              ||
     |             |PERSON     1ST        |                              ||
     |             |NUMBER     SINGULAR   |                              ||
     |                                                                   ||
     |  POSTP_wo   |PRED       'hon'                          |          ||
     |             |AMOD      |PRED        'tsumaranai'   |   |j         ||
     |                                                                   ||
     |  POSTP_ni   |PRED       'furuhonya_k' |                           ||
     |                                                                   ||
     |  TENSE       PAST                                                 ||
     |  PUNCT      |FORM       '.'            |                          ||
|ROOT|             |STMT-TYPE  DECLARATIVE    |                          ||
```

Figure 6.73. The functional structure for ‘Watashiwa furuhonyani tsumaranai honwo utta (I sold

uninteresting books to a second-hand bookshop).’

The dependency type “amod” in Japanese follows Mel’čuk’s Criterion A for SsyntRel because a noun and an adjective modifying the verb constitute a prosodic unit, and they have a fixed linear order whereby the adjective precedes the noun on which it depends. This dependency type also follows the revised Criterion A because a noun and its adjective modifier constitute a semantic unit.

This dependency type follows Mel’čuk’s Criterion B. The noun, not the adjective, determines the passive valence of the phrase. For example, the dependency relation between ‘tsumaranai (uninteresting)’ and ‘honwo (books)’ in the sentence above shows that ‘tsumaranai’ depends on the noun ‘honwo,’ which can be subordinated to the verb ‘utta (sold).’

This dependency type also follows the revised Criterion C proposed in Section 2.4.3.3, because it implies a certain kind of semantic relationship between a noun and an adjective that cannot be expressed by any other dependency type. The prototypical dependent of this dependency type is an adjective.

6.5.6 Det

The dependency type “det” represents a dependency between a noun and a determiner. The noun is the head and the determiner is the dependent.

Japanese has several kinds of determiners. They do not inflect and can express a variety of meanings such as the speaker’s judgment about a noun or an interrogative adjective, as shown in the following example.

The sentence (6.37) is an example in which the determiner ‘konna’ indicates the speaker’s pejorative judgment about a noun.

(6.37)

Watashi-wa	konna	hon-wo	yoma-nai.
I-postp	*this.sort.of*	*book-postp*	*read-neg*

'I will not read this sort of books.'

The dependency relationship among the syntactic units in the example sentence above are represented in the typed-dependency tree below. The dependency type "det" is used for the dependency relationship between the syntactic unit 'konna' and the syntactic unit 'honwo.'

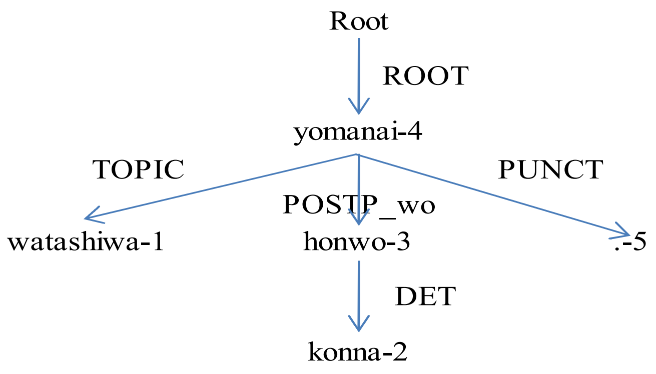


Figure 6.74. The typed-dependency tree for 'Watashiwa konna honwo yomanai (I will not read this sort of books).'

The typed-dependency tree above is equivalent to the functional structure below.

```
      PRED       'yomanai<SUBJ, OBJ>'
      SUBJ       |PRED      'PROi'
                 |FORM      ZERO

      OBJ        |PRED      'PROj'
                 |FORM      ZERO

      TOPIC      |PRED      'PROi'
                 |FORM      'watashi'
                 |PERSON    1ST
                 |NUMBER    SINGULAR

      POSTP_wo   |PRED      'hon'
                 |DET       |Form ''konna'
                 |          |TYPE PEJORATIVE        j
      TENSE      PAST
      NEG        +
      PUNCT      |FORM      '.'
ROOT             |STMT-TYPE DECLARATIVE
```

Figure 6.75. The functional structure for ‘Watashiwa konna honwo yomanai (I will not read this sort of books).’

The sentence (6.38) is an example in which the determiner ‘donna’ is an interrogative adjective:

(6.38)

Kimi-wa	ima-made-ni	donna	hon-wo	yomi-mashi-ta-ka?
you-topic	*now-postp-postp*	*det*	*book-postp*	*read-polite-past-end*

‘What kind of books have you ever read?’

The dependency relationships among the syntactic units in the example sentence above are represented in the typed-dependency tree below. The dependency type “det” is used for the dependency between ‘donna (what kind of)’ and ‘honwo (book).’

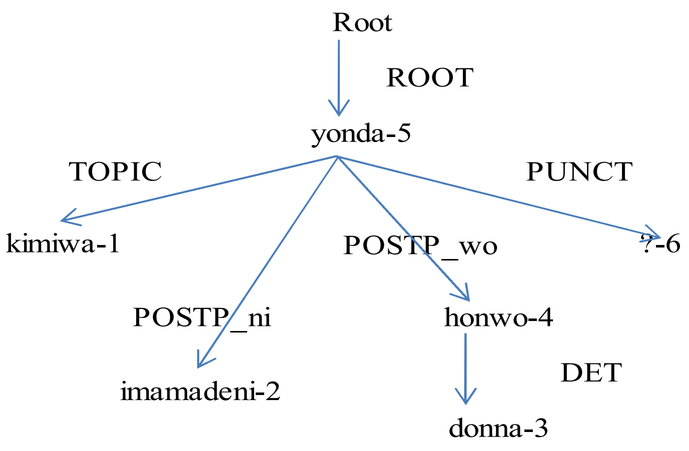


Figure 6.76. The typed-dependency tree for ‘Kimiwa imamadeni donna honwo yonda? (What kind of books have you ever read?)’

```
        | PRED       'yonda<SUBJ, OBJ>'                          ||
        | SUBJ       |PRED      'PRO_i'            |              ||
        |            |FORM      ZERO               |              ||
        |                                                         ||
        | OBJ        |PRED      'PRO_j'            |              ||
        |            |FORM      ZERO               |              ||
        |                                                         ||
        | TOPIC      |PRED      'PRO_i'            |              ||
        |            |FORM      'kimi'             |              ||
        |            |PERSON    2ND                |              ||
        |            |NUMBER    SINGULAR           |              ||
        |            |STYLE     INFORMAL           |              ||
        |                                                         ||
        | POSTP_wo   |PRED      'hon'                      |      ||
        |            |DET       |FORM ''donna'           | |      ||
        |            |          |TYPE INTERROGATIVE      | |j     ||
        |                                                         ||
        | POSTP_ni   |PRED      'imamade'          |              ||
        | TENSE      PAST                                         ||
        | PUNCT      |FORM      '?'                |              ||
| ROOT  |            |STMT-TYPE INTERROGATIVE      |              ||
```

Figure 6.77. The functional structure for ‘Kimiwa imamadeni donna honwo yomimashitaka?

(What kind of books have you ever read?)’

The dependency type “det” in Japanese follows Mel’čuk’s Criterion A for SsyntRel because a noun and its determiner constitute a prosodic unit and have a fixed linear order whereby the determiner precedes the noun on which it depends. This dependency type also follows the revised Criterion A because a noun and its determiner constitute a semantic unit.

This dependency type follows Mel’čuk’s Criterion B. The noun, not the determiner, determines the passive valence of the phrase. For example, the dependency relation between ‘donna (what kind of …)’ and ‘honwo (books)’ in the sentence above shows that ‘donna’ depends on the noun ‘honwo,’ which can be subordinated to the verb ‘yomimashitaka (did you read?).’

This dependency type also follows the revised Criterion C proposed in Section 2.4.3.3, because it implies a certain kind of semantic relationship between a noun and its determiner that cannot be expressed by any other dependency type. The prototypical dependent of this dependency type is a determiner.

6.5.7 Rcmod

The head of this dependency type is a noun and the dependent is the head of a relative clause modifying the noun, similar to the dependency type “rcmod” in Stanford Dependencies (see Section 5.3.4).

The sentence (6.39) is an example that contains a relative clause. The verb ‘nakunatta’ depends on the noun ‘hon,’ and the dependency type is “rcmod.”

(6.39)

Watashi-ga	kinou	kat-ta	hon-ga	nakunat-ta.
I-postp	*yesterday*	*buy-past*	*book-postp*	*be.lost-past*

‘The book I bought yesterday is lost.’

The dependency relationships among the syntactic units in the example sentence above are represented in the typed-dependency tree below. The dependency type “rcmod” is used for the dependency between ‘honga (book)’ and ‘nakunatta (lost).’

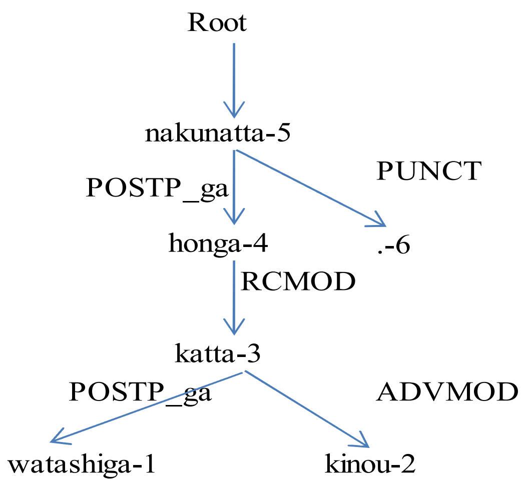


Figure 6.78. The typed-dependency tree for ‘Watashiga kinou katta honga nakunatta (The book I bought yesterday is lost).’

The typed-dependency tree above is equivalent to the functional structure below. Notice that the object zero pronoun of the verb ‘katta’ refers to the noun ‘hon.’ They are assigned with the same index ‘j;’ therefore, ‘hon’ functions as the object of the verb ‘katta.’ The subject zero pronoun of the verb ‘nakunatta’ refers to the local functional structure as the value of the attribute “postp_ga” of the same verb. They are assigned the same index ‘k;’ therefore, the phrase ‘watashiga katta hon’ functions as the subject of the verb ‘nakunatta.’

```
      PRED       'nakunatta<SUBJ>'
      SUBJ       PRED      'PROk'
                 FORM      ZERO

      POSTP_ga   PRED      'honj'
                 RCMOD     PRED       'katta<SUBJ,OBJ>'
                           SUBJ       PRED      'PROi'
                                      FORM      ZERO

                           OBJ        PRED      'PROj'
                                      FORM      ZERO

                           POSTP_ga   PRED      'PROi'
                                      FORM      'watashi'
                                      PERSON    1ST
                                      NUMBER    SINGULAR

                           ADVMOD     PRED      kinou'
                           TENSE      PAST                       k
      TENSE      PAST
      PUNCT      FORM      '.'
ROOT             STMT-TYPE DECLARATIVE
```

Figure 6.79. The functional structure for ‘Watashiga kinou katta honga nakunatta (The book I bought yesterday is lost).’

Japanese language has another type of relative clause in which the noun is not the argument of the predicate of the relative clause modifying the noun. Teramura (1992, p.202) called such relative clauses “*soto no kankei* (external relation).” The example sentence (6.40) contains an external relative clause. The noun ‘riyu (reason)’ is not the argument of the verb ‘yomanakatta.’

(6.40)

Sarah-ga sono hon-wo yoma-nakat-ta riyu-wa shira-nai.

Sarah-postp	det	book-postp	read-neg-past	reason-topic	know-neg

‘I don’t know the reason why Sarah did not read the book.’

The dependency relationships among the syntactic units in the example sentence above are represented in the typed-dependency below. The dependency type "rcmod" is used for the dependency between 'yomanakatta' and 'riyu.'

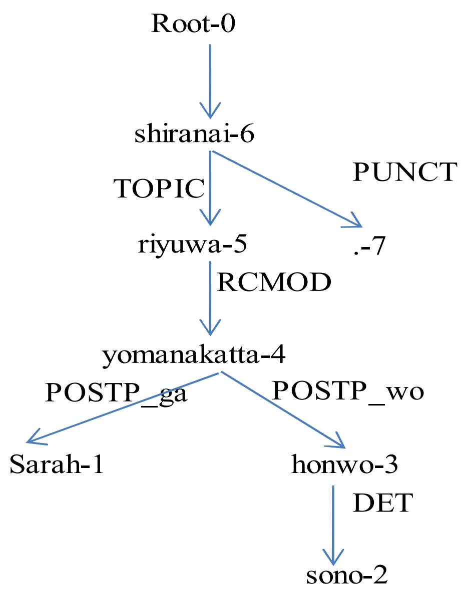
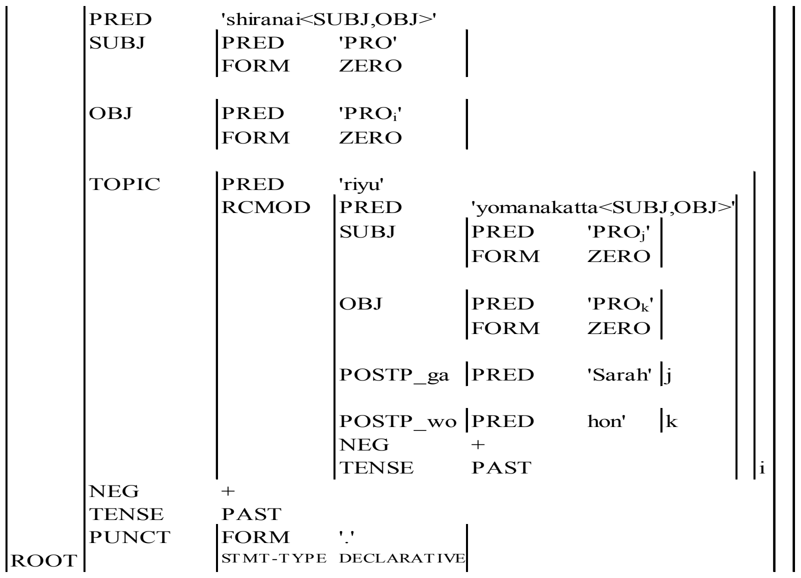


Figure 6.80. The typed-dependency tree for 'Sarahga sono honwo yomanakatta riyuwa shiranai.'

The typed-dependency tree above is equivalent to the functional structure below. The subject zero pronoun of the verb 'shiranai' refers to nothing within the sentence; it is conventionally interpreted to refer to the speaker of this sentence. The object zero pronoun of the verb 'shiranai' refers to the local functional structure which is the value of the attribute TOPIC of the verb 'shiranai' (this reference is indicated by the index 'i'). Neither the subject zero pronoun of the verb 'yomanakatta' nor the object zero pronoun of the same verb refer to the noun 'riyu.' Therefore, the noun 'riyu' in the functional structure below does not function as the subject or the object of the verb 'yomanakatta.'



PRED 'shiranai<SUBJ,OBJ>'
SUBJ PRED 'PRO' FORM ZERO
OBJ PRED 'PROi' FORM ZERO
TOPIC PRED 'riyu'
RCMOD PRED 'yomanakatta<SUBJ,OBJ>'
SUBJ PRED 'PROj' FORM ZERO
OBJ PRED 'PROk' FORM ZERO
POSTP_ga PRED 'Sarah' j
POSTP_wo PRED hon' k
NEG +
TENSE PAST
i
NEG +
TENSE PAST
PUNCT FORM '.' STMT-TYPE DECLARATIVE
ROOT

Figure 6.81. The functional structure for 'Sarahga sono honwo yomanakatta riyuwa shiranai.'

This dependency type follows Mel'čuk's Criterion A for SsyntRel because a noun and its relative-clause modifier constitute a prosodic unit (e.g., 'katta honga' in the example sentence above), and they have a fixed linear order whereby the relative-clause modifier precedes the noun on which it depends. In addition, they form a semantic unit in which the noun is the argument or adjunct of the verb in the relative clause. In this way, this type follows the revised Criterion A proposed in Section 2.4.3.1.

This dependency type follows Mel'čuk's Criterion B. The noun, not its relative-clause modifier, determines the passive valence of the phrase. For example, the dependency relation between 'honga' and 'katta' in the example sentence above shows that 'katta' depends on the noun 'honga,' which can be subordinated to the verb 'nakunatta.'

This dependency type also follows the revised Criterion C proposed in Section 2.4.3.3, because it implies a certain kind of semantic relationship between a noun and a relative-clause modifier. This semantic relationship cannot be expressed by any other dependency type, and the prototypical dependent of this dependency type is a verb.

6.5.8 Ccomp

The head of this dependency type is a verb and the dependent is the head of a clausal complement of the verb. In this type, “the grammatical function COMP labels a subordinate clause followed by a case particle ‘-to’ which is equivalent to an English complementizer ‘that’” (Oya 2010a, p.142). The sentence (6.40) illustrates this type in Japanese.

(6.40)

Kono	hon-wa	omoshiroi-to	ani-wa
this	*book-postp*	*interesting-ccomp*	*elder.brother-topic*
it-ta			
say-past			

‘My elder brother said that this book was interesting.’

The dependency relationships among the syntactic units in the example sentence above are represented in the following typed-dependency tree. The verbal syntactic unit ‘omoshiroito (that it is interesting)’ depends on ‘itta (said).’

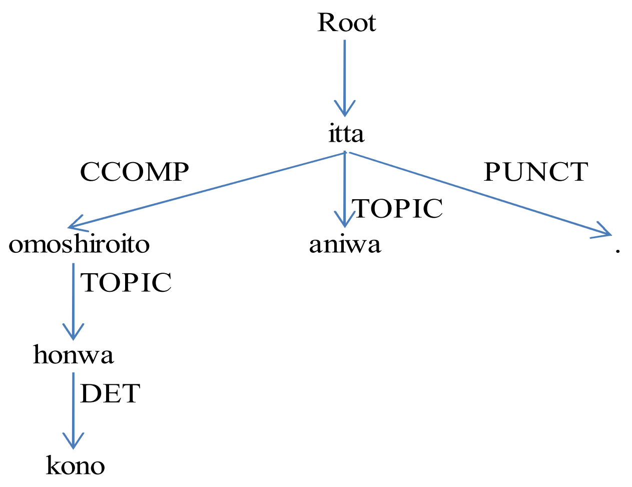


Figure 6.82. The typed-dependency tree for ‘Kono honwa omoshiroito aniwa itta (My elder brother said that this book was interesting).’

The typed-dependency tree above is equivalent to the functional structure below.

```
|      |PRED     'itta<SUBJ>'                                                      ||
|      |SUBJ     |PRED       'PRO$_i$'    |                                        ||
|      |         |FORM       ZERO         |                                        ||
|      |                                                                           ||
|      |TOPIC    |PRED       'ani'        |                                        ||
|      |         |GENDER     MASCULINE    |                                        ||
|      |         |NUMBER     SINGULAR     |i                                       ||
|      |                                                                           ||
|      |CCOMP    |PRED       'omoshiroi<SUBJ>'                                   | ||
|      |         |TOPIC     |PRED      'hon'                                  |  | ||
|      |         |          |DET       |FORM     'kono'               |      |  | ||
|      |         |          |          |TYPE     DEMONSTRATIVE        |      |j | ||
|      |         |                                                             | ||
|      |         |SUBJ      |PRED      'PRO$_j$'  |                            | ||
|      |         |          |FORM      ZERO       |                            | ||
|      |         |TENSE      PRESENT                                           | ||
|      |TENSE     PAST                                                           ||
|      |PUNCT    |FORM       '.'          |                                      ||
|ROOT  |         |STMT-TYPE  DECLARATIVE  |                                      ||
```

Figure 6.83. The functional structure for ‘Kono honwa omoshiroito aniwa itta (My elder brother said that this book was interesting).’

The dependency type “ccomp” follows Mel’čuk’s original Criterion A for SsyntRel because the dependent clause and the main verb have a fixed order whereby the dependent clause precedes the main verb, and they form a prosodic unit (e.g., ‘omoshiroito itta’ in the example above). In addition, they form a semantic unit and therefore this type follows the revised Criterion A proposed in Section 2.4.3.1.

This dependency type follows Mel’čuk’s Criterion B. In the example above, ‘omoshiroito’ depends on the verb ‘itta,’ not vice versa, because ‘itta’ depends on another element, namely, the root of this sentence.

This dependency type also follows the revised Criterion C proposed in Section 2.4.3.3, because it implies a certain kind of semantic relationship between verbs that cannot be expressed by any other dependency type, the prototypical dependent of this dependency type is a verb.

6.5.9 Advcl

The head of this dependency type is a verb and the dependent is the head of an adverbial clause modifying the verb. In this type, ‘sentential adjuncts or SADJ are the grammatical functions that are assigned to verbal units whose heads have inflections other than the base form or the –ta form’ (Oya 2010a, p.150). This study uses the term *advcl*, not SADJ used in Oya (2010a), in order to ensure the parallelism between the Stanford Parser output for English sentences and the KNP output for Japanese sentences. The sentence (6.48) below illustrates this type in which the verb ends with the inflection ‘-te.’ This inflection indicates the sequence of events. In the example below, the event of listening to the elder brother’s talk precedes the change in the speaker’s life.

(6.41)

Ani-no	hanashi-wo	kii-te	watashino
elder.brother-postp	*talk-postp*	*listen-infl*	*I-postp*

jinsei-wa	kawat-ta
life-topic	*change-past*

‘After listening to my elder brother’s talk, my life changed.’

The dependency relationships among the syntactic units in the example sentence above are represented in the following typed-dependency tree. The dependency type “advcl” is used for the dependency relationship between the verbal syntactic unit ‘kiite (after listening to)’ and ‘kawatta (changed).’

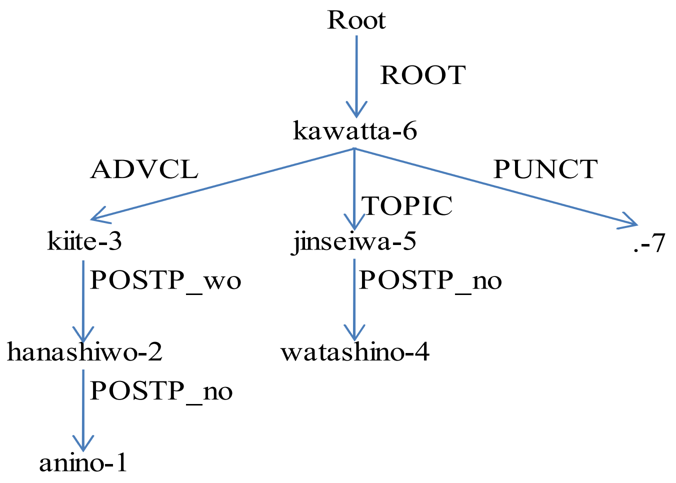


Figure 6.84. The typed-dependency tree for ‘Anino hanashiwo kiite watashino jinseiwa kawatta (After listening to my elder brother’s talk, my life changed).’

The typed-dependency tree above is equivalent to the following functional structure.

```
      | PRED      'kawatta<SUBJ>'                                        ||
      | SUBJ     |PRED      'PRO_i'   |                                  ||
      |          |FORM      ZERO      |                                  ||
      |                                                                  ||
      | TOPIC    |PRED      'jinsei'                     |               ||
      |          |POSTP_no |PRED       'PRO'        |    |               ||
      |          |         |PERSON     1ST          |    |               ||
      |          |         |NUMBER     SINGULAR     |    |               ||
      |          |         |FORM       'watashi'    |    |i              ||
      |                                                                  ||
      | ADVCL    |PRED      'kiite<SUBJ,OBJ>'                            ||
      |          |                                                       ||
      |          |SUBJ     |PRED       'PRO'        |                    ||
      |          |         |FORM       ZERO         |                    ||
      |          |                                                       ||
      |          |OBJ      |PRED       'PRO_j'      |                    ||
      |          |         |FORM       ZERO         |                    ||
      |          |                                                       ||
      |          |POSTP_wo |PRED       hanashi'                     |    ||
      |          |         |POSTP_no  |PRED      'ani'         |    |    ||
      |          |         |          |PERSON    3RD           |    |    ||
      |          |         |          |NUMBER    SINGULAR      |    |    ||
      |          |         |          |GENDER    MASCULINE     |    |j   ||
      | TENSE     PAST                                               ||
      | PUNCT    |FORM      '.'          |                           ||
ROOT  |          |STMT-TYPE DECLARATIVE  |                           ||
```

Figure 6.85. The functional structure for 'Anino hanashiwo kiite watashino jinseiwa kawatta (After listening to my elder brother's talk, my life changed).'

Different verbal inflections of the sentential adverbials indicate different semantic relationships between the main predicate and the sentential adverbials. For example, the verbal inflection '-nagara' in the sentence (6.42) below indicates that the event described by the main predicate and the event described by the sentential adverbial occurred at the same time.

(6.42)

Ani-no	hanashi-wo	kiki-nagara	watashi-wa
elder.brother-postp	*talk-postp*	*listen-infl*	*I-topic*

hon-wo	yonde-ita
book-postp	*read-past*

'I was reading a book while listening to my elder brother's talk.'

The dependency relationships among the syntactic units in the example sentence above are represented in the following typed-dependency tree. The dependency type "advcl" is used for the dependency relationship between the verbal syntactic unit 'kikinagara (while listening to)' and 'yondeita (was reading).'

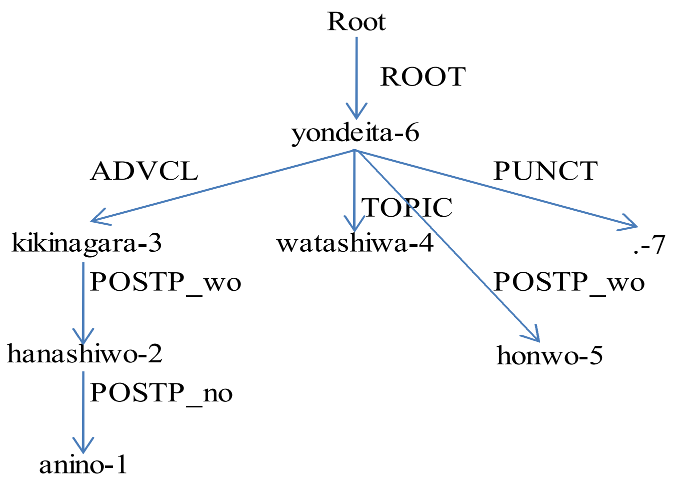


Figure 6.86. The typed-dependency tree for 'Anino hanashiwo kikinagara, watashiwa honwo yondeita (I was reading a book while listening to my elder brother's talk).'

```
     |PRED       'yondeita<SUBJ,OBJ>'                                          ||
     |SUBJ       |PRED      'PRO_i'     |                                      ||
     |           |FORM      ZERO        |                                      ||
     |                                                                         ||
     |SUBJ       |PRED      'PRO_j'     |                                      ||
     |           |FORM      ZERO        |                                      ||
     |                                                                         ||
     |TOPIC      |PRED      'PRO'       |                                      ||
     |           |PERSON    1ST         |                                      ||
     |           |NUMBER    SINGULAR    |                                      ||
     |           |FORM      'watashi'   |i                                     ||
     |                                                                         ||
     |POSTP_wo   |PRED      'hon'       |j                                     ||
     |                                                                         ||
     |ADVCL      |PRED      'kikinagara<SUBJ,OBJ>'                             ||
     |           |SUBJ      |PRED      'PRO'     |                             ||
     |           |          |FORM      ZERO      |                             ||
     |           |                                                             ||
     |           |OBJ       |PRED      'PRO_k'   |                             ||
     |           |          |FORM      ZERO      |                             ||
     |           |                                                             ||
     |           |POSTP_wo  |PRED      hanashi'                       |        ||
     |           |          |POSTP_no  |PRED      'ani'        |      |        ||
     |           |          |          |PERSON    3RD          |      |        ||
     |           |          |          |NUMBER    SINGULAR     |      |        ||
     |           |          |          |GENDER    MASCULINE    |      |k       ||
     |TENSE      PAST                                                          ||
     |ASPECT     PROGRESSIVE                                                   ||
     |PUNCT      |FORM      '.'         |                                      ||
ROOT |           |STMT-TYPE DECLARATIVE |                                      ||
```

Figure 6.87. The functional structure for ‘Anino hanashiwo kikinagara, watashiwa honwo yondeita (I was reading a book while listening to my elder brother’s talk).’

This dependency type follows Mel’čuk’s original Criterion A for SsyntRel because the dependent clause and the main verb have a fixed word order whereby the dependent clause precedes the main verb, and they form a prosodic unit. In addition, they form a semantic unit, thus following the revised Criterion A proposed in Section 2.4.3.1.

This dependency type follows Mel’čuk’s Criterion B. In the example sentence above, ‘kiite’ depends on ‘kawatta,’ not vice versa, because ‘kawatta’ depends on another element, namely, the root of this sentence.

This dependency type also follows the revised Criterion C proposed in Section 2.4.3.3, because it implies a certain kind of semantic relationship between verbs that cannot be expressed by any other dependency type, and the prototypical dependent of this dependency type is a verb.

6.5.10 Treatment of Coordinates

Unlike Stanford Dependency, the dependency-type inventory for Japanese in this dissertation does not contain any dependency type for coordinates. This is due to the fact that coordinates are indicated by certain types of postpositions such as '-to' or '-ya,' hence coordinates are treated as the dependents of the dependency type "postp." This treatment is also intended to highlight the *symmetric* structure of coordinates, as described later in this section.

Oya (2010a, p.151) stated that "… the coordinates must depend on one 'dummy' syntactic unit which has the grammatical function of the last coordinate." In addition, the dependency relation between the dummy and each coordinate is called "coord". An example of this type is presented below.

(6.43)

Watashi-wa	gengogaku-to	jinruigaku-wo	manan-da.
I-postp	*linguistics-postp*	*anthropology-postp*	*read-past*

'I studied linguistics and anthropology.'

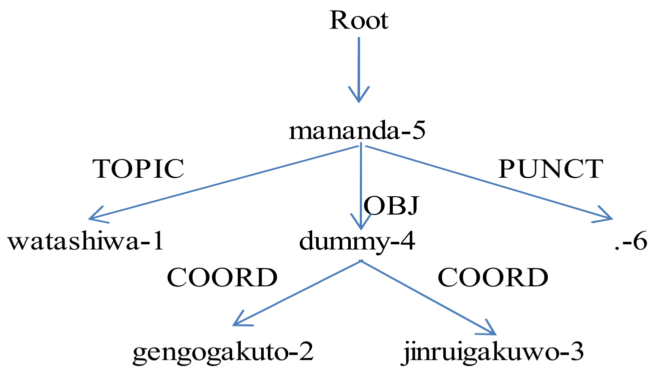


Figure 6.88. The typed-dependency tree for 'Watashiwa gengogakuto jinruigakuwo mananda (I studied linguistics and anthropology)' with a dummy syntactic unit for coordinates.

The use of dummy syntactic units was a measure taken in Oya (2010a) to transform the *asymmetric* coordinates in the output of KNP into *symmetric* ones. However, the presence of a dummy in a typed-dependency tree is an ad-hoc measure and not linguistically motivated. Therefore, this study discards the dummy syntactic unit for coordinate construction, and all the coordinates are treated as being dependent on one head, and the dependency type is "postp." The dependency type is further subtyped according to the postposition used, as shown in the figure below.

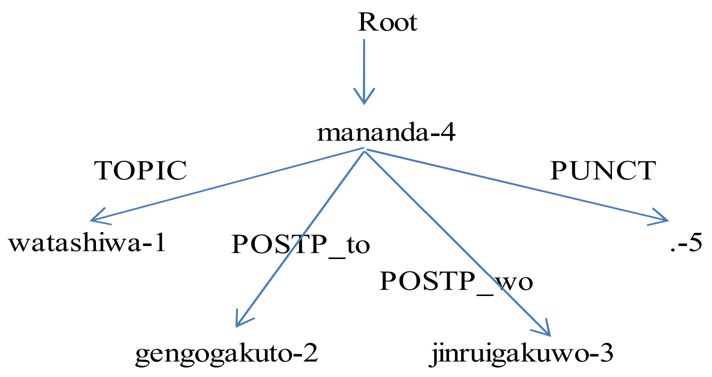


Figure 6.89. The typed-dependency tree for 'Watashiwa gengogakuto jinruigakuwo mananda (I studied linguistics and anthropology)' without a dummy syntactic unit for coordinates.

The typed-dependency tree above is equivalent to the functional structure below. The object zero pronoun of the verb 'mananda (*someone studied something*)' refers to both of the syntactic units 'gengogakuto (*linguistics and*)' and 'jinruigakuwo (*anthropology*).'

```
|     |PRED       'mananda<SUBJ,OBJ>'                  | | |
|     |SUBJ      |PRED      'PRO$_i$'      |            | | |
|     |          |TYPE      ZERO          |            | | |
|     |                                               | | |
|     |OBJ       |PRED      'PROj'        |            | | |
|     |          |TYPE      ZERO          |            | | |
|     |                                               | | |
|     |TOPIC     |PRED      'PRO$_i$'      |            | | |
|     |          |PERSON    1ST           |            | | |
|     |          |NUMBER    SINGULAR      |            | | |
|     |          |FORM      'watashi      |            | | |
|     |                                               | | |
|     |POSTP_to  |PRED      'gengogaku' |j             | | |
|     |                                               | | |
|     |POSTP_wo  |PRED      'jinruigaku' |j            | | |
|     |TENSE      PAST                                | | |
|     |PUNCT     |FORM      '.'                     | | | |
|ROOT |          |STMT-TYPE DECLARATIVE             | | | |
```

Figure 6.90. The functional structure for 'Watashiwa gengogakuto jinruigakuwo mananda (I studied linguistics and anthropology)' without a dummy syntactic unit for coordinates.

6.6 Summary

This chapter introduced KNP (Kurohashi & Nagao 1992, 1994, 1998; Kawahara & Kurohashi 2007), which is a rule-based dependency parser used for generating automatic typed-dependency tree representations for Japanese sentences. Section 6.2 briefly introduced KNP and its output format. Section 6.3 described the process through which KNP parsed output is annotated with dependency types. Section 6.4 dealt with zero pronouns in elliptic sentences often used in Japanese, and it is argued that Japanese verbs contain zero pronouns in its lexical entry so that they can stand as one-word sentences by themselves. In section 6.5, each dependency type used with this parser in this study was defined with reference to the criteria for surface syntactic relations by Mel'čuk (2009, 2011), along with example sentences for each of the dependency types, their typed dependency trees, and the functional structure representations equivalent to these trees. By doing this, each of the Japanese dependency types proposed in this section is given a theoretical backbone based on Mel’čuk’s Criteria, and the parse output of KNP annotated with dependency types is shown to be equivalent to functional-structure representation in the framework of LFG.

7. Data analyses

7.1 Introduction

This chapter attempts to answer the following question: *from which source are the graph centrality measures obtained, and what is the result?* In this section, the meaning of the word "result" here is further defined as follows: (1) the difference between the centrality measures of the parsed output of sentences and those of the correct typed-dependency trees for the same sentences, (2) the difference between the centrality measures of the parsed output for English sentences and those of the parsed output for Japanese counterparts, and (3) the difference among centrality measures of the parsed output for English sentences in different genres of texts.

These results have relevance to different issues; (1) is relevant to the accuracy of the parsers used in this study, (2) is relevant to the issue of how centrality measures capture cross-linguistic differences in terms of syntactic dependency structure, and (3) is relevant to the issue of how centrality measures capture intra-linguistic variations of syntactic dependency structure.

First, Section 7.3 addresses the issue of parsing accuracy for English sentences. The accuracy of Stanford Parser is examined by comparing the typed-dependency trees which are automatically obtained from the parsed output of the English sentences to their manually-corrected typed-dependency trees, and it is shown that the distributions of both degree centralities and closeness centralities before and after manual corrections are almost identical. Thus, Stanford Parser is found to be sufficiently accurate to obtain degree centralities and closeness centralities of English sentences.

Second, Section 7.4 addresses the issue of parsing accuracy for Japanese sentences. The accuracy of KNP is examined by comparing the typed-dependency trees which are automatically obtained from the parsed output of the Japanese sentences to their manually-corrected typed-dependency trees, and similarly to the result of parsing accuracy of English sentences using the Stanford Parser, the distributions of both degree centralities and closeness centralities before and after manual corrections are almost identical. Thus, KNP is found to be sufficiently accurate to obtain degree centralities and closeness centralities of Japanese sentences.

Third, section 7.5 addresses the issue of cross-linguistic differences of centrality measures. The distributions of degree centralities and of closeness centralities obtained from English typed-dependency trees are compared to those obtained from the Japanese counterparts, and it is shown that their distributions are different. Thus, the structural properties of the typed-dependency trees for the sentences in these two languages are different in terms of their degree centralities (flatness) and closeness centralities (embeddedness).

Lastly in section 7.6, the distributions of degree centralities and of closeness centralities obtained from the parsed output of sentences from different genres texts in Manually annotated sub-corpus of American National Corpus (MASC 500k) (Ide, Baker, Fellbaum, Fillmore, & Passonnau 2008) are compared to each other, and it is shown that their distributions are different; however, it is pointed out that these different distributions are dependent on the word counts of the sentences.

7.2 Features Extractable from Typed-Dependency Trees

In principle, a number of features can be extracted from the typed-dependency tree for a sentence.

1. A dependency relation between two words provides us with information about which word depends on which word. A dependency relation is a directed edge (or arc) from the head word to its tail word.
2. A dependency type provides us with information on the category to which the dependency

relation belongs. A dependency type is the label assigned to a dependency relation. This is equivalent to the term "grammatical function" used in other syntactic theories such as Lexical-Functional Grammar (LFG) (Kaplan & Bresnan 1982; Bresnan 2001).

3. Embeddedness of a typed-dependency tree for a sentence provides us with information on how embedded the dependency structure of the sentence is. Embeddedness of a typed-dependency tree can be represented as the *closeness centrality* (Freeman 1979) of the root node of the tree.
4. Flatness of a typed-dependency tree for a sentence provides us with information on how flat the dependency structure of the sentence is, that is, to what extent the dependency structure is concentrated on one particular node. Flatness of a typed-dependency tree can be represented as the *degree centrality* (Freeman 1979) of the entire tree.
5. Dependency distance of a dependency relation provides us with information on how many words a given tail is away from its head. Dependency relations of different dependency types are expected to have different dependency distances on average.

7.3 Parsing Accuracy of the Stanford Parser

This section deals with the parsing accuracy of the Stanford Parser (de Marneffe & Manning 2012), and the effect of parse errors to the centrality measures and dependency distances discussed above. First, the two types of dependency errors are defined. Then, the issue of parsing errors is briefly introduced.

7.3.1 Two types of dependency parse errors

There are two types of dependency parse errors in the output of the Stanford Parser:

dependency-relation errors and *dependency-type* errors. Dependency-relation errors are those in which the head-tail relationship between words is incorrectly parsed. Dependency-type errors are those in which the head-tail relationship between words is correctly parsed, but typed incorrectly. For example, consider the typed-dependency tree for the sentence ‘Sarah has written this book.’

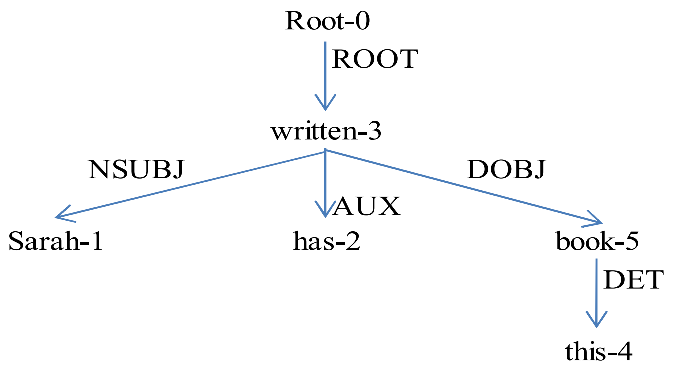


Figure 7.1. The typed-dependency tree for ‘Sarah has written this book.’

The following typed-dependency tree contains an incorrect dependency type between ‘book’ and ‘this.’ The dependency relation between these words is correct, while the dependency type is incorrect. The output states that the dependency type between ‘book’ and ‘this’ is “amod” (adjectival modification), but it must be “det” (determiner).

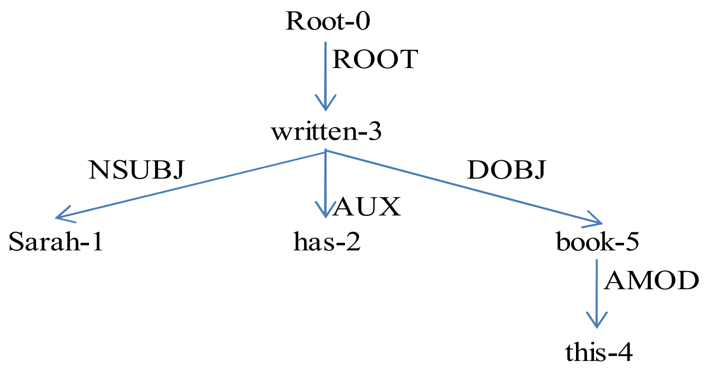


Figure 7.2. A typed-dependency tree for ‘Sarah has written this book’ with an incorrect dependency type between ‘book’ and ‘this.’

The following typed-dependency tree contains an incorrect dependency relation between ‘has’ and ‘this’; the dependency type “det” for the word ‘this’ is correct, but the head of the dependency is incorrect:

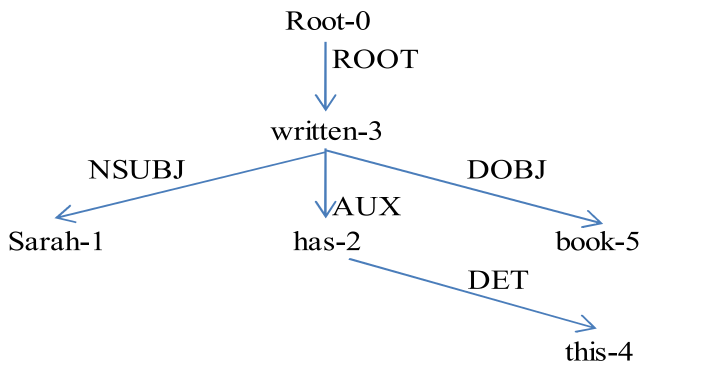


Figure 7.3. A typed-dependency tree for ‘Sarah has written this book’ with an incorrect dependency relation between ‘has’ and ‘this.’

Notice that the correctness of a dependency type is assumed to be determined with respect to the *tail* of the dependency.

Both of these types of parse errors can co-occur for the same word in a sentence. For example, the following typed-dependency tree is incorrect in terms of dependency relation and dependency type for the word ‘this.’ The tree shows that ‘this’ depends incorrectly on ‘has’ with an incorrect dependency type “amod”.

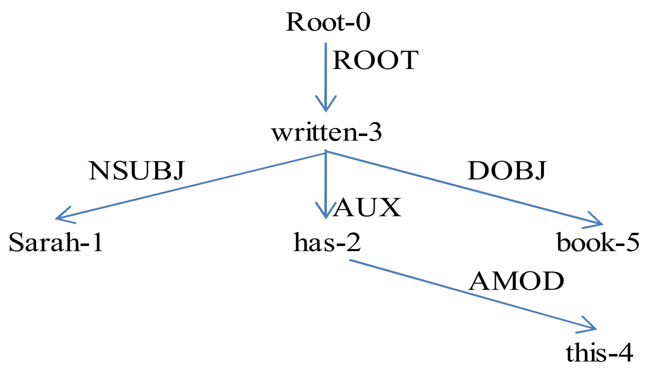


Figure 7.4. A typed-dependency tree for ‘Sarah has written this book’ with an incorrect dependency type and an incorrect dependency relationship

These errors shown above are constructed for the clarity of explanation of dependency-error types, although the parser does not yield such obvious errors.

There has been a significant amount of research on improving the accuracy of parsing. To name a few, Charniak (2000) uses the maximum-entropy model for calculating the most probable parse tree for a given sentence; Collins (1996) describes a statistical parser based on probabilities of dependencies between pairs of two words; Zeman & Žaborkrtský (2005) combine different parsers to improve parsing accuracy. The assumption shared by these researchers is that, if we can fix incorrect dependency analyses in the parsed output as much as possible, data processing based on the output will become more reliable. However, it is rather difficult to detect all the incorrect analyses in the original parsed output and fix all of them; no study has succeeded in achieving 100% accuracy in parsing. The Stanford Parser is one of the state-of-the-art dependency parsers available at present, but like the other parsers, it cannot yield a 100% correct dependency analysis for the sentences in a given text.

In this context, it is desirable to focus on the extent to which incorrect dependency analyses may affect on the use of the parsed output for different purposes. If the aim of dependency parsing is to create a best parser ever, all the dependency analysis errors must be eradicated regardless of the type of the errors. If, on the other hand, the aim of dependency parsing is not just obtaining the parsed output, but also extracting certain information from the parsed output, e.g., the parsed output's structural properties such as the centrality measures introduced above, it is desirable to show how significantly dependency analysis errors affect the structural properties calculated. In particular, if a parser correctly analyses the dependency relation between two words in a sentence, but it yields an error in terms of the dependency type given to this dependency relation, the degree and closeness centralities are not affected by this error because these centrality measures ignore the information about dependency types.

This claim does not diminish the importance of improving the accuracy of parser output. It is possible to detect the parse errors found in the parsed output of the sentences in a small corpus, so that we can obtain information about the accuracy of a parser, which can contribute to the work of parser developers. The small corpus must contain a variety of syntactic patterns, so that we can obtain balanced data. By parsing the sentences in the corpus, it is expected that we can identify syntactic patterns in which parse errors are found more often than in other syntactic patterns.

From this standpoint, in this study the accuracy of the Stanford Parser is examined by searching for incorrect dependency relations and types manually in the parsed output for a small corpus. Then, the degree and closeness centralities are calculated, both from the parser output before manual correction and from that after manual correction. The results are compared statistically in order to see how dependency parse errors affect the degree and closeness centralities of the parsed output.

7.3.2 Data description

The data chosen for the purpose introduced above are taken from "*Eisakubun Kihon 300 Sen*" (Basic 300 Sentences for English Composition; henceforth *Basic 300*) (Iida 2010). Basic 300 contains 339 English sentences along with their Japanese translations, and it is compiled for Japanese high school students to memorize basic syntactic structures of English; therefore, it contains important syntactic constructions of English. These data are also used in the calculation for degree and closeness centralities of English and Japanese sentences in Section 7.5.

7.3.3 Procedure

First, the English sentences in Basic 300 are parsed by the Stanford Parser ver.1.6.9[53]. The output option is set to Collapsed Tree (see Section 5.4.4) in order to keep the parallelism between English prepositional phrases and Japanese postpositional phrases. Second, the parsed output for each sentence is checked in terms of the dependency relation and dependency type. The parsed output file is converted by an original Ruby script (Appendix IV) into .net files for *Pajek* (Bategelj & Mrvar 1996), an application used for network analysis. *Pajek* shows us the typed-dependency tree for a sentence. For example, the .net file format of the

[53] The latest version of the Stanford Parser as of September 2013 is Version 3.2 (de Marneffe & Manning 2013).

typed-dependency tree for “Sarah has written this book” is as follows:

(7.1)

```
*Vertices 7
1 “ROOT”
2 “Sarah”
3 “have”
4 “write”
5 “this”
6 “book”
7 “.”
*Arcs
4 2 1 l “nsubj”
4 3 1 l “aux”
1 4 1 l “root”
6 5 1 l “det”
4 6 1 l “dobj”
4 7 1 l “punct”
```

Pajek reads a .net file in the format above, and outputs a graph, as shown in Figure 7.5. In this output, the auxiliary ‘has’ and the verb ‘written’ are shown in their lemmas.

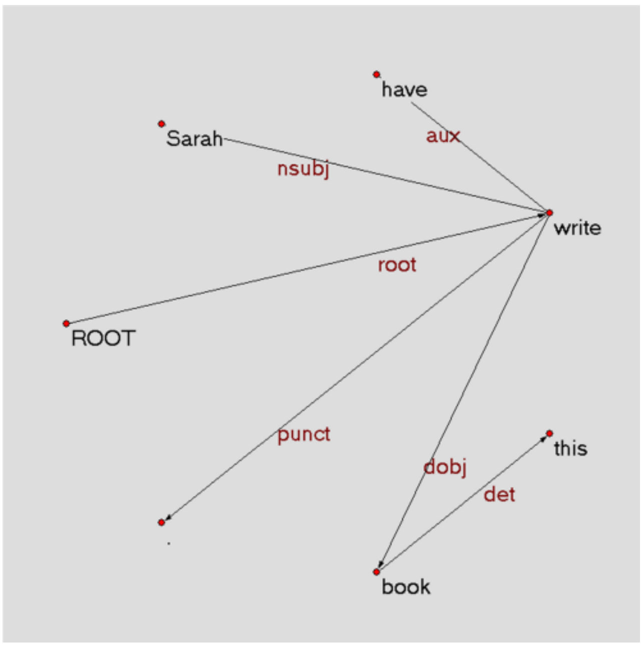


Figure 7.5. The typed-dependency tree for “Sarah has written this book” in Pajek

In this study, Pajek is used for the manual correction of parsed output because it is much easier to detect incorrect dependency errors in the form of Pajek-style typed-dependency trees than in the form of Stanford-Parser-style triples.

If any of the dependency relations and types in the output file is found incorrect, they are manually corrected. For example, (7.2) is the .net file for ‘Sarah has written this book’ with an incorrect dependency type, and Figure 7.6 shows the incorrect typed-dependency tree. The dependency type “amod” between “this” and “book” is incorrect. It should be corrected to “det.”

(7.2)

*Vertices 7

1 “ROOT”

2 “Sarah”

3 “have”

4 “write”

5 “this”

6 “book”

7 “.”

*Arcs

4 2 1 1 “nsubj”

4 3 1 1 “aux”

1 4 1 1 “root”

6 5 1 1 “*amod*”

4 6 1 1 “dobj”

4 7 1 1 “punct”

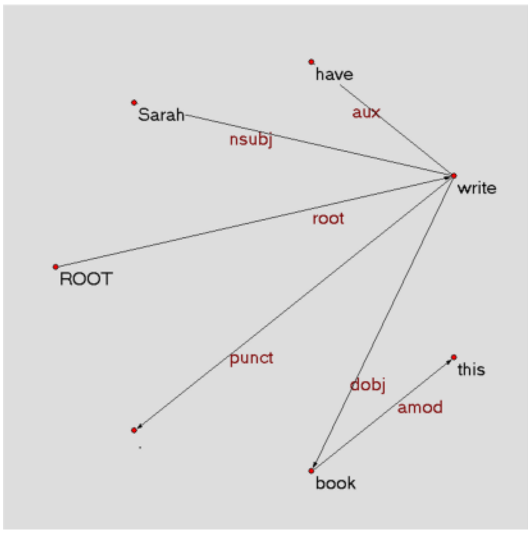


Figure 7.6. An incorrect typed-dependency tree for “Sarah has written this book” in Pajek

Incorrect dependency types and failed dependency types are counted as follows. Suppose that the typed-dependency tree below is a parsed output for an example sentence “Sarah has read this." before manual correction.[54] The dependency relationship between 'read' and 'this' is typed incorrectly as "det;" hence, this "det" is counted as an incorrect dependency type.

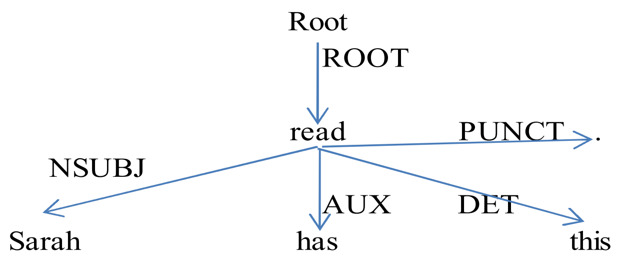


Figure 7.7. The typed-dependency tree for "Sarah has read this." (before manual correction)

The typed-dependency tree after manual correction of the tree above is shown below. The incorrect dependency type "det" is replaced by a correct type "dobj." This "dobj" has been *failed* to be parsed by the parser; hence, this "dobj" is counted as a failed dependency type.

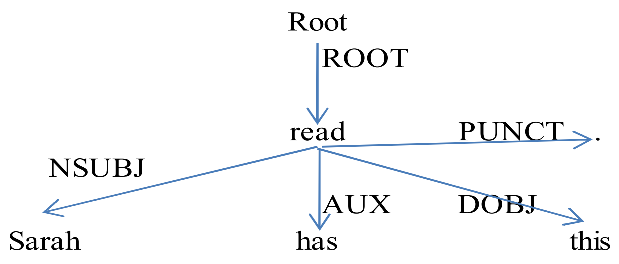


Figure 7.8. The typed-dependency tree for "Sarah has read this." (after manual correction)

The precision, recall and f-score of each dependency type are calculated by a Ruby script

[54] This parsed error is constructed for the clarity of explanation of incorrect and failed types, and the parser does not yield such an obvious error.

(see Appendix VI), in order to see which dependency type often fails to be correctly parsed. The precision, recall, and f-score are calculated according to Manning & Schütze (1999, p.268-269). The precision of a dependency type P_t is calculated by the following formula, where cp_t means the number of correctly parsed instances of a dependency type t, and ip_t means the number of incorrectly parsed instances of the dependency type t.

$$P_t = \frac{cp_t}{cp_t + ip_t} \qquad (1)$$

(Manning & Schütze 1999, p.268)

The recall of a dependency type R_t is calculated by the following formula, where cp_t means the number of correctly parsed instances of a dependency type t, and fp_t means the number of instances of the dependency type t which failed to be correctly parsed.

$$R_t = \frac{cp_t}{cp_t + fp_t} \qquad (2)$$

(Manning & Schütze 1999, p.269)

The f-score of a dependency type F_t is the harmonic mean of the precision and the recall of the dependency type, calculated by the following formula.

$$F_t = \frac{2R_tP_t}{(R_t + P_t)} \qquad (3)$$

(Manning & Schütze 1999, p.269)

The centrality measures of both the parsed output *before* and *after* manual correction are also calculated by a Ruby script (see Appendix VII), then compared with each other.

7.3.4 Results

The list of English sentences with incorrect parses is presented in Appendix II. Of all the English sentences in Basic300 (339 in total), the number of sentences which contain at least one wrong dependency is 120. This is around 35.39% of all the English sentences in Basic300. Of all the dependencies in the English sentences in Basic300 (3405 in total), the number of correctly parsed dependencies is 3161; hence, the parsing accuracy of the Stanford Parser is more than 92% for Basic300.

Of all the incorrect dependencies, the number of dependencies that are parsed with an incorrect relation and the correct type is 39 (about 1% of all the dependencies). The number of dependencies that are parsed with an incorrect type and the correct relation is 94 (about 2.7% of all the dependencies). The number of dependencies that are parsed with both an incorrect relation and an incorrect type is 107 (about 3.1% of all the dependencies).

The recall, precision, and f-score of each dependency type in English are shown in Table 7.1. In the table, the column 'C' shows the number of each dependency type which is correct in the parsed output (cp_t in the formula (1) and (2)). The column 'I' shows the number of each dependency type which is incorrect in the parsed output (ip_t in the formula (1)). The column 'F' shows the number of each dependency type which failed to be parsed, and added to the manually corrected parsed output (fp_t in the formula (2)).

Table 7.1. The precision, recall, and f-score of each dependency type in the English sentences in *Basic300*

type	C	I	FL	P	R	F
acomp	12	0	1	1.000	0.923	0.960
acomp_and	1	0	0	1.000	1.000	1.000
advcl	74	2	27	0.974	0.733	0.836
advcl_and	0	0	1	0.000	0.000	0.000
advmod	197	6	26	0.970	0.883	0.925
amod	140	3	5	0.979	0.966	0.972
appos	2	1	0	0.667	1.000	0.800
aux	294	0	4	1.000	0.987	0.993
auxpass	28	0	1	1.000	0.966	0.982
cc	1	0	1	1.000	0.500	0.667
ccomp	70	16	19	0.814	0.787	0.800
ccomp_but	1	0	0	1.000	1.000	1.000
ccomp_or	0	0	1	0.000	0.000	0.000
complm	22	2	1	0.917	0.957	0.936
conj	0	0	2	0.000	0.000	0.000
cop	101	1	4	0.990	0.962	0.976
csubj	5	1	0	0.833	1.000	0.909
det	299	4	7	0.987	0.977	0.982
dobj	218	14	26	0.940	0.893	0.916
dobj_and	2	0	0	1.000	1.000	1.000
expl	9	0	0	1.000	1.000	1.000
infmod	5	1	5	0.833	0.500	0.625
iobj	4	0	4	1.000	0.500	0.667
mark	64	3	20	0.955	0.762	0.848
mwe	2	0	0	1.000	1.000	1.000
neg	94	1	6	0.989	0.940	0.964
neg_or	0	0	1	0.000	0.000	0.000
nn	26	7	0	0.788	1.000	0.881
nn_and	1	1	0	0.500	1.000	0.667
npadvmod	8	0	0	1.000	1.000	1.000
nsubj	535	23	12	0.959	0.978	0.968
nsubj_and	2	0	0	1.000	1.000	1.000
nsubj_or	1	0	0	1.000	1.000	1.000
nsubjpass	24	0	0	1.000	1.000	1.000
num	24	2	0	0.923	1.000	0.960
num_and	0	1	1	0.000	0.000	0.000
number	1	1	1	0.500	0.500	0.500
parataxis	5	1	0	0.833	1.000	0.909
partmod	10	3	3	0.769	0.769	0.769
pobj	4	3	0	0.571	1.000	0.727
poss	90	2	3	0.978	0.968	0.973
possessive	0	0	1	0.000	0.000	0.000
predet	9	0	2	1.000	0.818	0.900
prep	7	3	0	0.700	1.000	0.824
prep_about	2	1	1	0.667	0.667	0.667
prep_across	2	0	0	1.000	1.000	1.000
prep_after	3	0	0	1.000	1.000	1.000
prep_around	1	0	0	1.000	1.000	1.000
prep_as	3	1	0	0.750	1.000	0.857
prep_at	9	0	3	1.000	0.750	0.857
prep_before	1	0	0	1.000	1.000	1.000
prep_behind	0	1	1	0.000	0.000	0.000
prep_besides	1	0	0	1.000	1.000	1.000
prep_between	1	0	0	1.000	1.000	1.000
prep_but	0	0	1	0.000	0.000	0.000
prep_by	5	1	3	0.833	0.625	0.714
prep_except_for	0	0	1	0.000	0.000	0.000
prep_for	22	2	3	0.917	0.880	0.898
prep_from	5	0	0	1.000	1.000	1.000
prep_in	33	4	9	0.892	0.786	0.835
prep_into	4	0	0	1.000	1.000	1.000
prep_like	5	1	0	0.833	1.000	0.909
prep_next_to	1	0	0	1.000	1.000	1.000
prep_of	45	0	0	1.000	1.000	1.000
prep_on	18	3	3	0.857	0.857	0.857
prep_on_or	1	0	0	1.000	1.000	1.000
prep_out_of	1	0	0	1.000	1.000	1.000
prep_over	1	0	0	1.000	1.000	1.000
prep_since	1	0	0	1.000	1.000	1.000
prep_than	2	0	0	1.000	1.000	1.000
prep_through	1	0	0	1.000	1.000	1.000
prep_to	30	0	0	1.000	1.000	1.000
prep_until	1	0	0	1.000	1.000	1.000
prep_upon	1	0	0	1.000	1.000	1.000
prep_while	1	0	0	1.000	1.000	1.000
prep_with	16	0	0	1.000	1.000	1.000
prep_within	1	0	0	1.000	1.000	1.000
prep_without	2	0	0	1.000	1.000	1.000
prepc_for	3	0	0	1.000	1.000	1.000
prepc_from	2	0	0	1.000	1.000	1.000
prepc_of	1	0	0	1.000	1.000	1.000
prepc_on	1	0	0	1.000	1.000	1.000
prepc_to	1	0	0	1.000	1.000	1.000
prepc_while	2	0	0	1.000	1.000	1.000
prepc_with	1	0	0	1.000	1.000	1.000
prepc_without	3	0	0	1.000	1.000	1.000
prt	27	1	1	0.964	0.964	0.964
purpcl	1	0	0	1.000	1.000	1.000
quantmod	12	0	0	1.000	1.000	1.000
rcmod	27	2	7	0.931	0.794	0.857
root	325	14	14	0.959	0.959	0.959
root_and	12	0	0	1.000	1.000	1.000
root_but	18	1	0	0.947	1.000	0.973
root_or	2	0	0	1.000	1.000	1.000
tmod	34	2	7	0.944	0.829	0.883
xcomp	77	7	5	0.917	0.939	0.928
SUM	3161	143	244			

(C; correctly parsed, I; incorrectly parsed, FL; failed to be parsed, P; precision, R; recall, F; F-score)

Table 7.2 is the descriptive statistics of word count, degree centralities, closeness centralities, and dependency distances before and after manual corrections of the typed-dependency trees for the English sentences in Basic300.

Table 7.2. The descriptive statistics of word count, degree centralities, closeness centralities, and dependency distances before and after manual corrections of the typed-dependency trees for the English sentences in Basic300.

	Wordcount	Degree centralities		Closeness centralities		Dependency Distance	
		before	after	before	after	before	after
Average	11.059	0.426	0.432	0.428	0.432	2.505	2.489
S.E.	0.165	0.010	0.011	0.004	0.004	0.031	0.029
median	11.00	0.389	0.389	0.419	0.421	2.400	2.417
mode	11.00	0.389	0.267	0.500	0.400	2.000	2.000
S.D.	3.036	0.184	0.197	0.082	0.082	0.568	0.543
Var.	9.215	0.034	0.039	0.007	0.007	0.323	0.295
Kurtosis	-0.397	2.182	1.723	0.925	1.062	0.300	0.352
Skewness	-0.429	1.385	1.350	0.873	0.910	0.626	0.654
Range	14	0.9	0.9	0.460	0.496	3.500	3.188
Min.	4	0.1	0.1	0.254	0.254	1.000	1.250
Max	18	1.0	1.0	0.714	0.750	4.500	4.438
Sum	3749	144.364	146.510	145.066	146.381	849.292	843.810
Sample	339	339	339	339	339	339	339

The figures below show the distribution of degree centralities, closeness centralities and dependency distances before and after manual corrections, respectively. As their near-linear distributions indicate, these three measures do not greatly change before and after manual correction.

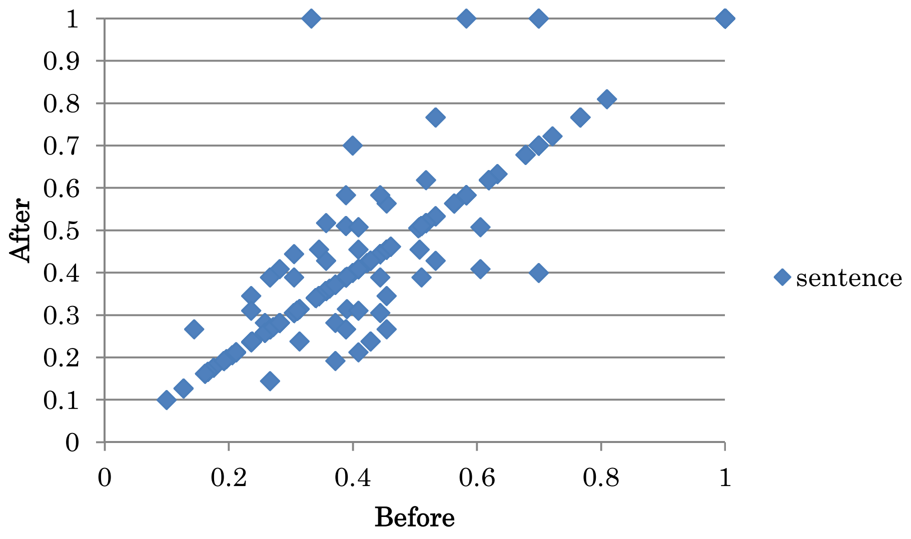


Figure 7.9. The distribution of degree centralities of the English sentences in *Basic300* before and after manual correction (Before: the degree centralities before manual correction; After: the degree centralities after manual correction).

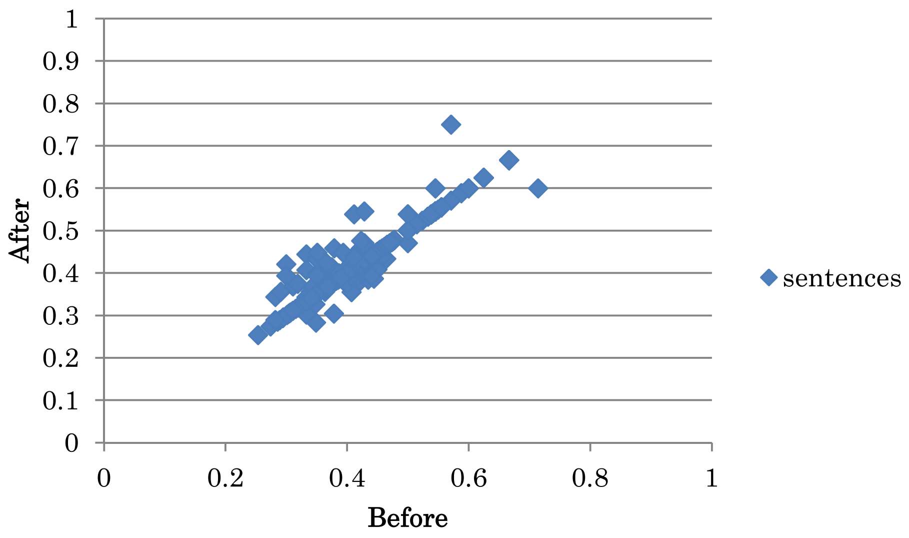


Figure 7.10. The distribution of closeness centralities of the English sentences in *Basic300* before and after manual correction (Before: the closeness centralities before manual correction; After: the closeness centralities after manual correction).

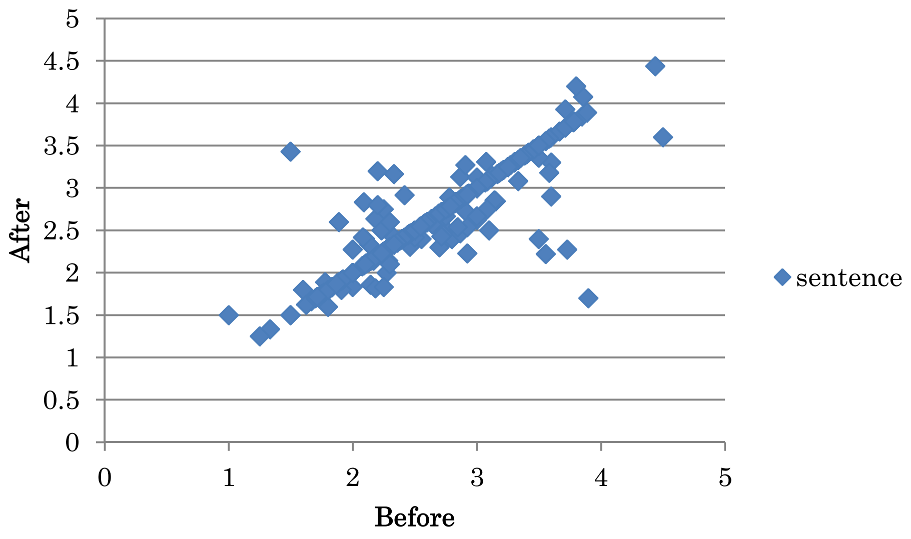


Figure 7.11. The distribution of dependency distances of the English sentences in *Basic300* before and after manual correction (Before: the dependency distances before manual correction; After: the dependency distances after manual correction)

7.3.4.1 Distribution of degree centralities of the English sentences in Basic300 before and after manual correction

The distribution of the English sentences in Basic300 in terms of their degree centralities (flatness measures) did not change dramatically before and after manual correction. In order to determine if the degree centralities of the English sentences in Basic 300 before and after manual correction were normally distributed, the Kolmogorov-Smirnov test for normality was conducted. The software used for this test was R version 2.15.0. The results indicated that the distribution of the degree centralities before manual correction deviated from a normal distribution (D=0.146, p<0.05), and that the distribution of the degree centralities after manual correction also deviated from a normal distribution (D=0.147, p<0.05).

Since the degree centralities before and after manual correction were not normally distributed, a t-test could not be conducted to compare the degree centralities of the English sentences in

Basic 300 before manual correction to those after manual correction. Therefore, a Mann-Whitney U test was conducted to compare them, with alpha set at the 5% level, and the null hypothesis is that the average degree centrality before manual correction is the same as the average degree centrality after manual correction. The results were not significant (*p*=0.97); hence the null hypothesis could not be rejected.

7.3.4.2 Distribution of closeness centralities of the English sentences in Basic300 before and after manual correction

The distribution of the sentences in Basic300 in terms of their closeness centralities (embeddedness measures) also did not change dramatically before and after manual correction, as figures below show. In order to determine if the closeness centralities of the English sentences in Basic 300 before and after manual correction were normally distributed, the Kolmogorov-Smirnov test for normality was conducted. The software used for this test was R version 2.15.0. The results indicated that the distribution of the closeness centralities before manual correction deviated from a normal distribution (*D*=0.104, *p*<0.05), and that the distribution of the degree centralities after manual correction also deviated from a normal distribution (*D*=0.114, *p*<0.05).

Since the closeness centralities before and after manual correction were not normally distributed, which was the same case for the degree centralities, a t-test could not be conducted to compare the closeness centralities of the English sentences in Basic 300 before manual correction to those after manual correction. Therefore, a Mann-Whitney U test was conducted to compare them, with alpha set at the 5% level, and the null hypothesis is that the average closeness centrality before manual correction is the same as the average closeness centrality after manual correction. The results were not significant (*p*=0.52); hence the null hypothesis could not be rejected.

7.3.4.3 Distribution of dependency distances of the English sentences in Basic300 before and after manual correction

The distribution of the sentences in Basic300 in terms of their average dependency distances also does not change dramatically before and after manual correction. In order to determine if the dependency distances of the English sentences in Basic 300 before and after manual correction were normally distributed, the Kolmogorov-Smirnov test for normality was conducted. The software used for this test was R version 2.15.0. The results indicated that the distribution of the dependency distances before manual correction deviated from a normal distribution (D=0.081, p<0.05), and that the distribution of the dependency distances after manual correction also deviated from a normal distribution (D=0.076, p<0.05).

Since the dependency distances before and after manual correction are not normally distributed, similar to the cases for the degree centralities and the closeness centralities, a t-test could not be conducted to compare the dependency distances of the English sentences in Basic 300 before manual correction to those after manual correction. Therefore, a Mann-Whitney U test was conducted to compare them, with alpha set at the 5% level, and the null hypothesis is that the average dependency distance before manual correction is the same as the average dependency distance after manual correction. The results were not significant (p=0.81); hence the null hypothesis could not be rejected.

7.3.5 Discussion

Statistical analyses show that the distributions of degree centralities, closeness centralities, and dependency distances of the English typed-dependency trees of Basic300 (parsed output of the English sentences by the Stanford Parser) before manual correction were not significantly different from those after manual correction. This result is a desirable one, especially for researchers who examine these measures obtained from the sentences in larger-scale corpora, where manual correction of the parsed output is a laborious and time-consuming task.

7.3.6 Related work

Cer, de Marneffe, Jurafsky & Manning (2010) compared a number of parsers in terms of Stanford Dependency representation. The data used in their study was the section 22 of Penn Treebank. The parse output of each of the different parsers is systematically converted into those using Stanford Dependency. The f-score of "*labeled attachment*" (correctly parsed dependency relation with the correct dependency type) of Stanford Parser was 84.2. Their research question was the trade-offs between parsing accuracy and parsing speed; therefore, they did not report in detail the accuracy and precision of each dependency type in the parse output of these different parsers; however, they reported that these parsers often yield errors of dependency relationships and dependency types for subordinated clauses, prepositional and adverbial phrases.

7.4 Parsing Accuracy of KNP

7.4.1 Data description

The data chosen for the analysis described below are the Japanese sentences in Basic 300, which is the same corpus used for investigating the parsing accuracy of the Stanford Parser (see Section 7.3). This corpus contains 339 English sentences, along with their Japanese translations.

7.4.2 Four types of dependency parse errors

Along with the two types of parse errors found in the output of the Stanford Parser (*dependency-relation errors* and *dependency-type* errors; see Section 7.3.1), there are two other types of parse errors in the output of KNP. The first type of parse error is that two or more syntactic units are incorrectly segmented as one single syntactic unit. The second type of parse error is that one syntactic unit is incorrectly segmented as more than one syntactic unit.

Both types are due to syntactic-unit segmentation errors in the output of JUMAN, the morphological analyzer. KNP uses the output of this morphological analyzer as it is; therefore, incorrect morphological analyses by JUMAN yield incorrect dependency analyses by KNP.

This study does not attempt to improve the morphological analyses of Japanese sentences by JUMAN, because it is not the main topic of this study. Rather, in this study, the accuracy of KNP is examined by identifying incorrect dependency relations and types manually in the parsed output of a small corpus. If any incorrect syntactic-unit segmentation is found in the parsed output of a sentence, all the triples are treated as failed parses. Then, the degree and closeness centralities are calculated, both from the parser output before manual correction and from that after manual correction. The results are compared in order to see how dependency parse errors affect the degree and closeness centralities of the parsed output.

7.4.3 Procedure

First, the Japanese sentences in Basic 300 are morphologically analyzed by JUMAN; then the output is parsed by KNP. The output is then converted into Stanford-Parser style triples by an original Ruby script (see Appendix V), which is further converted by an original Ruby script (see

Appendix IV) into .net files for *Pajek*, in order to simplify the manual correction compared to that of the Stanford-Parser-style triples. If any of the dependency relations and dependency types in the sentence is incorrect, the content of the .net file is manually corrected. Incorrect morphological analyses are also manually corrected. The precision, recall and f-score of each dependency type are calculated, in order to see which dependency type fails to be correctly parsed most often. The centrality measures of both the parsed output *before* and *after* manual correction are also calculated by a Ruby script (see Appendix VI) written by the author of this thesis, then compared with each other.

7.4.4 Results

The list of Japanese sentences with incorrect parses is presented in Appendix III. Of all the 339 Japanese sentences in Basic300, 32 sentences (about 9.4%) are segmented into syntactic units incorrectly. Of all the Japanese sentences in Basic300, the number of sentences that contain at least one incorrect dependency is 69. This is around 20.35% of all the Japanese sentences in Basic300. Of all the dependencies in the Japanese sentences in Basic300 (1937 in total), the number of correctly parsed dependencies is 1715; hence, the parsing accuracy of KNP with automatic dependency-type annotation is more than 88% for Basic300.

Of all the incorrect dependencies, the number of dependencies that are parsed with an incorrect relation and the correct type is 72 (about 3.7% of all the dependencies). The number of dependencies that are parsed with an incorrect type and the correct relation is 38 (about 1.9% of all the dependencies). The number of dependencies that are parsed with an incorrect type and an incorrect relation is 117 (about 6% of all the dependencies).

The recall, precision, and f-score of each dependency type in Japanese are shown in Table 7.3.

Table 7.3. The recall, precision, and f-score of each dependency type in Japanese

type	C	I	FL	P	R	F	type	C	I	FL	P	R	F
advcl	14	5	6	0.737	0.700	0.718	advcl_yori	1	0	0	1.000	1.000	1.000
advcl_ba	12	1	3	0.923	0.800	0.857	advcl_youni	0	0	2	0.000	0.000	0.000
advcl_de	1	1	2	0.500	0.333	0.400	advmod	196	16	20	0.925	0.907	0.916
advcl_deatte	2	0	0	1.000	1.000	1.000	amod	38	1	2	0.974	0.950	0.962
advcl_dokoroka	0	0	1	0.000	0.000	0.000	appos	1	0	1	1.000	0.500	0.667
advcl_ga	19	1	1	0.950	0.950	0.950	ccomp	34	3	4	0.919	0.895	0.907
advcl_ka	0	0	1	0.000	0.000	0.000	det	92	4	3	0.958	0.968	0.963
advcl_kara	4	4	10	0.500	0.286	0.364	focus	43	7	5	0.860	0.896	0.878
advcl_keredomo	4	0	0	1.000	1.000	1.000	nn	0	0	3	0.000	0.000	0.000
advcl_kouga	1	0	0	1.000	1.000	1.000	postp_de	51	0	0	1.000	1.000	1.000
advcl_ku	4	0	0	1.000	1.000	1.000	postp_ga	117	13	20	0.900	0.854	0.876
advcl_kute	2	0	0	1.000	1.000	1.000	postp_he	5	0	0	1.000	1.000	1.000
advcl_kutemo	1	0	0	1.000	1.000	1.000	postp_kara	7	2	3	0.778	0.700	0.737
advcl_made	2	1	1	0.667	0.667	0.667	postp_ni	137	16	20	0.895	0.873	0.884
advcl_mama	0	0	1	0.000	0.000	0.000	postp_nitotte	0	0	1	0.000	0.000	0.000
advcl_nagara	2	0	0	1.000	1.000	1.000	postp_nitsuite	0	0	1	0.000	0.000	0.000
advcl_ni	2	0	1	1.000	0.667	0.800	postp_niyotte	1	0	0	1.000	1.000	1.000
advcl_nodakara	2	1	1	0.667	0.667	0.667	postp_no	113	2	3	0.983	0.974	0.978
advcl_node	8	0	0	1.000	1.000	1.000	postp_to	16	1	1	0.941	0.941	0.941
advcl_tara	8	1	1	0.889	0.889	0.889	postp_wo	143	12	11	0.923	0.929	0.926
advcl_te	17	2	2	0.895	0.895	0.895	postp_yori	3	0	0	1.000	1.000	1.000
advcl_temo	8	0	0	1.000	1.000	1.000	rcmod	105	12	20	0.897	0.840	0.868
advcl_tewa	3	0	1	1.000	0.750	0.857	root	304	35	36	0.897	0.894	0.895
advcl_to	9	0	3	1.000	0.750	0.857	topic	183	29	31	0.863	0.855	0.859
								1715	170	222			

(C; correctly parsed, I; incorrectly parsed, FL; failed to be parsed, P; precision, R; recall, F; f-score)

Table 7.4 is the descriptive statistics of word counts, degree centralities and closeness centralities before and after manual corrections of the typed-dependency trees for the Japanese sentences in Basic300.

Table 7.4. The descriptive statistics of word counts, degree centralities and closeness centralities before and after manual corrections of the typed-dependency trees for the Japanese sentences in Basic300.

	Word count	Degree centralities		Closeness centralities		Dependency distances	
		before	after	before	after	before	after
Average	6.614	0.529	0.511	0.501	0.494	2.347008	2.32387
S.E.	0.111	0.014	0.014	0.007	0.007	0.0261	0.025537
median	7.000	0.444	0.429	0.471	0.471	2.4	2.333333
mode	8.000	1.000	1.000	0.500	0.500	2	2
S.D.	2.049	0.256	0.262	0.130	0.132	0.480556	0.470181
Var.	4.196	0.065	0.069	0.017	0.017	0.230934	0.22107
Kurtosis	-0.351	-0.558	-0.532	3.024	2.996	0.476541	0.478793
Skewness	-0.043	0.661	0.649	1.362	1.358	-0.20517	-0.19362
Range	10	0.933	0.952	0.788	0.750	3	2.875
Min.	2	0.067	0.048	0.212	0.250	1	1
Max	12	1.000	1.000	1.000	1.000	4	3.875
Sum	2242	179.479	173.096	169.824	167.447	795.6358	787.792
Sample	339	339	339	339	339	339	339

The figures below show the distribution of degree centralities, closeness centralities and dependency distances before and after manual corrections, respectively. As their near-linear distributions indicate, these three measures do not dramatically change before and after manual correction.

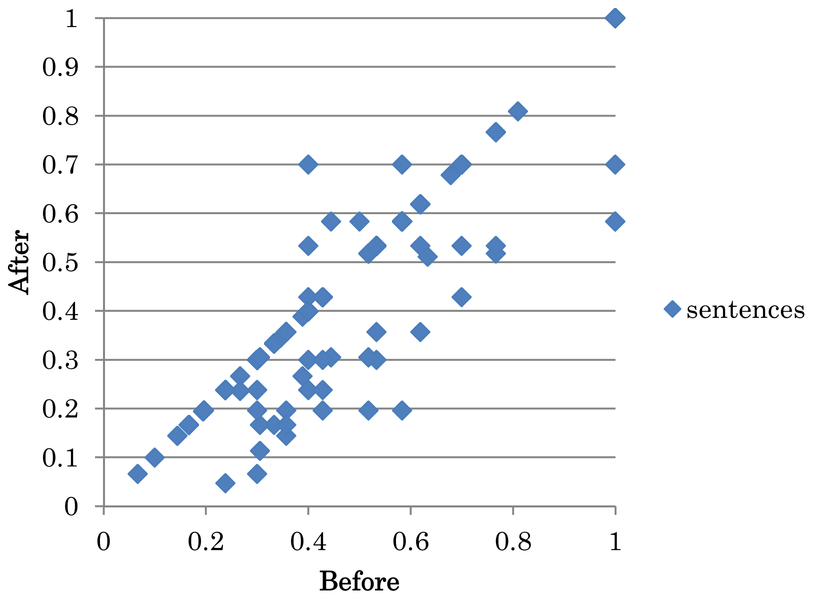


Figure 7.12. The distribution of degree centralities of the Japanese sentences in *Basic300* before and after manual correction (Before: the degree centralities before manual correction; After: the degree centralities after manual correction).

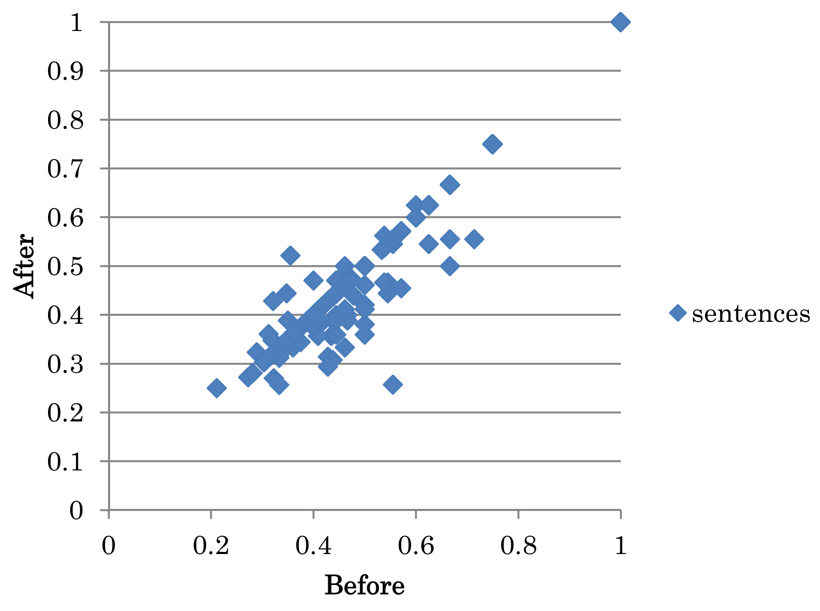


Figure 7.13. The distribution of closeness centralities of the Japanese sentences in *Basic300* before and after manual correction (Before: the closeness centralities before manual correction; After: the closeness centralities after manual correction).

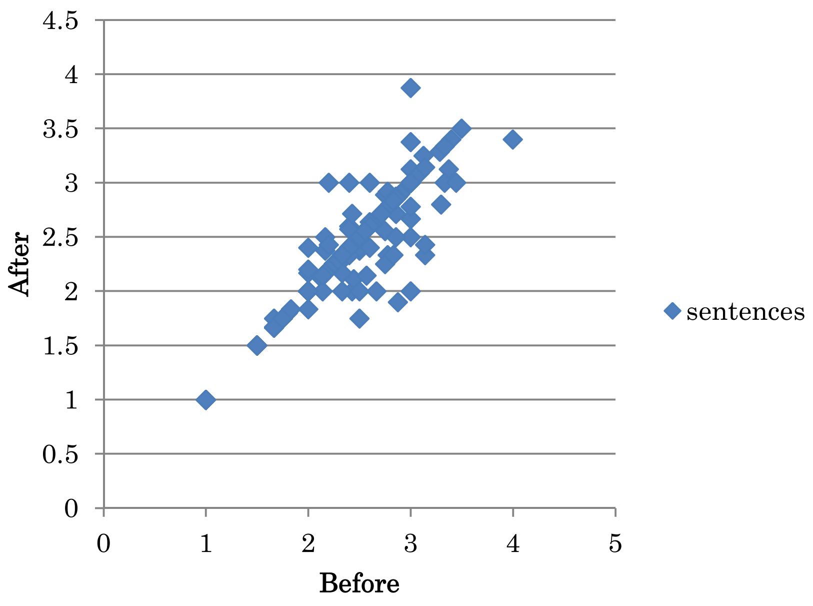


Figure 7.14. The distribution of dependency distances of the Japanese sentences in *Basic300* before and after manual correction (Before: the dependency distances before manual correction; After: the dependency distances after manual correction)

7.4.4.1 Distribution of degree centralities of the Japanese sentences in Basic300

The distribution of the Japanese sentences in Basic300 in terms of their degree centralities (flatness measures) did not change dramatically before and after manual correction. In order to determine if the degree centralities of the Japanese sentences in Basic 300 before and after manual correction were normally distributed, the Kolmogorov-Smirnov test for normality was conducted, as was the case for the English sentences. The software used for this test was R version 2.15.0. The results indicated that the distribution of the degree centralities before manual correction deviated from a normal distribution (D=0.151, p<0.05), and that the distribution of the degree centralities after manual correction also deviated from a normal distribution (D=0.145, p<0.05).

Since the degree centralities before and after manual correction were not normally distributed, a t-test could not be conducted to compare the degree centralities of the Japanese sentences in Basic 300 before manual correction to those after manual correction. Therefore, a Mann-Whitney U test was conducted to compare them, with alpha set at the 5% level, and the null hypothesis is that the average degree centrality before manual correction is the same as the average degree centrality after manual correction. The results were not significant (*p*=0.276); hence the null hypothesis could not be rejected.

7.4.4.2 Distribution of closeness centralities of the Japanese sentences in Basic300

The distribution of the Japanese sentences in Basic300 in terms of their closeness centralities (embeddedness measures) also did not change dramatically before and after manual correction. In order to determine if the closeness centralities of the Japanese sentences in Basic 300 before and after manual correction were normally distributed, the Kolmogorov-Smirnov test for normality was conducted. The software used for this test was R version 2.15.0. The results indicated that the distribution of the closeness centralities before manual correction deviated from a normal distribution (*D*=0.143, *p*<0.05), and that the distribution of the degree centralities after manual correction also deviated from a normal distribution (*D*=0.137, *p*<0.05).

Since the closeness centralities before and after manual correction were not normally distributed, which was also the case for degree centralities, a t-test could not be conducted to compare the closeness centralities of the Japanese sentences in Basic 300 before manual correction to those after manual correction. Therefore, a Mann-Whitney U test was conducted to compare them, with alpha set at the 5% level, and the null hypothesis is that the average closeness centrality before manual correction is the same as the average closeness centrality after manual correction. The results were not significant (*p*=0.408); hence the null hypothesis could not be rejected.

7.4.4.3 Distribution of dependency distances of the Japanese sentences in Basic300

The distribution of the Japanese sentences in Basic300 in terms of their average dependency distance also did not change dramatically before and after manual correction. In order to determine if the dependency distances of the Japanese sentences in Basic 300 before and after manual correction were normally distributed, the Kolmogorov-Smirnov test for normality was conducted. The software used for this test was R version 2.15.0. The results indicated that the distribution of the dependency distances before manual correction deviated from a normal distribution (D=0.111, p<0.05), and that the distribution of the dependency distances after manual correction also deviated from a normal distribution (D=0.112, p<0.05).

Since the dependency distances before and after manual correction are not normally distributed as is the case in the degree centralities and the closeness centralities, a t-test could not be conducted to compare the dependency distances of the Japanese sentences in Basic 300 before manual correction to those after manual correction. Therefore, Mann-Whitney U test was conducted to compare them, with alpha set at the 5% level, and the null hypothesis is that the average dependency distance before manual correction is the same as the average dependency distance after manual correction. The results were not significant (p=0.483); hence the null hypothesis could not be rejected.

7.4.5 Discussion

Statistical analyses show that the distributions of degree centralities, closeness centralities, and dependency distances of the Japanese typed-dependency trees of Basic300 (i.e., Stanford-Dependency-style triples which were obtained from the parsed output of the Japanese sentences by KNP) before manual correction were not significantly different from those after manual correction, as was the case in the measures obtained from the English typed-dependency trees. This result is a desirable one, especially for researchers who examine these measures among the sentences in larger-scale corpora, where manual correction of the parsed output will be a laborious and time-consuming task.

7.4.6 Related work

Kurohashi & Nagao (1998) reports that the accuracy of KNP is 91.1% with respect to about 40,000 sentences in a manually corrected corpus. Their result cannot be compared to mine directly, because they did not take the dependency type of each dependency relation into consideration, because the output of KNP does not include the dependency type of each dependency relation. The result in my study, on the other hand, includes the accuracy of the dependency type of each dependency relation in the parse output.

7.5 Degree and Closeness Centralities of English-Japanese Sentence Pairs[55]

This section deals with the degree and closeness centralities of English-Japanese sentence pairs, in order to see how these centrality measures reflect their structural differences and similarities in terms of flatness and embeddedness (see Section 4.4).

[55] The content of this section is based on Oya (2013b).

7.5.1 Data description

The data used here are the same as in Section 7.3 for parsing accuracy of the Stanford Parser, and in Section 7.4 for parsing accuracy of KNP, viz. the English-Japanese sentence pairs in *Basic300* (Iida 2010).

7.5.2 Procedure

First, the English sentences in Basic300 are parsed by the Stanford Parser (the output option is set to Collapsed Tree; see Section 5.4.4), and the Japanese counterparts are parsed by KNP.

Second, the parsed output for each sentence is checked in terms of the dependency relation and dependency type. If any of the dependency relations and types in the output file is found incorrect, they are manually corrected.

Then, the degree and closeness centrality of each sentence is calculated by a Ruby script originally written by the author of this thesis (see Appendix VII).

The procedure taken in this section does not take the semantics of English-Japanese pairs into consideration. The ten-word sentences in English do not necessarily correspond to ten-word Japanese sentences. The aim of the data analysis in this section is to focus on the structural setting of dependency trees in two particular languages, abstracting away the semantics of each sentence. If it is shown that the distributions of the degree centralities or the closeness centralities are found similar in both languages, then it can be argued that the sentences of these languages share the similar structural settings for their syntactic dependency trees, regardless of their meanings. If, on the other hand, the distributions of them are found different in both languages, then it can be argued that the sentences of these languages do not share the similar structural settings for their syntactic dependency trees.

7.5.3 Results

7.5.3.1 Descriptive statistics

The descriptive statistics presented in Table 7.5 show that English sentences have smaller degree centralities than Japanese ones on average, which means that English sentences tend to have less flat typed-dependency trees than Japanese ones. They have smaller closeness centralities than Japanese ones on average, which means that English sentences tend to have more embedded typed-dependency trees than Japanese ones. These two observations can be subsumed to the fact that the English sentences are longer than Japanese ones on average, as Oya (2012) indicated that longer sentences have smaller degree centralities (see Section 4.5); therefore, the degree centralities of the English sentences of a certain word count are compared to those of the Japanese sentences with the same word count, in order to abstract away the influence of different word counts on the degree centralities of these two languages. The same type of comparison is also conducted for closeness centralities.

Table 7.5. The descriptive statistics of the degree centralities, closeness centralities, and word counts of the sentences in Basic300 (n = 339) (Oya 2013b, p.159)

	Degree		Closeness		Word per sentence	
	English	Japanese	English	Japanese	English	Japanese
Mean	0.39	0.53	0.43	0.50	11.04	6.61
SD	0.18	0.26	0.08	0.13	3.03	2.04

7.4.3.2 Distributions of degree centralities

The distribution of sentences with the horizontal axis the degree centralities (flatness measures) and with the vertical axis the word counts reveals that the variation of English sentences in terms of their degree centralities is wider than that of Japanese sentences, as is shown in the figure below.

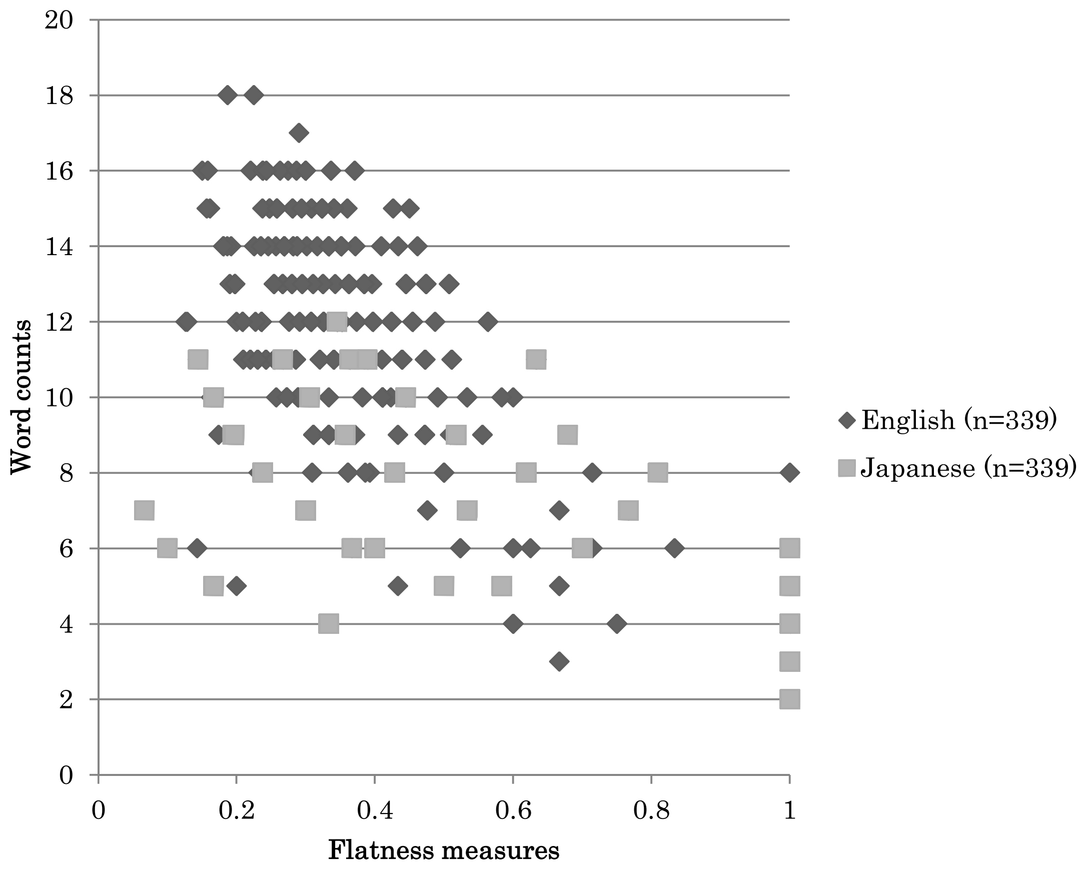


Figure 7.15. The distribution of degree centralities (flatness measures) and word counts (n = 339) (Oya 2013b, p.159)

A Mann-Whitney U test was conducted to compare the degree centralities of the English sentences and those of the Japanese sentences, with alpha set at the 5% level, and the null hypothesis is that the average degree centrality of the English sentences is the same as that of the Japanese sentences. The results were significant (p<0.05), hence the null hypothesis was rejected.

7.4.3.3 Distributions of closeness centrality

The distribution of closeness centralities makes a good contrast with that of degree centralities. As the figure below shows, closeness centralities (embeddedness measures) decrease in proportion to the word counts in the sentences:

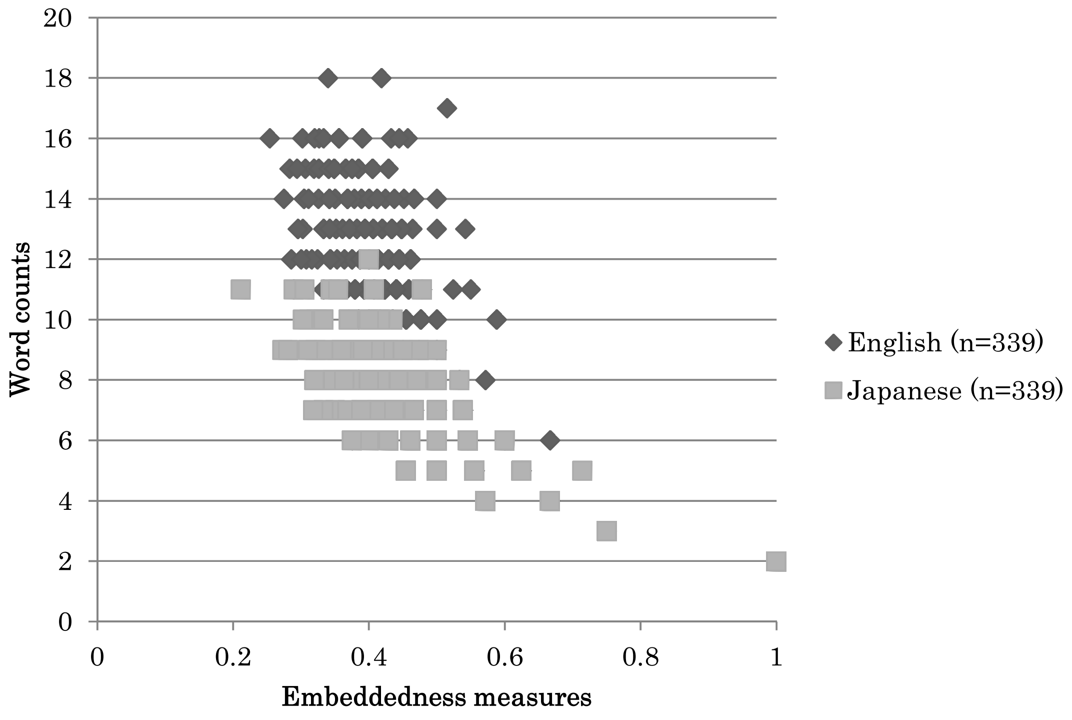


Figure 7.16. The distribution of closeness centralities and word counts (n = 339) (Oya 2013b, p.161)

As in the case of degree centralities, a Mann-Whitney U test was conducted to compare the closeness centralities of the English sentences and those of the Japanese sentences, with alpha set at the 5% level. The null hypothesis is that the average closeness centrality of the English sentences is the same as that of the Japanese sentences. The results were significant (p<0.05); hence the null hypothesis was rejected.

7.4.3.4 Distributions of Degree Centralities and Closeness Centralities among the Sentences of the Same Word Count

The different distributions of degree centralities and closeness centralities can be explicated if we focus on the sentences with the same word count. Here, we focus on eight-word, nine-word, and ten-word sentences of English and Japanese.

The figure below shows the ratios of different flatness measures (degree centralities) of all the eight-word sentences in Basic 300. 50% of Japanese eight-word sentences in Basic300 have flatness measure 0.428. No such prominent flatness measure is found in English eight-word sentences in Basic300.

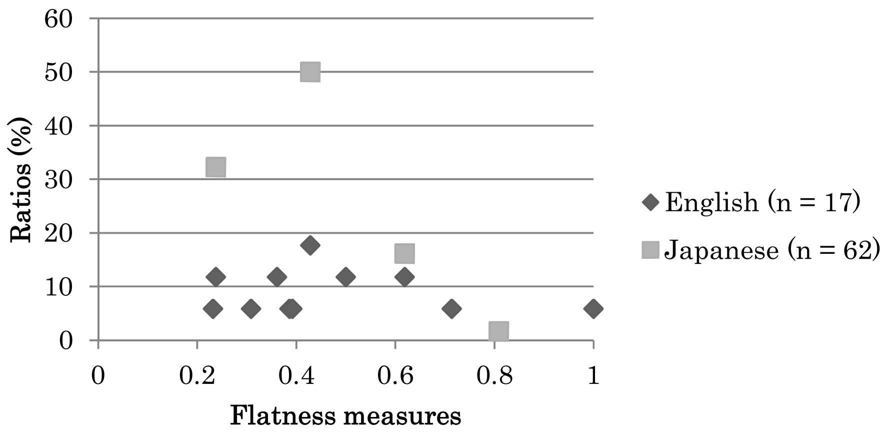


Figure 7.17. The ratios of different flatness measures (degree centralities) of all the eight-word

sentences in Basic300

The figure below shows the ratios of different flatness measures (degree centralities) of all the nine-word sentences in Basic 300. About 42% of Japanese nine-word sentences in Basic300 have the flatness measure 0.357. This flatness measure is also shared by about 25% of English nine-word sentences in Basic300.

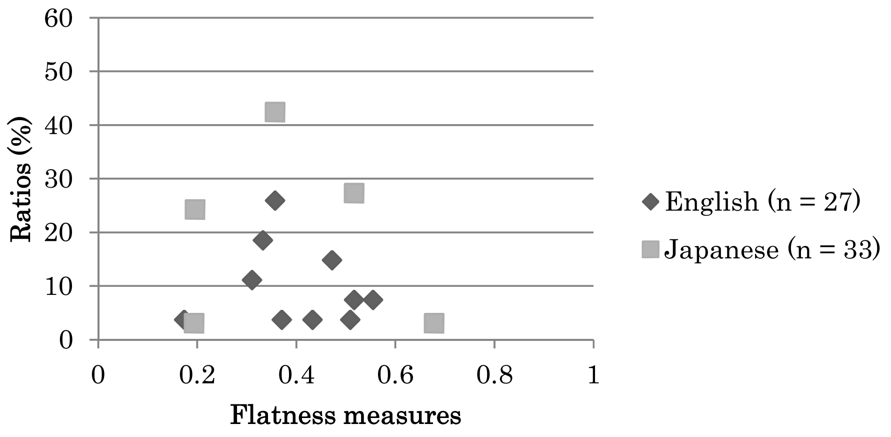


Figure 7.18. The ratios of different flatness measures (degree centralities) of all the nine-word sentences in Basic300

The figure below shows the ratios of different flatness measures (degree centralities) of all the ten-word sentences in Basic 300. About 43% of Japanese ten-word sentences in Basic300 have the flatness measure 0.16, and about 24% of English ten-word sentences in Basic300 have the flatness measure 0.411.

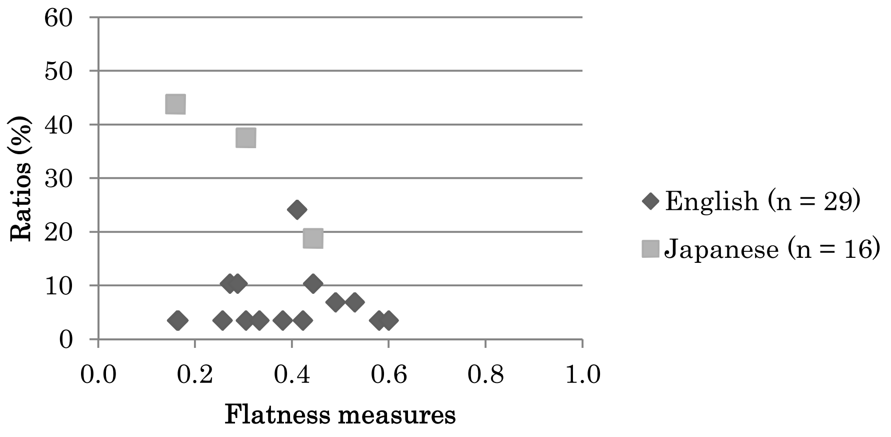


Figure 7.19. The ratios of different flatness measures (degree centralities) of all the ten-word sentences in Basic300

The figure below shows the ratios of different embeddedness measures (closeness centralities) of all the eight-word sentences in Basic 300. About 29% of Japanese eight-word sentences in Basic300 have the embeddedness measure 0.47, and about 26% of English eight-word sentences in Basic300 have the embeddedness measure 0.44.

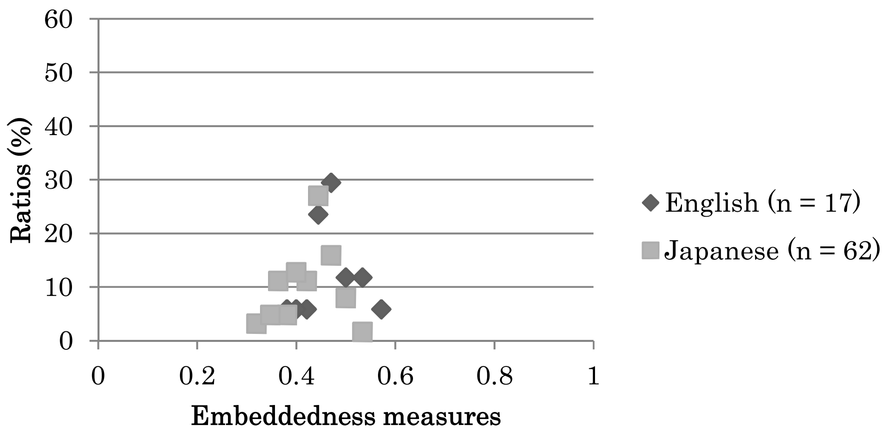


Figure 7.20. The ratios of different embeddedness measures (closeness centralities) of all the eight-word sentences in Basic300

The figure below shows the ratios of different embeddedness measures of all the nine-word sentences in Basic 300. About 22% of Japanese nine-word sentences in Basic300 have the embeddedness measure 0.42, and about 18% of English nine-word sentences in Basic300 also have the embeddedness measure 0.42.

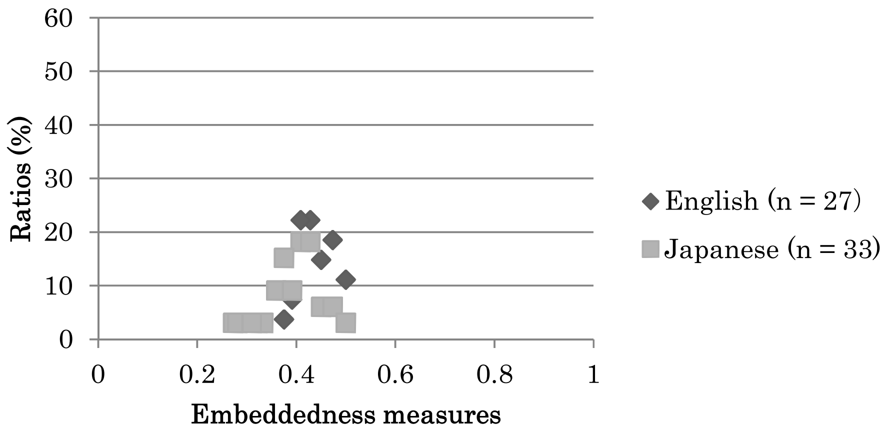


Figure 7.21. The ratios of different embeddedness measures of all the nine-word sentences in Basic 300.

The figure below shows the ratios of different embeddedness measures of all the ten-word sentences in Basic 300.

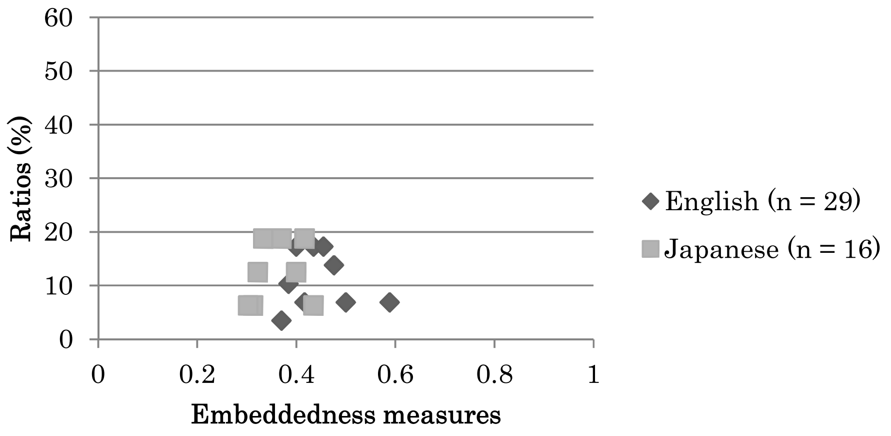


Figure 7.22. . The ratios of different embeddedness measures of all the ten-word sentences in Basic 300.

The table below summarizes the number of different degree centralities and closeness centralities among eight-word, nine-word and ten-word sentences of English and Japanese.

Table 7.6. The numbers of different values of degree centralities and of closeness centralities among eight-word, nine-word, and ten-word sentences of English and Japanese

	Word Count					
	10		9		8	
	English	Japanese	English	Japanese	English	Japanese
D	15	3	10	5	11	4
C	9	8	7	13	8	10

D: the number of different values of degree centrality
C: the number of different values of closeness centrality

This table shows that degree centralities of English sentences of the same word count (within the range of word counts from 8 to 10) are more diverse than those of Japanese sentences of the same word count. This indicates that the word count of a given English sentence does not determine its degree centrality as uniquely as that of a given Japanese sentence.

Closeness centralities of English sentences, on the other hand, of the same word count tend to be less diverse than those of Japanese sentences of the same word count (except for ten-word sentences).

7.5.4 Discussion

The result shows that the distribution of degree centralities of English typed-dependency trees is more diverse than the distribution of degree centralities of Japanese typed-dependency trees. This indicates that the structural settings of English typed-dependency trees are more diverse than the structural settings of Japanese typed-dependency trees, in terms of their flatness. The difference of the distribution of their closeness centralities, on the other hand, is not as obvious as that of their degree centralities. This indicates that the structural settings of English typed-dependency trees are as much diverse as the structural settings of Japanese typed-dependency trees, in terms of their embeddedness. These results can be interpreted as follows; degree centralities reflect the structural differences between English and Japanese, while closeness centralities reflect the structural similarities between them.

As is mentioned in section 7.5.2, the procedure of this analysis does not take the semantics of sentences into consideration. Comparing ten-word English sentences with ten-word Japanese sentences ignores one of the essential aspects of language, viz. their meanings, and this might be argued to be a drawback of this study. Therefore, the next research question is to take the semantics of dependency trees into consideration, i.e., comparing the syntactic typed-dependency trees of English sentences and their Japanese translation counterparts, which will be one the topic of my research in future.

7.6 Degree and Closeness Centralities of sentences from Manually Annotated Sub Corpus of American National Corpus (MASC 500k)[56]

Manually annotated sub-corpus of American National Corpus (MASC 500k) (Ide et al. 2008)

[56] This subsection is based on Oya (2013a).

contains approximately 500,000 words of contemporary American English, drawn from Open American National Corpus (OANC) (Ide & Suderman 2004). Original MASC 500k contains various kinds of manually-annotated tags such as sentence boundaries, token, lemma, POSs, noun and verb chunks, and named entities. MASC 500k covers a wide range of genres: the written section contains texts from newspapers, fictions, non-fictions, technical reports, short fictions taken from a website Ficlet (now closed), travel guides, essays, government documents, jokes, blogs, emails, spam emails, movie scripts; the spoken section contains texts from speeches and debates.

7.6.1 Data description

The descriptive statistics of each subsection in the written section of MASC 500k is as follows: the e-mail section is not included in this study because it contains too many repetitions of the same text due to the citations in the reply messages. The movie-script section is also not included in this study because the sentences in this section are intended to be spoken by the actors and actresses in the movies; hence, they must be regarded as spoken data.

Table 7.7. The total number of sentences, the total number of words and the mean length of a sentence in each genre (Oya 2013a)

subsections	Sentences	Words	WPS	
			Mean	S.D
Blog	1524	28381	18.62	12.34
Essay	1072	27367	25.52	13.18
Ficlets	2645	30555	13.34	7.15
Fiction	2639	37531	14.22	8.32
Govt-doc	1028	24277	23.61	12.55
Jokes	2254	31751	14.08	8.6
Journal	867	21997	25.37	14.45
News	1196	26877	22.47	10.43
Non-Fiction	1278	26441	20.68	11.41
Technical	825	19787	23.98	13.58
TravelGuide	1196	24187	20.23	8.6
	16524	299151		

WPS: Word per sentence

The table shows that the mean WPSs and the standard deviations in the subsections Fiction, Ficlets and Jokes are relatively smaller than those in other subsections, and that the largest WPS is found in the subsection Essay.

7.6.2 Procedure

The raw texts without tags (downloaded as a data-only file from the website of ANC: http://www.anc.org/MASC/Download.html) are parsed by Stanford Parser after manual extraction of unnecessary part of texts such as titles or dates. Then, the degree centrality, closeness centrality and dependency distance of the output typed-dependency trees for the sentences in the texts are calculated automatically. This is exerted by an original script written in Ruby (see Appendix VII), in order to see whether the different genres of texts have different distribution of degree centrality, closeness centrality, and dependency distances. In order to examine their distributions more precisely without the effect of word counts of a sentence (see Section 4.5), these three values of sentences of the same word count are chosen from each genre, and the distributions of them are compared with each other.

7.6.3 Results

Table 7.8 shows the descriptive statistics of the degree centrality, closeness centrality and dependency distance (Dep.Dist.) of the sentences in each subsection of the written section of MASC500k.

Table 7.8. The descriptive statistics of the degree centrality, closeness centrality, and Dep.Dist. of the sentences in each subsection of the written section of MASC500k

	Degree		Closeness		Dep.Dist.	
	Mean	S.D.	Mean	S.D.	Mean	S.D.
Blog	0.43	0.26	0.39	0.12	2.99	1.19
Essay	0.26	0.19	0.32	0.09	3.58	1.09
Ficlets	0.57	0.29	0.47	0.11	2.31	1.01
Fiction	0.54	0.27	0.43	0.11	2.65	1.05
Govt-doc	0.26	0.18	0.33	0.10	3.41	1.12
Jokes	0.51	0.27	0.44	0.13	2.55	1.15
Journal	0.27	0.19	0.33	0.09	3.63	1.35
News	0.27	0.17	0.33	0.09	3.34	0.94
Non-Fiction	0.37	0.22	0.36	0.10	3.20	1.07
Technical	0.32	0.22	0.34	0.12	3.37	1.27
TravelGuide	0.35	0.18	0.36	0.09	3.25	0.97

The top-three largest mean degree centralities are found in the subsections Ficlets, Fiction and Jokes, whose standard deviation is also larger than those of other sections. The smallest degree centrality among them is found in the subsection Essay.

The top-three largest mean closeness centralities are found in the subsections Ficlets, Fiction and Jokes. The largest standard deviation among these subsections is found in Jokes. The subsections Blog and Technical have standard deviations which are larger than those of Ficlets and Fiction.

The top-three shortest dependency distances are found in the subsections Fiction, Ficlets and Jokes. The longest dependency distance among these subsections is found in Journal, whose standard deviation is also the largest. The smallest standard deviation is found in News.

Oya (2013a, p.48) argued that "example typed-dependency trees taken from these subsections will illustrate the claim that flatter trees have larger degree centralities and more embedded trees have smaller closeness centralities." For example, the figure below is the typed-dependency tree for an example of 10-word sentences[57], selected from the subsection Journal.

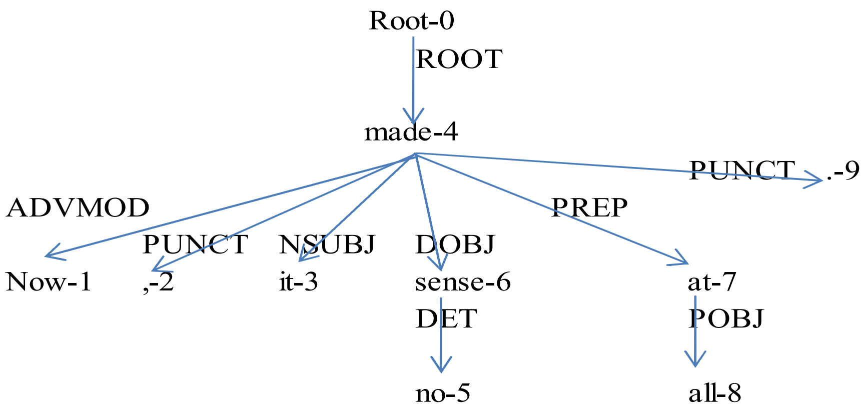


Figure 7.23. The typed-dependency tree for "Now, it made no sense at all." (Oya 2013a, p.48)

Figure 7.24 is the typed-dependency tree for another 10-word sentence selected from the subsection Journal.

[57] Notice that the term "word count" throughout this section includes punctuations (commas, quotation marks, or periods) and the abstract node ROOT.

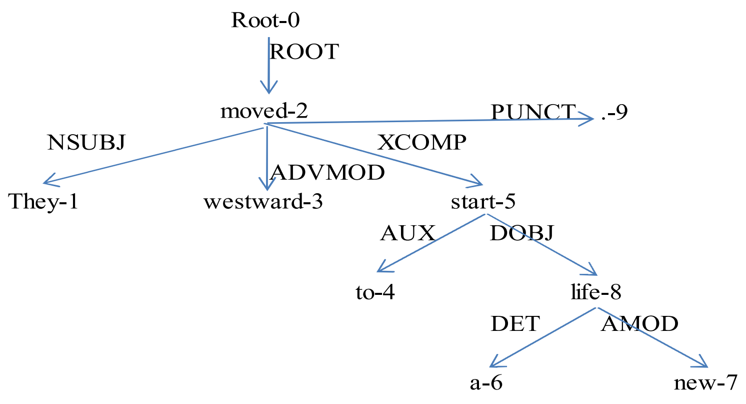


Figure 7.24. The typed-dependency tree for “They moved westward to start a new life.” (Oya 2013a, p.48)

Figure 7.25 is the typed-dependency tree for an example 10-word sentence selected from the subsection Blog.

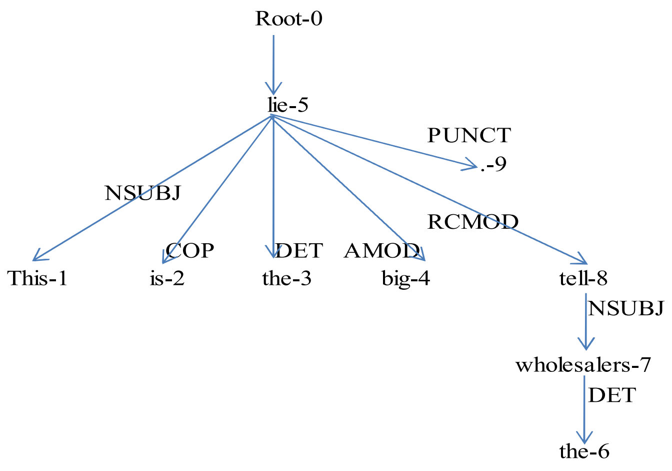


Figure 7.25. The typed-dependency tree for “This is the big lie the wholesalers tell.” (Oya 2013a, p.49)

Figure 7.26 is the typed-dependency tree for another 10-word sentence selected from the subsection Blog.

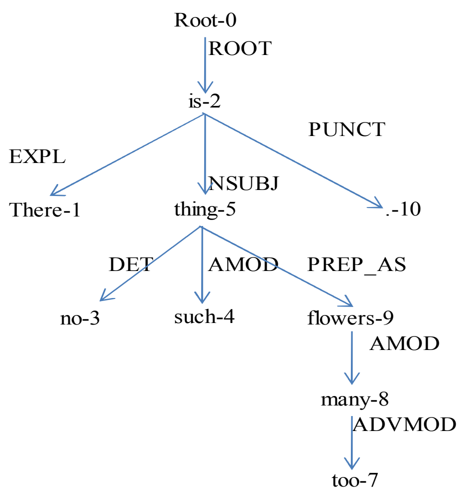


Figure 7.26. The typed-dependency tree for "There is no such thing as too many flowers." (Oya 2013a, p.49)

The degree centralities, closeness centralities, and average dependency distances of the typed-dependency trees for the example 10-word sentences across different subsections are summarized in Table 7.9 below. The same degree centrality of the trees in Figure 7.23 and in Figure 7.25 indicates that these trees share the same flatness. The typed-dependency tree in Figure 7.23 has the largest closeness centrality compared to other trees; hence, it is the least embedded than others. The typed-dependency tree in Figure 7.26 has the smallest degree centrality and the smallest closeness centrality; hence, it is the least flat and the most embedded one among them. The average dependency distances of these typed-dependency trees do not vary as much as their degree centralities and their closeness centralities.

Table 7.9. The degree centralities, closeness centralities, and average dependency distances of the

example typed-dependency trees (Oya 2013a, p.49)

	Degree	Closeness	Dep.Dist
Figure 7.23	0.722	0.473	2.444
Figure 7.24	0.444	0.391	2.333
Figure 7.25	0.722	0.450	2.666
Figure 7.26	0.305	0.360	2.555

Notice that the different distributions of degree centrality and closeness centrality among sentences in different genres can be “nothing but a paraphrase of different distribution of WPSs among these sentences (Oya 2013a, p.49)”. For example, the degree centrality and closeness centrality of the sentences in Fiction, Ficlet and Jokes are larger than those in other subsections. This can be the result of the fact that they contain shorter sentences compared to those in other genres on average, as their smaller WPSs indicate.[58] In order to address this issue, it will be desirable to control the number of words in a sentence to examine how both centralities of the sentences of equal word count show different distributions.

7.6.3.1 Distributions of the degree centralities in MASC500k

This section deals with the distributions of degree centralities of the sentences of the same word count. Figure 7.27 is the distribution of degree centralities of 10-word sentences in each genre of MASC500k, and Figure 7.28 is the distribution of degree centralities of 20-word sentences in the same corpus.

[58] Degree centrality tends to become smaller in proportion to the number of words in sentences (Satoshi Yoshida, p.c.).

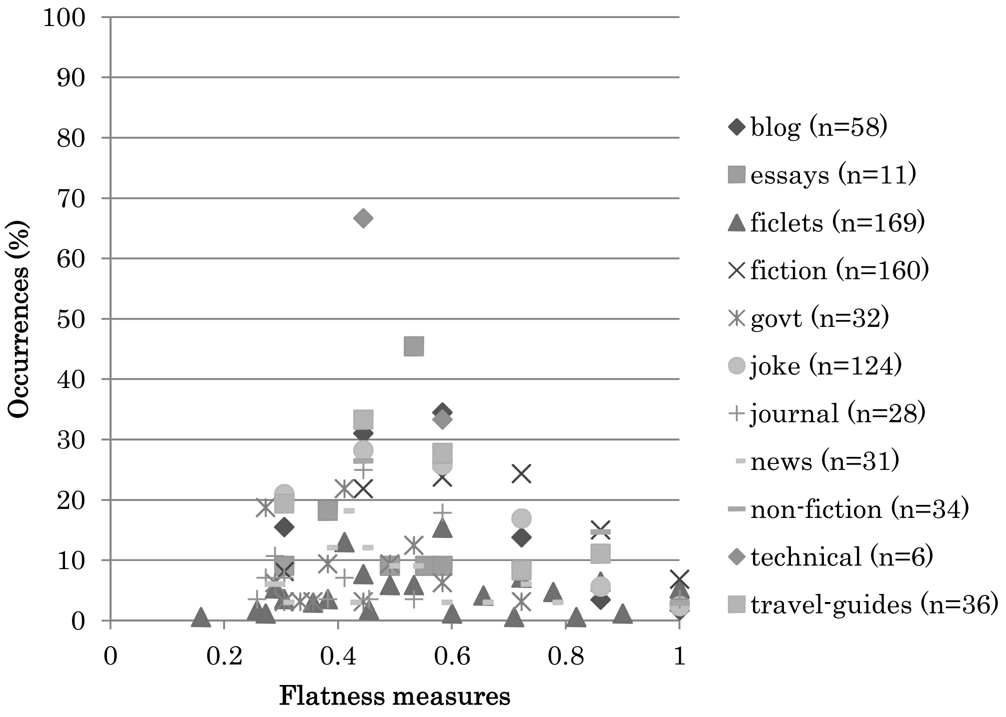


Figure 7.27. The distribution of degree centralities (flatness measures) of 10-word sentences in each genre of MASC500k

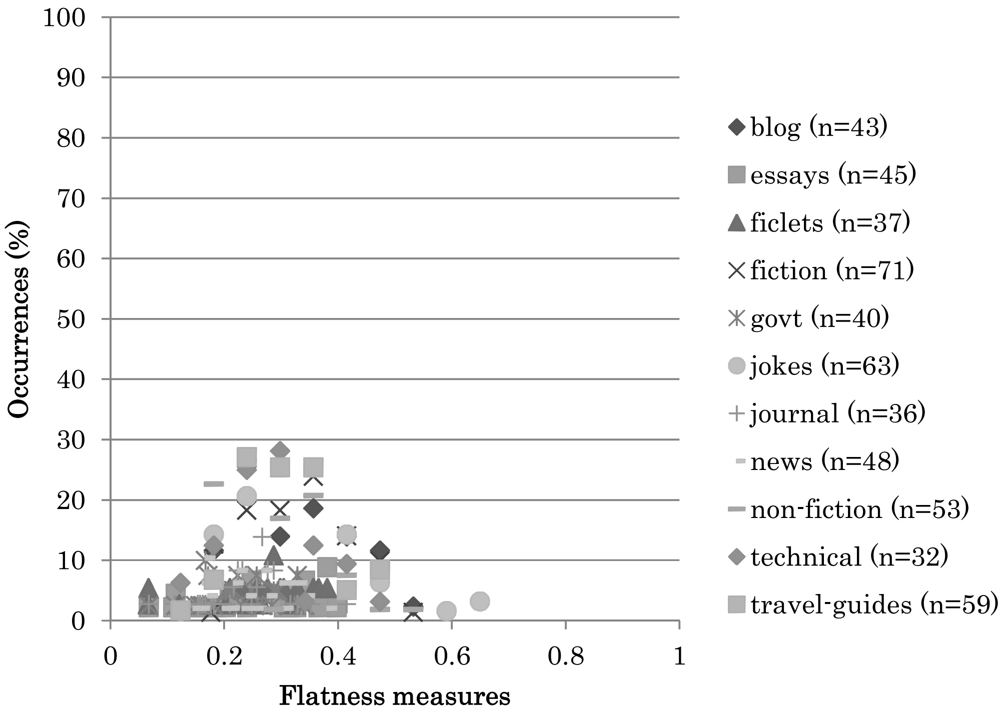


Figure 7.28. The distribution of degree centralities (flatness measures) of 20-word sentences in each genre of MASC500k

We can see that the distribution of degree centralities of 10-word sentences are more varied than that of 20-word sentences. There are only a small number of degree centralities more than 0.5 in the distribution of 20-word sentences.

Sentences of the same word count in different genres show different distributions of degree centralities. We can see the different distributions more explicitly if we concentrate on one particular degree centrality across different genres. For example, as for Fiction (n=160), 39 sentences of all the 10-word sentences have the degree centrality 0.72. This means that approximately 24% of these sentences in Fiction have the degree centrality 0.72. On the other hand, only 2 sentences of all the 10-word sentences in Journal (n=28) have the degree centrality 0.72. This means that approximately 7% of 10-word sentences in Journal have the degree centrality 0.72. Oya (2013a, p.50) points out that "sentences in Fiction tend to be flatter than those in Journal, as far as 10-word sentences in these genres are concerned."

Next, as for 20-word sentences, 17 sentences of all the 20-word sentences in Fiction (n=71) have the degree centrality 0.35, indicating that approximately 24% of these sentences in Fiction have the degree centrality 0.35. On the other hand, only 1 sentence of all the 20-word sentences in Journal (n=35) has the degree centrality 0.35, indicating that approximately 2% of these sentences in Journal have the degree centrality 0.35. Again, Oya (2013a, p.50) points out that "sentences in Fiction tend to be flatter than those in Journal, as far as 20-word sentences are concerned."

7.6.3.2 Distributions of the closeness centralities in MASC500k

This section deals with the distributions of closeness centralities of the sentences of the same word count. The distribution of closeness centralities is somewhat different from that of degree centralities. Figure 7.29 is the distribution of closeness centralities of 10-word sentences, and Figure 7.30 is the distribution of closeness centralities of 20-word sentences.[59]

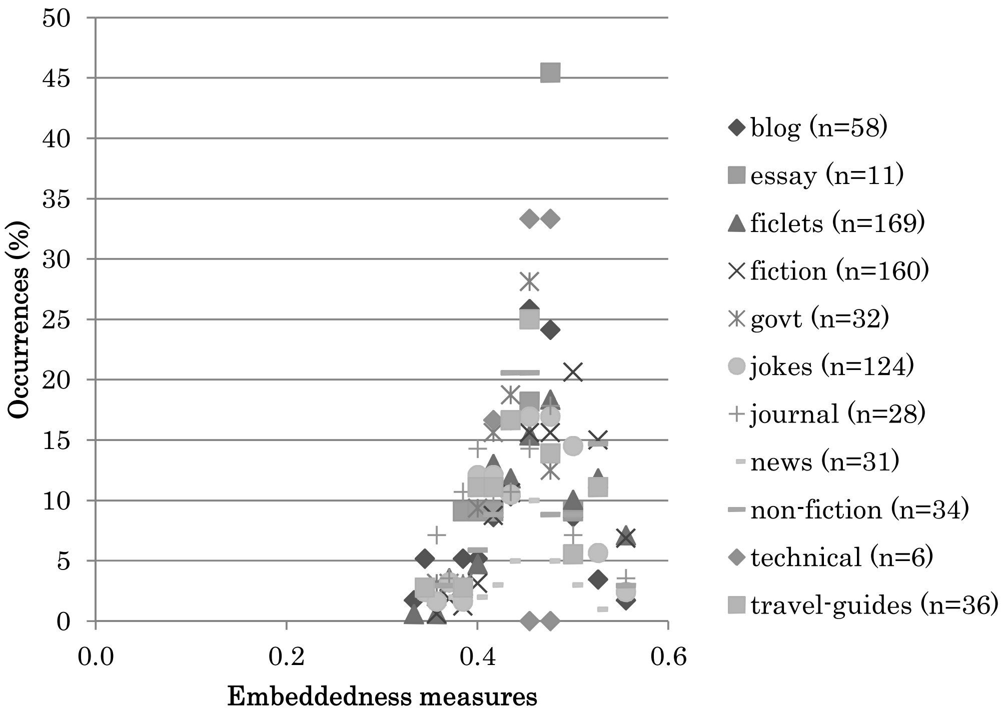


Figure 7.29. The distribution of closeness centralities (embeddedness measures) of 10-word sentences in each genre of MASC500k

[59] The x-axis of these graphs is set with the maximum of 0.6, because there is no embeddedness measure more than 0.6. The y-axis of them is set with the maximum of 50%, because there is no occurrences more than 50%.

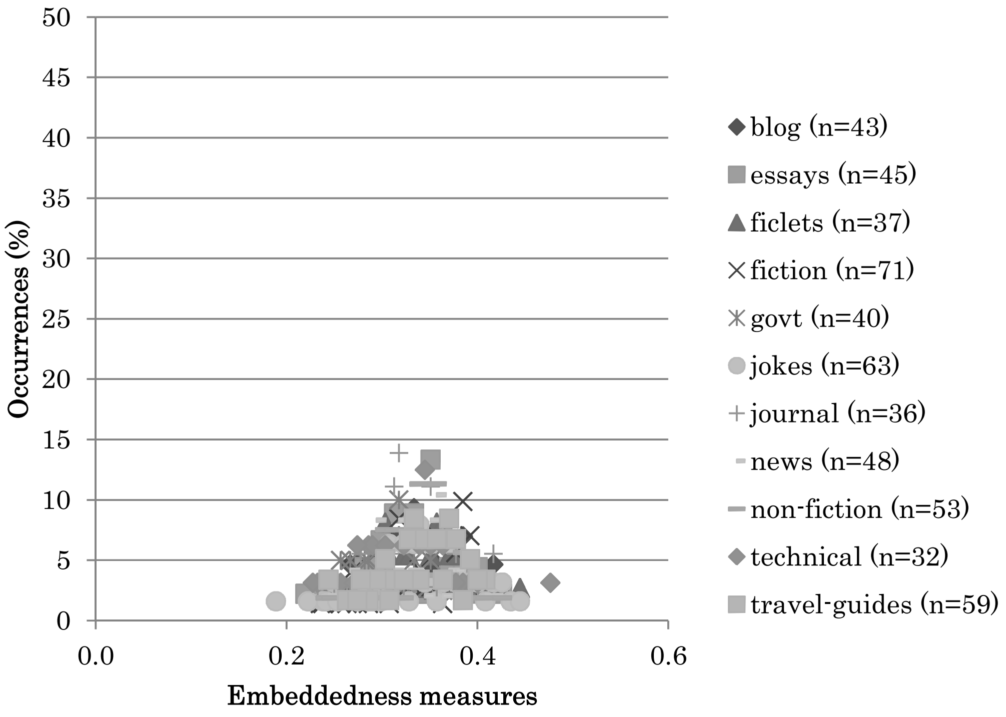


Figure 7.30. The distribution of closeness centralities (embeddedness measures) of 20-word sentences in each genre of MASC500k

We can see that the distribution of closeness centralities of 10-word sentences are more varied than that of 20-word sentences, and we can find no closeness centrality more than 0.5 in the distribution of 20-word sentences. On the other hand, the number of different closeness centralities increases as the word count increases; for example, there are 11 different values of closeness centralities in the 10-word sentences in Fiction, while there are 26 different values of closeness centralities for the 20-word sentences in the same subsection Fiction. The closeness centralities are shown in Table 7.10 below.

Table 7.10. The different values of closeness centrality and the number of sentences which have the same closeness centrality value (Oya 2013a, p.51)

10-word sentences in fiction	
Closeness	frequency
0.3571	1
0.3704	3
0.3846	2
0.4000	5
0.4167	14
0.4348	17
0.4545	25
0.4762	25
0.5000	33
0.5263	24
0.5556	11

20-word sentences in fiction			
Closeness	frequency	Closeness	frequency
0.2353	1	0.3226	2
0.2381	1	0.3333	2
0.2564	1	0.3390	3
0.2632	1	0.3448	2
0.2703	3	0.3509	2
0.2740	1	0.3571	5
0.2857	1	0.3636	1
0.2941	4	0.3704	6
0.2985	2	0.3774	4
0.3030	4	0.3846	7
0.3077	1	0.3922	5
0.3125	2	0.4000	2
0.3175	6	0.4082	2

As is the case in degree centralities, sentences of the same word count in different genres show different distributions of closeness centralities. As for the 10-word sentences, Fiction has 33 10-word sentences with the closeness centrality 0.384 out of 160 (approximately 20%), while Journal has 2 10-word sentences with the closeness centrality 0.384 out of 28 (approximately 10%). Oya (2013a, p.52) points out that "sentences in Journal tend to be more embedded than those in Fiction" as far as 10-word sentences are concerned. As for the closeness centralities of 20-word sentences, we cannot find the same closeness centrality both in Fiction and in Journal.

7.6.4 Discussion

The distributions of degree centralities and those of closeness centralities among different genres of texts suggest that both of the centrality measures of a sentence are dependent on the word count of the sentence. The larger number of words a sentence has, the smaller degree centralities and closeness centralities, and more diverse values of closeness centralities. This result may indicate that both of these centrality measures cannot show the difference in genre by themselves. However, Oya (2013a, p.52) argued that the difference in genre can be reflected on the number of sentences of the same degree centrality and of the same closeness centrality if we control the word count of the sentences taken from different genres, as is the case in Journal and Fiction. In order to have a broader understanding of the distributions of degree centralities and closeness centralities, we need to explore their differences across different word counts and different genres, which will be the research topic in future.

7.7 Summary

This chapter attempted to answer the question: *from which source are the graph centrality measures obtained, and what is the result?* Section 7.1 introduced the three issues that this study focuses on: (1) the difference between the centrality measures of the parsed output of sentences and those of the correct typed-dependency trees for the same sentences; (2) the difference between the centrality measures of the parsed output for English sentences and those of the parsed output for Japanese counterparts; (3) the difference among centrality measures of the parsed output for English sentences in different genres of texts. Section 7.2 summarized a number of features which can be extracted from the typed-dependency tree for a sentence.

Section 7.3 addressed the issue of parsing accuracy for English sentences, and it was shown that the Stanford Parser was sufficiently accurate to obtain both degree centralities and closeness centralities of English sentences. Section 7.4 addressed the issue of parsing accuracy for Japanese sentences, and it was shown that KNP was also sufficiently accurate to obtain those of Japanese sentences. Section 7.5 addressed the issue of cross-linguistic differences of centrality measures of English-Japanese sentence pairs, in order to see how these centrality measures reflect their structural differences and similarities in terms of flatness and embeddedness. The result of comparing the degree centralities of English sentences and those of Japanese counterparts in the small-scale parallel corpus showed that the typed-dependency trees for

English sentences tend to have more varied structural settings than those for Japanese sentences in terms of their flatness. Section 7.6 addressed the issue of intra-language variations of centrality measures using a large-scale corpus (MASC 500k). The distributions of degree centralities and of closeness centralities obtained from the parsed output of sentences from different genres of texts in MASC 500k are compared to each other. It is shown that sentences from different genres have different distributions of these measures; sentences in the subsections Fiction, Ficlets and Jokes are flatter and more embedded than sentences in other subsections. However, it is pointed out that these different distributions were dependent on the word counts of the sentences. It was also pointed out that controlling the word count of the sentences taken from different genres could make explicit that difference in genre is reflected on the number of sentences of the same degree centrality and of the same closeness centrality.

8. Conclusion

Chapter 1 stated the aim of this thesis: to introduce the typed-dependency trees for English and Japanese sentences, and to introduce graph-centrality measures to capture the structural characteristics of these typed-dependency trees. Typed-dependency trees are syntactic structures for sentences that illustrate the dependency relationships among the words in a sentence as a network of words. The structural characteristics of the network can be captured by a number of measures that have been developed in the field of graph theory and network analysis. Introducing these measures into the typed-dependency trees for sentences allows us to capture the structural characteristics of these sentences as networks of words, and ultimately, this analysis sheds new light on Japanese and English speakers' syntactic intuitions.

Chapter 1 also raised the following four questions to be answered in order to accomplish the aim: (1) *What are the dependency relationships among the words in a sentence*? (2) *What are the graph centrality measures, and how are they calculated?* (3) *How can we obtain the typed-dependency trees for given sentences, and what are their characteristics*? And (4) *From which source are the graph centrality measures obtained, and what is the result?*

Chapter 2 and 3 attempted to answer the question: *what are the dependency relationships among the words in a sentence*? Chapter 2 introduced the concept of dependency grammar through a discussion of Tesnière's (1959) seminal assumption about dependency along with more recent theories of dependency grammar proposed by I. Mel'čuk and his colleagues (Iordanskaja & Mel'čuk 2000: Mel'čuk & Pertsov 1987; Mel'čuk 1988; Mel'čuk 2003; Mel'čuk 2004; Mel'čuk 2009; Mel'čuk 2011). This chapter also addressed the difference between dependency grammar and phrase-structure grammar. Section 2.2 presented an overview of dependency grammar and Section 2.3 focused specifically on Tesnière's (1959) seminal work on dependency grammar. Section 2.4 discussed Mel'čuk's work on Deep Syntactic Relations and Surface

Syntactic Relations as a development of Tesnière's (1959) concept of dependency. Finally, the difference between dependency and phrase-structure grammar was briefly discussed in Section 2.5 with reference to Osborne et al. (2011).

Chapter 3 examined whether a typed-dependency tree for a sentence is equivalent to a functional-structure representation according to Lexical-Functional Grammar (LFG) (Bresnan 1978; Bresnan 1982; Kaplan & Bresnan 1982). The basic architecture of the framework of LFG was summarized in Section 3.2, with emphasis on structural correspondence between the constituent structure and the functional structure for a sentence, and on the well-formedness constraints for a functional structure. Next, Section 3.3 showed that a functional-structure representation for a sentence and its typed-dependency tree are equivalent. This equivalence supports the idea that LFG represents one direction of development of the dependency grammar. This section also introduced the idea that Mel'čuk's Criterion A for the existence of dependency relationship between two words can be revised in terms of their possibility to constitute a fragment functional structure.

Chapter 4 attempted to answer the question: *what are the graph centrality measures, and how are they calculated?* This chapter continued the discussion of typed-dependency by exploring the representation of a typed-dependency syntactic tree as a directed acyclic graph (DAG). Moreover, this chapter also examined the idea of quantifying the structural property in terms of graph centrality. The advantage of dependency grammar representation is that a sentence's dependency can be interpreted as a DAG, allowing the formal syntactic properties to be defined and analyzed mathematically in terms of graph theory (Oya 2010b, Oya 2011). Section 4.2 introduced the basic tenets of graph theory. Section 4.3 examined centrality measures, including degree centrality and closeness centrality. The process for employing these centrality measures to analyze structural properties of typed-dependency trees was discussed in Sections

4.4 and 4.5. Specifically, Section 4.4 explored the application of centrality measures to show the similarity of functional-structure representations, and Section 4.5 illustrated how stylistic differences across genres are reflected by different distributions of centralities. Finally, Section 4.6 addressed the role of dependency distance in these representations.

Chapter 5 and 6 attempted to answer the following question: *how can we obtain the typed-dependency trees for given sentences, and what are their characteristics*? Chapter 5 introduced Stanford Parser (de Marneffe & Manning 2012), which is a state-of-the-art parser used in this study for acquiring typed-dependency tree representations for English sentences. Section 5.2 described the output format of the Stanford Parser. Section 5.3 provided the definition of each dependency type used in the parsed output of the Stanford Parser, with reference to the criteria for surface syntactic relations by Mel'čuk (2009, 2011), along with example sentences for each of the dependency types, their typed dependency trees, and the functional structure representations equivalent to these trees. Section 5.4 explained the differences among the different output styles of the Stanford Parser.

Chapter 6 introduced KNP (Kurohashi & Nagao 1992, 1994, 1998; Kawahara & Kurohashi 2007), which is a rule-based dependency parser used for generating automatic typed-dependency tree representations for Japanese sentences. Section 6.2 briefly introduced KNP and its output format. Section 6.3 described the process through which KNP parsed output is annotated with dependency types. Section 6.4 dealt with zero pronouns in elliptic sentences often used in Japanese. In section 6.5, each dependency type used with this parser in this study was defined with reference to the criteria for surface syntactic relations by Mel'čuk (2009, 2011), along with example sentences for each of the dependency types, their typed dependency trees, and the functional structure representations equivalent to these trees.

In Chapter 5 and 6, each of the dependency types of English and Japanese is given a theoretical backbone based on Mel'čuk's Criteria, in other words, (1) which word depends on which (Mel'čuk's Criterion A), (2) which word is the governor (or the head) of a given dependency (Mel'čuk's Criterion B), and (3) what is the name of a given dependency (Mel'čuk's Criterion C). This result implies that the study of dependency in other languages will also be further developed along the line proposed in this study, which will be the topic of further study.

In Chapter 5 and 6, the typed-dependency trees of English and Japanese are shown to be equivalent to functional-structure representation in the framework of LFG. It can be pointed out that the functional structures in this study sometimes deviate from the orthodox functional-structure representation in terms of the issue of control (see Section 5.3.3 on the dependency type "xcomp"), subcategorization (see some of the functional-structure representation in Section 5.3.3), or zero pronouns (see Section 6.4). The study of both dependency grammar and LFG will be further developed if the gap between these two frameworks is bridged, and we should not exclude the possibility at this moment that they eventually unite to establish a new theoretical framework, with the long tradition and intuitive appeal of dependency grammar, and with the computer-friendliness and the richness of cross-linguistic study of LFG.

Chapter 7 attempted to answer the following question: *from which source are the graph centrality measures obtained, and what is the result?* The accuracies of the Stanford Parser and the KNP were examined by comparing the typed-dependency trees obtained from the parsed output of the English sentences and their Japanese counterparts in a small-scale parallel corpus (Iida 2010) to their manually corrected typed-dependency trees. Results of the comparison showed that the distributions of both degree centralities and closeness centralities before and after manual corrections were almost identical. Thus, the Stanford Parser and KNP are accurate enough to obtain degree centralities and closeness centralities. Next, the distributions of degree and closeness centralities for English typed-dependency trees were compared to those for their Japanese counterparts, and results showed that their distributions were different. Thus, the structural properties of the typed-dependency trees for sentences in these two languages are different in terms of their degree centralities (flatness) and closeness centralities (embeddedness). Lastly, the distributions of degree centralities and of closeness centralities obtained from the parsed output of sentences from different genres texts in Manually annotated sub-corpus of American National Corpus (MASC 500k) were compared to each other, and it was shown that their distributions were different; sentences in the subsections Fiction, Ficlets and Jokes are flatter and more embedded than sentences in other subsections. However, it is pointed out that these different distributions could be dependent on the word counts of the sentences. It was also pointed out that controlling the word count of the sentences taken from different genres could make explicit that the difference in genre is reflected on the number of sentences of the same degree centrality and of the same closeness centrality.

This dissertation attempted to show that graph centrality measures (degree centralities and closeness centralities) can be used to show the structural properties of typed-dependency trees of English and Japanese sentences. The results of corpus analyses in Chapter 7 suggest that these centrality measures need further improvement. These measures, however, open the possibility to capture the structural property of typed-dependency trees of English and Japanese sentences in a more objective manner. As far as the typed-dependency tree of a sentence can be represented as a directed acyclic graph, the graph centrality measures other than degree centrality or closeness centrality can be used to show the structural properties of a sentence more objectively than linguists' intuition can. Applying other graph centrality measures will be one of the research issues in future.

References

Adams, R., Nicolae, G., Nicolae, C. & Harabagiu, S. (2007). Textual entailment through extended lexical overlap and lexico-semantic matching. *Proceedings of the ACLPASCAL Workshop on Textual Entailment and Paraphrasing*, 119-124, Prague, June 2007. Retrieved from http://delivery.acm.org/10.1145/1660000/1654560/p119-adams.pdf?ip=133.9.4.12&acc=OPEN&CFID=147639811&CFTOKEN=80564240&__acm__=1346117205_d96e6b8d56d9bad7c975c3ab305b9d81

Batagelj, V. & Mrvar, A. (1996). *Pajek: Program for large network analysis.*

Beauchamp, M.A. (1965). An improved index of centrality. *Behavioral Science. 10*, 161-163.

Biggs, N. L., Lloyd, E. K. & Wilson, R. J. (1999). *Graph theory 1736-1936*. Oxford, UK: Oxford University Press.

Bresnan, J. (1978). A realistic transformational grammar. In M. Halle, J. Bresnan, & G.A. Miller (Eds), *Linguistic theory and psychological reality* (pp.1-59). Cambrige, MA: The MIT Press.

Bresnan, J. (ed.) (1982). *The mental representation of grammatical relations*. Cambridge, MA: The MIT Press.

Bresnan, J. (2001). *Lexical-Functional Syntax.* Oxford, UK: Blackwell.

Buchholz, S. & Marsi, E. (2006). Conll-X shared task on multilingual dependency parsing. *Proceedings of CONLL-X*. 149-164.

Butt, M., Nino, M. E., & Segond, F. (1996, September). Multilingual processing of auxiliaries within LFG. In *KONVENS* (pp. 111-122).

Butt, M., King, T. H., Niño, M. E. & Segundo, F. (1999). *A grammar writer's cookbook*. Stanford, CA: CSLI Publications.

Cer, D., de Marneffe, M., Jurafsky, D. & Manning, C. (2010). Parsing to Stanford Dependencies: trade-offs between speed and accuracy. *7th International Conference on Language Resources and Evaluation*. Retrieved from http://nlp.stanford.edu/pubs/lrecstanforddeps_final_final.pdf.

Chafe, W. L. (1970). *Meaning and the structure of language*. Chicago, IL: The University of Chicago Press.

Charniak, E. (2000). A maximum-entropy-inspired parser. *NAACL 2000 Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*, 132-139.

Chomsky, N. (1981). *Lectures on Government and Binding: The Pisa lectures*. Mouton de Gruyter.

Chomsky, N. (1995). *The Minimalist Program.* Cambridge, MA: MIT Press.

Chomsky, N. (1999). *Derivation by phrase*. Cambridge, MA: MIT Press.

Collins, M. (1996). A new statistical parser based on bigram lexical dependencies. *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*, 184-191.

Collins, M. (1999). Head-driven statistical models for natural language parsing. Ph.D. thesis, University of Pennsylvania. Retrieved from http://www.dfki.de/~neumann/dop-seminar/References/collins-thesis.pdf

Davies, W.D. & Dubinsky, S. (2004). *The grammar of raising and control.* Blackwell.

Dalrymple, M. (2001). *Syntax and semantics: Lexical Functional Grammar*. San Diego, CA : Academic Press.

Dalrymple, M., Dyvik, H., & King, T. H. (2004). Copular complements: closed or open? *Proceedings of the LFG04 Conference*. 188-198.

Debusmann, R. (2003). Dependency Grammar as graph description. *Prospects and Advances in the Syntax-Semantics Interface*, Nancy. Retrieved from http://www.ps.uni-saarland.de/~rade/papers/passi03.pdf

Debusmann, R. & Kuhlmann, M. (2007). *Dependency Grammar: Classification and exploration. Project report (CHORUS, SFB 378)*. Retrieved from http://www.ps.uni-saarland.de/~rade/papers/sfb.pdf

De Marneffe, M. C., MacCartney, B., & Manning, C. D. (2006). Generating typed dependency parses from phrase structure parses. In *LREC 2006*.

De Marneffe, M. C., & Manning, C. D. (2008). The Stanford typed dependencies representation. *COLING Workshop on Cross-framework and Cross-domain Parser Evaluation*. Retrieved from http://nlp.stanford.edu/pubs/dependencies-coling08.pdf.

De Marneffe, M. C. & Manning, C. D. (2012). *Stanford Typed Dependency Manual Revised for the Stanford Parser v.2.0.4*. Retrieved from http://nlp.stanford.edu/software/dependencies_manual.pdf.

De Marneffe, M.C. & Manning, C. D. (2013). *Stanford Typed Dependency Manual Revised for the Stanford Parser v.3.2 in June 2013.* Retrieved from http://nlp.stanford.edu/software/dependencies_manual.pdf.

De Nooy, W., Mrvar, A., & Batagelj, V. (2005). *Exploratory network analysis with Pajek*. Cambridge, UK: Cambridge University Press.

Ding, Y., & Palmer, M. (2004a). Automatic learning of parallel dependency treelet pairs. *First International Joint Conference on NLP (IJCNLP-04).* Retrieved from http://pdf.aminer.org/000/391/911/automatic_learning_of_parallel_dependency_treelet_pairs.pdf

Ding, Y., & Palmer, M. (2004b). Synchronous dependency insertion grammars: A grammar formalism for syntax based statistical MT. *Workshop on Recent Advances in Dependency Grammars, COLING-04*. Retrieved from http://acl.ldc.upenn.edu/W/W04/W04-1513.pdf

Ding, Y., & Palmer, M. (2005). Machine translation using probabilistic synchronous dependency insertion grammars. *Proceedings of the 43rd Annual Meeting of the ACL, 541-548.* Retrieved from http://acl.ldc.upenn.edu/P/P05/P05-1067.pdf

Falk, Y. (1984). The English auxiliary system: A Lexical-Functional analysis. *Language* 60. 483-509.

Frazer, J. (1912). *The Golden Bough.* In Wikisource, The Free Library. Retrieved from http://en.wikisource.org/w/index.php?title=The_Golden_Bough&oldid=92983

Freeman, L. (1979). Centrality in social networks. *Social Networks vol.1*, 215-239.

Genzel, D. (2010). Automatically learning source-side reordering rules for large scale machine translation. In *COLING-2010*.

Gerdes, K., & Kahane, S. (2011). Defining dependencies (and constituents). *Proceedings of International Conference on Dependency Grammar*, 17-27. Retrieved from http://depling.org/proceedingsDepling2011/

Gerdes, K., Hajičová, E., & Wanner, L. (eds.). (2011). *Proceedings of International Conference on Dependency Grammar*. Retrieved from http://depling.org/proceedingsDepling2011/

Gibson, E. (1998). Linguistic complexity: locality of syntactic dependencies. *Cognition*, 68, 1-76.

Gibson, E. (2000). Dependency locality theory: A distance-based theory of linguistic complexity. In A. Marantz, Y. Miyashita, & W. O'Neil (Eds.), *Image, language, brain: Papers from the first mind articulation project symposium*. 95-126. Cambridge, MA: MIT Press.

Grimshaw, J. (1990). *Argument Structure*. MIT Press.

Hall, E. (1976). *Beyond Culture*. Knopf Doubleday Publishing Group.

Halliday, M.A.K. (1967). Notes on transitivity and theme in English Part I. *Journal of Linguistics, vol.3, Issue 1*. 37-81.

Hashimoto, S. (1948). *Kokugogaku Kenkyu* "*A Study of Japanese Language*." Tokyo, Japan: Iwanami Shoten.

Horáček, P., Zámečníková, E., & Burgetová, I. (2011). *Dependency Grammars* [PowerPoint slides]. Retrieved from http://147.229.9.23/units/UIFS/grants/index.php?file=%2Fproj%2F533%2FPrintable%2Ffmnl08-dependency-handout.pdf&id=533

Ide, N., & Suderman, K. (2004). The American National Corpus first release. *Proceedings of the Fourth Language Resources and Evaluation Conference (LREC)*, 1681-84.

Ide, N., Baker, C., Fellbaum, C., Fillmore, C., & Passonnau, R. (2008). MASC: The Manually Annotated Sub-Corpus of American English. *Proceedings of the Eighth Language Resources and Evaluation Conference (LREC),* 2455-2460.

Iida, Y. (2010). *Eisakubun Kihon 300 Sen* "Basic 300 Sentences for English Composition." Tokyo, Japan: Sundai Publishing.

Iordanskaja, L., & Mel'čuk, I. (2000). The Notion of Surface-Syntactic Relation Revisited (Valence-Controlled Surface-Syntactic Relations in French). In L.L. Iomdin and L.P. Krysin (eds.) 2000, 391-433. Retrieved from http://olst.ling.umontreal.ca/pdf/IordanskajaMelcukSSyntRels.pdf

Kanatani, T. (2002). *Nihongoni Shugowa Iranai,* "*Japanese Language does not require subjects*." Tokyo, Japan: Kodansha.

Kaplan, R. & Bresnan, J. (1982). Lexical-Functional Grammar: A formal system for grammatical representation. In J. Bresnan (ed.), *The mental representation of grammatical relations* (pp.173-281). Cambridge, MA: The MIT Press.

Kawahara, D. & Kurohashi, S. (2007). Daikibokakuframeni modozuku koubun kakukaisekino tougouteki kakuritumoderu “An integrated probabilistic model for syntactic and case analyses based on a large-scale case frames.” *Natural Language Processing* vol. 14, no.4, 67-81.

Kessler, J.S. (2008). Polling the blogosphere: a rule-based approach to belief classification. In *International Conference on Weblogs and Social Media*, 2008. Retrieved from http://www.aaai.org/Papers/ICWSM/2008/ICWSM08-016.pdf

Klein, D. & Manning, C.D. (2003). Accurate unlexicalized parsing. *Proceedings of the 41st Meeting of the Association for Computational Linguistics*, 423-430.

Kuno, S. (1972). Functional sentence perspective: A case study from Japanese and English. *Linguistic Inquiry*, 3(3): 269-320.

Kurohashi, S. & Nagao, M. (1992). A method for analyzing conjunctive structures in Japanese. *Journal of Information Processing Society of Japan* vol.33, no. 8. 1022-1031.

Kurohashi, S. & Nagao, M. (1994). A syntactic analysis method of long Japanese sentences based on coordinate structure detection. *Journal of Natural Language Processing vol.1, no.1*. 35-57.

Kurohashi, S. & Nagao, M. (1998). Building a Japanese parsed corpus while improving the parsing system. *Proceedings of the 1st International Conference on Language Resources and Evaluation*. 719-724.

Lin, D. (1998). Dependency-based evaluation of MINIPAR. In Workshop on the Evaluation of Parsing Systems, Granada, Spain.

Lu, X. (2010). Automatic analysis of syntactic complexity in second language writing.

International Journal of Corpus Linguistics, 15:4, 474-496.

Manning, C. D. & Schütze, H. (1999). *Foundations of statistical natural language processing*. Cambridge, MA: MIT Press.

Matsumoto, Y. (1996). *Complex predicates in Japanese: A syntactic and semantic study of the notion 'word.'* Stanford. CA: CSLI Publications and Tokyo: Kuroshio Publishers.

Masuichi, H., & Ohkuma, T. (2003). Constructing a practical Japanese parser based on Lexical-Functional Grammar. *Journal of Natural Language Processing*, 10(2), 79-109.

Masuoka, T. & Takubo, Y. (1992). *Kiso Nihongo Bumpo* "Basic Japanese Grammar." Tokyo, Japan: Kuroshio Publishers.

McDonald, R & Nivre, J. (2011). Analyzing and integrating dependency parsing. *Computational Linguistics. Vol.37, No.1*. 197-230.

McDonald, R., Pereira, F., Ribarov, K. & Hajič, J. (2005). Non-projective dependency parsing using spanning tree algorithms. *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, 523-530. Retrieved from http://acl.ldc.upenn.edu/H/H05/H05-1066.pdf

Mel'čuk, I. (1988). *Dependency syntax: theory and practice.*Albany, NY: The SUNY Press.

Mel'čuk, I. (2003). Levels of dependency in linguistic description: Concepts and problems. In V. Agel, L. Eichinnger, H.-W. Eroms, P. Hellwig, H.J. Herringer, & H. Lobin (eds), *Dependency and valency: An international handbook of contemporary research, vol. 1* (pp. 188-229). Berlin - New York: W. de Gruyter.

Mel'čuk, I. (2004). Actants in semantics and syntax. I,II, *Linguistics*, 42:1,1-66;42:2,247-291.

Mel'čuk, I. (2009). Dependency in natural language. In A. Polguère & I. Mel'čuk (eds), *Dependency in linguistic description* (pp.1-110). Amsterdam: Benjamins.

Mel’čuk, I. (2011). Dependency in language-2011. *Proceedings of International Conference on Dependency Linguistics*, 1-16.

Mel’čuk, I. & Pertsov, N. (1987). *Surface syntax of English: A formal model within the Meaning-Text framework*. Amsterdam: Benjamins.

Mikami, A. (1972). *Zoku Gendaigohou Josetsu* “*Introduction to Modern Japanese Grammar vol.2*.” Tokyo: Kuroshio Publishers.

Mohanan, K. P. (1983). Functional and anaphoric control. *Linguistic Inquiry* vol.14, No.4. 641-674.

Nakano, M. (1998). *Kihon Eisakubun Text* “A text for basic English writing.” Tokyo, Japan: Kenkyusha.

Nivre, J., Hall, J., Kübler, S., McDonald, R., Nilsson, J., Riedel, S., & Yuret, D. (2007). The CoNLL 2007 shared task on dependency parsing. *Proceedings of the Conference on Empirical Methods in Natural Language Processing and Conference on Computational Natural Language Learning (EMNLP-CoNLL)*. 915-932.

Nomura, M., & Koike, K. (1992). *Nihongo Jiten,* “*A Dictionary of Japanese Language*.” Tokyo, Japan: Tokyodo Shuppan.

Ono, S. & Nakagawa, H. (1997). Semantics of Japanese sentence final particles presented by hierarchical memory model: about YO, NE, NA, ZE and ZO. *Cognitive Studies: Bulletin of the Japanese Cognitive Science Society* 4(2), 39-57.

Osuga, T., Horiuchi, Y., & Ichikawa, A. (2003). Estimating syntactic structure from prosody in Japanese speech. *IEICE TRANSACTIONS on Information and Systems. Vol.E86-D No.*3, 558-564. Retrieved from http://search.ieice.org/bin/pdf.php?lang=E&year=2003&fname=e86-d_3_558&abst=

Osborne, T., Putnam, M., & Gross, T.M. (2011). Bare phrase structure, label-less trees, and specifier-less syntax: Is Minimalism becoming a Dependency Grammar? *The Linguistic Review* 28, 315-364.

Oya, M. (2004). On the properties of Japanese case marker '-ga': An LFG account. *Proceedings of the 8th Conference of Pan-Pacific Association of Applied Linguistics*. 349-360.

Oya, M. (2009). A method of automatic acquisition of typed-dependency representation of Japanese syntactic structure. *Proceedings of the 14th Conference of Pan-Pacific Association of Applied Linguistics*. 337-340.

Oya, M. (2010a). Treebank-based automatic acquisition of wide coverage, deep linguistic resources for Japanese. M.Sc. thesis, School of Computing, Dublin City University.

Oya, M. (2010b). Directed acyclic graph representation of grammatical knowledge and its application for calculating sentence complexity. *Proceedings of the 15th International Conference of Pan-Pacific Association of Applied Linguistics,* 393-400.

Oya, M. (2011). Syntactic dependency distance as sentence complexity measure. *Proceedings of the 16th Conference of Pan-Pacific Association of Applied Linguistics.* 313-316.

Oya, M. (2012). Degree centralities, closeness centralities, and dependency distances of different genres of texts. *Proceedings of the 17th International Conference of Pan-Pacific Association of Applied Linguistics.* 89-90.

Oya, M. (2013a). Degree centralities, closeness centralities, and dependency distances of different genres of texts. *Selected Papers from the 17th International Conference of Pan-Pacific Association of Applied Linguistics*, 42-53.

Oya, M. (2013b). Syntactic dependency structures of English and Japanese. *Mejiro Journal of Humanities vol.9*, 151-164.

Oya, M. (2013c). Lexical-Functional Grammar and Dependency Grammar. *Information Communication Technology Practice & Research 2012*, 19-32.

Oya, M. (2013d). Treatment of zero pronouns in the framework of Lexical-Functional Grammar. *Proceedings of the 18th International Conference of Pan-Pacific Association of Applied Linguistics.*

Oya, M. (2014). Treatment of zero pronouns in the framework of Lexical-Functional Grammar. *Selected Papers from the 18th International Conference of Pan-Pacific Association of Applied Linguistics.*

Palmer, F. R. (2001). *Mood and modality (2nd edition)*. Cambridge, UK: Cambridge University Press.

Polguère, A. & Mel'čuk, I. (eds). (2009). *Dependency in linguistic description*. Amsterdam: Benjamins.

Prince, E. (1981). Topicalization, focus-movement, and Yiddish-movement: A pragmatic differentiation. *Proceedings of the Seventh Annual Meeting of the Berkeley Linguistics Society,* 249-264.

Prince, E.F. (1998). On the limits of syntax, with reference to Topicalization and Left-Dislocation. In P. Culicover & L. McNally (Eds.), *Syntax and Semantics 29: The limits of syntax* (pp. 281-302). NY: Academic Press.

Pyysalo, S., Ginter, F., Haverinen, K., Heimonen, J., Salakoski, T., & Laippala, V. (2007). On the unification of syntactic annotations under the Stanford dependency scheme: A case study on BioInfer and GENIA. *Proceedings of BioNLP 2007: Biological, Translational, and Clinical Language Processing (ACL07),* 2007. Retrieved from http://delivery.acm.org/10.1145/1580000/1572397/p25-pyysalo.pdf?ip=133.9.4.12&acc=OPE

N&CFID=147639811&CFTOKEN=80564240&__acm__=1346117950_df739b58de8da5c0e6455c868dcb47be

Quirk, R., Greenbaum, S., Leech, G., & Svartvik, J. (1985). *A Comprehensive Grammar of the English language.* London: Longman.

Sabidussi, G. (1966). Then centrality index of a graph. *Psychometrika.* 31, 581-603.

Sadler, L. & Spencer, A. (2004). *Projecting morphology*. Stanford, California: CSLI Publications.

Sag, I. (1976). *Deletion and logical form.* Ph.D. dissertation, MIT.

Sag, I., Kaplan, R., Karttunen, L., Kay, M., Pollard, C., Shieber, S., & Zaenen, A. (1986). Unification and grammatical theory. *Proceedings of the West Coast Conference on Formal Linguistics*, 238-254. Stanford Linguistics Association, Stanford University. Retrieved from http://lingo.stanford.edu/sag/papers/sagetal-1986.pdf.

Scott, J.(ed.). (2002). *Social networks vol.1*. London: Routridge.

Sleator, D. D., & Temperley, D. (1993). Parsing English with a link grammar. *Third International Workshop on Parsing Technologies*.

Snow, R., Jurafsky, D., & Ng, A. (2005). Learning syntactic patterns for automatic hypernym discovery. *Proceedings of NIPS 17.* Retrieved from http://www.stanford.edu/~jurafsky/paper887.pdf

Temperley, D. (2007). Minimization of dependency length in written English. *Cognition*. 105, 300-333.

Teramura, H. (1992). *Teramura Hideo Ronbunshu I* "*The Treatises of Hideo Teramura vol.1*." Tokyo, Japan: Kuroshio Publishers.

Tesnière, L. (1959). *Éléments de syntaxe structural*. Paris: Klincksieck.

Tomioka, S. (2003). The semantics of Japanese null pronouns and its cross-linguistic implications. In K. Schwabe & S. Winkler (Eds.), *The Interfaces: Deriving and Interpreting Omitted Structures* (pp. 321-339). Amsterdam; Philadelphia: J. Benjamins.

Toyama, S. (1973). *Nihongono Ronri "The Logic of Japanese Language*." Tokyo: Chuo Koronsha.

Tsujimura, N. (2007). *An introduction to Japanese linguistics (2nd ed.).* Oxford: Blackwell.

Valdman, A. & Gass, S. (eds.). (2010). *Studies in Second Language Acquisition.* Retrieved from http://journals.cambridge.org/action/displayJournal?jid=SLA

Wasserman, S. & Faust, K. (1994). *Social network analysis*. Cambridge, UK: Cambridge University Press.

Wilson, R.J. (1975). *Introduction to Graph Theory*. London: Longman

Yoshida, S., Kumaki, H. & Watanabe, A. (2009). Range analysis of Cross-Cultural Distance Learning (CCDL) reflection work sheet: A pilot study. *Proceedings of the 14th Conference of Applied Linguistics*, 351- 356.

Zeman, D. & Žaborkrtský, Z. (2005). Improving parsing accuracy by combining diverse dependency parsers. *Proceedings of the Ninth International Workshop on Parsing Technologies (IWPT)*, 171-178.

Appendix I: The Hierarchy of Dependency Types (de Marneffe & Manning 2012)

- **dep** - dependent
    - **aux** - auxiliary
        - **auxpass** - passive auxiliary
        - **cop** - copula
    - **arg** - argument
        - (**agent** – agent)
        - **comp** - complement
            - **acomp** - adjectival complement
            - **attr** - attributive
            - **ccomp** - clausal complement with internal subject
            - **xcomp** - clausal complement with external subject
            - **pcomp** – clausal complement of a preposition
            - **compl** - complementizer
            - **obj** - object
                - **dobj** - direct object
                - **iobj** - indirect object
                - **pobj** - object of preposition
        - **subj** - subject
            - **nsubj** - nominal subject
            - **nsubjpass** - passive nominal subject
            - **csubj** - clausal subject
            - **csubjpass** - passive clausal subject
    - **cc** - coordination
    - **conj** - conjunct
    - **expl** - expletive (expletive “there”)
    - **mod** - modifier
        - **neg** - negation modifier

det - determiner
prep - prepositional modifier
amod - adjectival modifier
advmod - adverbial modifier
poss - possession modifier
possessive - possessive modifier ('s)
infmod - infinitival modifier
partmod - participial modifier
advcl - adverbial clause modifier
mark - marker (word introducing an adverbial clause)
rcmod - relative clause modifier
rel - relative (word introducing a relative clause modifier)
purpcl - purpose clause modifier
prt - phrasal verb particle
predet - predeterminer
preconj - preconjunct
mwe – multi-word modifier
quantmod - quantifier modifier
tmod - temporal modifier
measure - measure-phrase modifier
nn - noun compound modifier
npadvmod – noun phrase adverbial modifier
num - numeric modifier
number - element of compound number
abbrev - abbreviation modifier
appos - appositional modifier

parataxis - parataxis

punct - punctuation

ref – referent

root – the root of the sentence

(**sdep** - semantic dependent)

(**xsubj** - controlling subject)

Appendix II: The list of incorrect parses for English sentences in Basic 300 (Iida 2010) by Stanford Parser

10eng.: “Those oranges taste too sour, so I don’t like them.”

The word “so” is incorrectly parsed to depend on “taste” with an incorrect type “dep.” The correct parse is that it depends on “like” with the type “mark.”

17eng.: “I’m sure a cup of coffee will wake you up.”

The word “wake” is parsed to depend on “sure” with an incorrect type “dep.” The correct type is “ccomp.”

22eng.: “I'd appreciate it if you could call the airline and reserve seats for me.”

The word “the” and “airline” are incorrectly parsed to depend on “seats;” the dependency type between this “the” and this “seats” is “det,” and that between “airline” and “seats” is “nn.” The correct parse is that this “the” depends on “airline” with the dependency type “det,” and this “airline” depends on the “call” with the dependency type “dobj.”

The word “reserve” is incorrectly analyzed to depend on “airline” with an incorrect type “nn_and.” The correct dependency relation is that it depends on “appreciate” with the type “advcl_and.”

The word “searts” is incorrectly analyzed to depend on “call” with the correct type “dobj.” The correct dependency relation is that it depends on “reserve.”

23eng. “Please explain to me why you failed to come yesterday.”

The word “Please” is parsed to depend on “explain” with an incorrect type “dep.” The correct type is “advmod.”

25eng.: “Experts say too much exercise does us more harm than good.”

The word “us” is incorrectly parsed to depend on “harm” with an incorrect dependency type “nsubj.” The correct dependency relation is that it depends on the verb “does” and the correct dependency type is “iobj.”

The word “harm” is correctly parsed to depend on “does,” yet with an incorrect dependency type “xcomp;” the correct dependency type is “dobj.”

26eng.: “A short walk will give my father a good appetite.”

The word “father” is correctly parsed to depend on “give,” yet with an incorrect dependency type “dep.” Its correct dependency type is “iobj.”

28eng.: “A glance at the map will tell you where you are.”

The word “you” is correctly parsed to depend on “tell,” yet with an incorrect dependency type “dobj.” Its correct dependency type is “iobj.”

31eng.: “You should make sure that he is at home before you call on him.”

The word “at” is incorrectly parsed to depend on “call,” with an incorrect dependency type “advmod.” The word “home” is parsed to depend on “at,” with an incorrect dependency type “dep.” These two words’ dependency should be collapsed, because the “at” is a preposition;

therefore, the correct parse is that the “home” depends on “be” with the dependency type “prep_at.”

The word “before” is correctly parsed to depend on “call,” yet with an incorrect dependency type “dep.” Its dependency type is “mark.”

The verb “call” is incorrectly parsed to depend on “be” with an incorrect dependency type “ccomp.” Its correct parse is that it depends on “make” with the dependency type “advcl.”

32eng “The doctor advised me neither to drink nor to smoke.”

The word “neither” is correctly parsed to depend on the verb “advise,” but with an incorrect type “preccomp.” This type is the result of converting asymmetric coordination into symmetric coordination; the original Stanford-Parser dependency type “preconj” is converted incorrectly, with the part of the word “conj” being replaced by “ccomp.” The correct dependency type for the word “neither” and the verb “advise” in this sentence is “ccomp.”

The word “drink” and “smoke” are both incorrectly parsed to depend on the verb “advise.” The correct parse is that they depend on “neither,” with the dependency type “conj.”

The word “nor” is collapsed in the output option “collapsed tree;” hence, it is absent in the original output. However, in the symmetric analysis of sentences that contain preconjuncts, it must be present in the typed-dependency tree. The correct parse is that the word “nor” depends on the word “neither,” with the dependency type “cc.”

36eng.: “I’d like these pieces of furniture sent to my house by the weekend.”

The word “by” is incorrectly analyzed to indicate the agent of the passive “sent,” therefore it is parsed to depend on “sent” with an incorrect dependency type “agent.” The correct dependency type is “prep_by.”

38eng.: “If you’ll be a good boy, we’ll let you watch television tonight.”

The word “television” is incorrectly parsed to depend on “tonight” with an incorrect type “nn,” and the word “tonight” is incorrectly parsed to depend on “watch” with an incorrect type “dobj.” The correct parse is that the “television” depends on “watch” with the type “dobj,” and “tonight” on “watch” with the type “tmod.”

40eng.: “No, I had my brother help it.”

The word “No” depends on “had” with an incorrect type “dep.” The correct type is “advmod.”

41eng.: “I’ve never had such a terrible thing happen to me before.”

The word “thing” incorrectly parsed to depend on “had” with an incorrect type “dobj,” and “happen” on “had” with “dep.” The correct parse is that “thing” depends on “happen” with “nsubj” and “happen” on “had” with “ccomp.”

44eng.: “I shouted loudly, but I couldn’t make myself heard across the large room.”

The word “heard” is parsed to depend on “make,” but with an incorrect type “dep;” the correct type is “ccomp.”

46eng.: “I was watching TV, so I didn’t notice you come in.”

The word “so” is incorrectly parsed to depend on “watching” with an incorrect type “dep;” the correct parse is that it depends on “notice” with “mark.”

47eng.: “The teacher caught a student cheating on the test this afternoon.”

The word “cheating” is parsed to depend on “caught” with an incorrect type “dobj,” and both the “a” and the “student” are parsed to depend on the “cheating.” The dependency type between the “student” and “cheating” is also incorrect; it is parsed to be “nn.” In addition, the preposition “on” is incorrectly parsed to depend on “caught.” The correct parse is that the “cheating” depends on the “caught” with the type “ccomp,” the “a” depends on the “student,” the “student” depends on the “cheating” with “nsubj,” and the “on” depends on the “cheating.”

48eng.: “Yesterday for the first time I saw that boxer knocked down.”

The word “Yesterday” is correctly parsed to depend on “saw,” yet with an incorrect dependency type “nsubj.” The correct parse is that it depends on “say” with the type “tmod.”

The word “for” is incorrectly parsed to depend on “Yesterday,” and also incorrectly parsed to have “I” as its dependent. The correct parse is that the dependency between this “for” and the “time” is collapsed and the “time” depends on “saw” with the type “prep_for.”

The word “the” is incorrectly parsed to depend on “I.” The correct parse is that it depends on “time” with the type “det.”

The word “first” is also incorrectly parsed to depend on “I.” The correct parse is that it depends on “time” with the type “amod.”

The word “I” is also incorrectly parsed to depend on “Yesterday” with an incorrect type “prep_for.” The correct parse is that it depends on “saw” with the type “nsubj.”

The word “that” is also incorrectly parsed to depend on “knocked” with an incorrect type “complm.” The correct parse is that it depends on the “boxer” with the type “det.”

54eng.: “I had my passport stolen while travelling in Italy several years ago.”

The word “Italy” is incorrectly parsed to depend on “years” with an incorrect type “nn.” The correct parse is that there is a dependency between the “in” and this “Italy,” which is collapsed to form a dependency between the “travelling” and “Italy” with the type “prep_in.”

The word “year” is parsed to depend on the “travelling,” yet with an incorrect type “prep_in.” The correct type for this dependency is “tmod.”

58eng. “My car is being repaired, so I borrowed my brother’s.”

The word “so” is incorrectly parsed to depend on “repaired” with an incorrect type “dep;” the correct parse is that it depends on “borrowed” with the type “mark.”

The word “borrowed” is parsed to depend on “repaired,” yet with an incorrect type “ccomp;” the correct type is “advcl.”

The possessive “’s” is incorrectly analyzed to be a copula that takes “brother” as its nominal subject. The correct parse is that the “brother” depends on the “borrowed” with the type “dobj” and “’s” depends on the “brother” with the type “possessive.” The intuition behind this analysis is that a noun with a possessive can function pronominally, viz. the noun phrase “my brother’s” can refer to something which the person possesses, and therefore can be a dependent of an argument-taking element (“borrow” in the sentence above). In this study, no element is supposed to be elided in such constructions for the sake of simplicity.

59eng.: “She is always saying unpleasant things about other people behind their backs.”

The word “about” is incorrectly parsed to depend on the “saying;” the correct parse is that it depends on the word “things.”

The word “behind” is incorrectly parsed to depend on the word “people;” the correct parse is that it depends on the word “saying.”

61eng.: “This time next year I’ll be working for a trading company in Tokyo.”

The word “this” is parsed to depend on the word “time,” yet with an incorrect type “nsubj;” the correct type is “det.”

The word “time” is incorrectly parsed to be the root of the sentence; the correct parse is that it depends on “working” with the type “tmod.”

The word “working” is incorrectly parsed to depend on the word “time” with an incorrect type “ccomp;” the correct parse is that it is the root of the sentence.

63eng.: “I’ll go and see who it is.”

The word “is” is incorrectly parsed to depend on the word “go;” the correct parse is that it depends on the word “see.”

64eng.: “I’ll be visiting a friend of mine in Kyoto this weekend.”

The word “in” is incorrectly parsed to depend on the word “mine;” the correct parse is that it depends on the word “friend.”

The word “weekend” is incorrectly parsed to depend on “mine;” the correct parse is that it depends on the word “visiting.”

68eng.: “No, it’s gone.”

The word “No” is correctly parsed to depend on the word “gone,” yet with an incorrect type “dep;” the correct type is “advmod.”

69eng.: “I’m all right now.”

The word “all” is correctly parsed to depend on the word “right,” yet with an incorrect type “dep;” the correct type is “advmod.”

70eng.: “Have you ever been to Hawaii?”

The word “ever” is incorrectly parsed to depend on “you;” the correct parse is that it depends on the word “been.”

71eng.: “No, I never have”

The word “No” is correctly parsed to depend on the word “have,” yet with an incorrect type “dep;” the correct type is “advmod.”

74eng.: “So you know my uncle.”

The word “So” is correctly parsed to depend on the word “know,” yet with an incorrect type “dep;” the correct type is “advmod.”

75eng.: “How long have you known him?”

The word “long” is correctly parsed to depend on the word “known,” yet with an incorrect type “dep;” the correct type is “advmod.”

76eng. “Mr. Suzuki has been teaching here ever since he came to Tokyo 25 years ago.”

The word “since” is parsed to depend on “came” with an incorrect type “dep.” The correct type is “mark.”

77eng.: “I recognized him right away because I had seen him on TV quite a few times.”

The word “seen” is incorrectly parsed to depend on the word “away,” with an incorrect type “dep;” the correct parse is that it depends on “recognized” with the type “advcl.”

78eng.: “We'd been married for two years when we had our first child.”

The word “had” is parsed to depend on the word “when” with an incorrect type “dep;” the correct type is “rcmod.”

79eng.: “I hadn't seen him for about a year when he died.”

The word “died” is parsed to depend on the word “year” with an incorrect type “dep;” the correct type is “rcmod.”

81eng.: “My son had hardly lain down on the bed when he fell asleep.”

The word “fell” is incorrectly parsed to depend on the word “bed” with an incorrect type “dep;” the correct parse is that it depends on the word “lain” with the type “advcl.”

84eng.: “The new hall will have been built by the end of this month.”

The “by” is incorrectly analyzed to indicate the agent of the passive “built,” therefore it is parsed to depend on “built” with an incorrect dependency type “agent.” The correct dependency type is “prep_by.”

90eng.: “Weren't you taught at school that whales are mammals?”

The word “Were,” “n’t” and “you” are incorrectly parsed to depend on the word “school.” The correct parse is that they depend on “taught.”

The word “taught” is incorrectly parsed to depend on the word “you” with an incorrect type “partmod;” the correct parse is that it is the root of the sentence.

The word “school” is incorrectly parsed to be the root of the sentence. The correct parse is that it depends on the word “taught” with the type “prep_at.”

The word “mammals” is incorrectly parsed to depend on “school” with an incorrect type “dep;” the correct parse is that it depends on “taught” with the type “ccomp.”

Subjunctive mood

95eng.: “If you hadn’t had to go to school yesterday, what would you have done?”

The word “have” is parsed to depend on the word “do,” yet with an incorrect type “dep;” the correct type is “advcl.”

The word “what” is incorrectly parsed to depend on the word “would” with an incorrect type “nsubj;” the correct parse is that it depends on the word “do” with the type “dobj.”

The word “would” is incorrectly parsed to depend on the word “yesterday” with an incorrect type “rcmod;” the correct parse is that it depends on “do” with the type “aux.”

97eng.: “If I were Hanako, I wouldn't have married such a terrible alcoholic.”

The word “such” is incorrectly parsed to depend on the word “married” with an incorrect type “prep;” the correct parsed is that it depends on “alcoholic” with the type “predet.”

The word “alcoholic” is incorrectly parsed to depend on the word “such” with an incorrect type “dep;” the correct parse is that it depends on the word “marry” with the type “dobj.”

99eng.: “If you should win 300,000,000 yen in the lottery, let me have half of it.”

The word “win” is correctly parsed to depend on the word “let,” yet with an incorrect type “dep;” the correct type is “advcl.”

100eng.: “If Taro's parents met Hanako, I'm sure they would like her.”

The word “like” is correctly parsed to depend on the word “sure,” yet with an incorrect type “dep;” the correct type is “ccomp.”

101eng.: “Everyone in the Suzuki family treated me kindly, as if I were a family member.”

The word “if” is correctly parsed to depend on the word “member,” yet with an incorrect type “dep;” the correct type is “mark.”

The word “member” is correctly parsed to depend on the word “treated,” yet with an incorrect type “dep;” the correct type is “advcl.”

102eng.: “Does your son like his new school?”

The word “Does” is incorrectly parsed to depend on the word “son” with an incorrect type “dep;” the correct parse is that it depends on the word “like” with the type “aux.”

The word “your” is correctly parsed to depend on the word “son,” yet with an incorrect parse “nsubj;” the correct type is “poss.”

The word “son” is incorrectly parsed to be the root of the sentence; the correct parse is that it is the subject of the word “like.”

The word “like” is incorrectly parsed to a preposition; the correct parse is that it is the root of the sentence.

The word “school” is correctly parsed to be the object of the word “like;” however, this “like” is incorrectly parsed to be a preposition that is incorrectly parsed to depend on “son.” Therefore, the dependency of this preposition is collapsed, and the result is that the word “school” is incorrectly parsed to be dependent on the word “son” with an incorrect type “prep_like.” The correct parse is that it depends on the word “like” with the type “dobj.”

103eng. “Oh, I wish he did.”

The word “Oh” is parsed to depend on “wish” with an incorrect type “dep.” The correct type is “advmod.”

104eng. “If only my father had seen a doctor a little sooner!”

The word “If” is incorrectly parsed to be the root of the sentence. The correct parse is that it depends on “seen” with the type “mark.”

The word “seen” is incorrectly parsed to depend on “if” with an incorrect type “dep.” The correct parse is that it is the root of the sentence.

106eng. “If only the rain would stop!”

The word “If” is incorrectly parsed to be the root of this sentence. The correct parse is that it depends on the word “stop” with the type “mark.”

The word “stop” is incorrectly parsed to depend on “If” with an incorrect type “dep.” The correct parse is that it is the root of the sentence.

Auxiliaries

118eng. “Yesterday I left the office before eleven, so I was able to catch the last train.”

The word “so” is incorrectly parsed to depend on “left” with an incorrect type “dep.” The correct parse is that it depends on “able” with the type “mark.”

121eng. “You must come and see us.”

The word “us” is incorrectly parsed to depend on “come.” The correct parse is that it depends on “see” with the type “dobj.”

125eng. “Did you study English last night?”

The word “English” is incorrectly parsed to depend on “night” with an incorrect type “amod.” The correct parse is that it depends on “study” with the type “dobj.”

126eng. “No, but I should have.”

The word “no” is correctly parsed to depend on “have,” but with an incorrect type “dep;” the correct type is “advmod.”

The word “I” is incorrectly parsed to depend on ROOT with an incorrect type “root_but;” the correct parse is that it depends on “have” with the type “nsubj.”

131eng. “I may have met him somewhere before, but I can’t recall where.”

The word “where” is correctly parsed to depend on “recall,” but with an incorrect type “dep;” the correct type is “dobj.”

Infinitives, gerunds, and participles

135eng. “Please remember to feed the goldfish every three days when I am away.”

The word “Please” is correctly parsed to depend on “remember,” but with an incorrect type “dep;” the correct type is “advmod.”

136eng. “Every time I travel abroad, I regret not studying English harder when I was young.”

The word “English” is incorrectly parsed to depend on “young” with an incorrect type “nsubj;” the correct parse is that it depends on “studying” with the type “dobj.”

The word “harder” is incorrectly parsed to depend on “young” with an incorrect type “dep;” the correct parse is that it depends on “studying” with the type “advmod.”

The word “young” is correctly parsed to depend on “studying,” but with an incorrect type “xcomp;” the correct type is “advcl.”

140eng. “I’ll do everything else.”

The word “everything” is incorrectly parsed to depend on “else” with an incorrect type “nsubj.” The correct parse is that it depends on “do” with the type “dobj.”

The word “else” is incorrectly parsed to depend on “do” with an incorrect type “xcomp;” the correct type is that it depends on “everything” with the type “amod.”

142eng. “I’ve got something important to tell you.”

The word “something” is incorrectly parsed to depend on “important” with an incorrect type “nsubj;” the correct parse is that it depends on “got” with the type “dobj.”

The word “important” is incorrectly parsed to depend on “got” with an incorrect type “xcomp;” the correct parse is that it depends on “something” with the type “amod.”

The word “tell” is incorrectly parsed to depend on “important” with an incorrect type “xcomp;” the correct parse is that it depends on “something” with the type “infmod.”

145eng. “I hurried to the airport so as not to miss the flight, but it was too late.”

The word “not” is correctly parsed to depend on “miss,” but with an incorrect type “dep;” the correct type is “neg.”

148eng. “Be careful not to catch a cold because you have a job interview next week.”

The word “catch” is parsed to depend on “careful” with an incorrect type “dep.” The correct type is “xcomp.”

150eng. “The class goes too fast for me to keep up with.”

The word “for” is incorrectly parsed to depend on “keep” with an incorrect type “mark,” and the word “me” is incorrectly parsed to depend on “keep” with an incorrect type “nsubj;” the correct parse is that these dependencies are collapsed and the word “me” depends on “fast” with the type “prep_for.”

The word “keep” is correctly parsed to depend on “fast,” but with an incorrect type “dep;” the correct type is “infmod.”

160eng. “Seen from a distance, the huge rock looked like a human face.”

The word “Seen” is correctly parsed to depend on “look,” but with an incorrect type “nsubj;” the correct type is “partmod.”

The word “rock” is incorrectly parsed to depend on “distance” with an incorrect type “appos;” the correct parse is that it depends on “look” with the type “nsubj.”

161eng. “Last night I stayed up until three o'clock, watching the Open on TV.”

The word “watching” is correctly parsed to depend on “stayed,” but with an incorrect type “xcomp;” the correct type is “partmod.”

164eng. “I watched the movie standing because every seat was taken.”

The word “the” is incorrectly parsed to depend on “standing.” The correct parse is that it depends on “movie.”

The word “movie” is incorrectly parsed to depend on “standing” with an incorrect type “nn.” The correct parse is that it depends on “watched” with the type “dobj.”

The word “standing” is correctly parsed to depend on “watched,” but with an incorrect type “dobj;” the correct type is “partmod.”

166eng. “Judging from the way he speaks, I'm sure he is not a native of Osaka.”

The word “Judging” is correctly parsed to depend on “sure,” but with an incorrect type “dep;” the correct type is “partmod.”

The word “speaks” is incorrectly parsed to depend on “sure” with an incorrect type “parataxis;” the correct parse is that it depends on “way” with the type “rcmod.”

The word “native” is correctly parsed to depend on “sure,” but with an incorrect type “dep;” the correct type is “ccomp.”

Adverbial clauses

174eng. “I can’t bring myself to go out when tired.”

The word “tired” is incorrectly parsed to depend on “go” with an incorrect type “dep.” The correct parse is that it depends on “bring” with the type “advcl.”

181eng. “He was late again this morning.”

The word “late” is incorrectly parsed to depend on “again;” the correct parse is that it depends on “was.”

182eng. “He arrived an hour after school began”

The word “hour” is incorrectly parsed to depend on “begin” with an incorrect type “dep.” The correct parse is that it depends on “after” with the type “tmod.”

The word “after” is parsed to depend on “began” with an incorrect type “dep.” The correct type is “mark.”

The word “began” is parsed to depend on “arrived” with an incorrect type “ccomp.” The correct type is “advcl.”

186eng. “Now that our children are grown up, we can do anything we like.”

The word “that” is parsed to depend on “grown” with an incorrect type “dep.” The correct type is “mark.”

The word “grown” is parsed to depend on “do” with an incorrect type “dep.” The correct type is “advcl.”

192eng. “Please put the magazine back where you found it.”

The word “Please” is correctly parsed to depend on “put,” but with an incorrect type “dep;” the correct type is “advmod.”

196eng. “Don’t look down upon a person just because he or she is poor.”

The word “because” is correctly parsed to depend on “poor,” but with an incorrect type “dep;” the correct type is “mark.”

197eng. “I set the alarm for five in the morning so that I could study earlier.”

The word “morning” is incorrectly parsed to depend on “set” with the correct type “prep_in;” the correct parse is that it depends on “five.”

The word “that” is correctly parsed to depend on “study,” but with an incorrect type “dep;” the correct type is “mark.”

198eng. “Wear your raincoat so that you will not get wet.”

The word “that” is correctly parsed to depend on “wet” with an incorrect type “dep.” The correct type is “mark.”

The word “get” is correctly parsed to depend on “wet” as a copula, but with an incorrect type “dep.” The correct type is “cop.”

199eng. “Take these sandwiches with you in case you get hungry on the way.”

The word “Take” is incorrectly parsed to depend on “hungry” with an incorrect type “dep.” The correct parse is that it is the root of the sentence.

The word “case” is incorrectly parsed to depend on “you” with the type “prep_in.” The correct parse is that it depends on “Take.”

The word “hungry” is incorrectly parsed to depend on “Take” with an incorrect type “dep.” The correct parse is that it depends on “case” with the type “ccomp.”

The word “get” is correctly parsed to depend on “hungry,” but with an incorrect type “dep.” The correct type is “cop.”

200eng. “I lowered my voice for fear I might be overheard.”

The word “lowered” is incorrectly parsed to depend on “overheard” with an incorrect type “dep.” The correct parse is that it is the root of the sentence.

The word “overheard” is incorrectly parsed to be the root of the sentence. The correct parse is that it depends on “fear” with the type “ccomp.”

202eng. “Even though I love the Beatles, I’m tired of listening to *Yesterday*.”

The word “though” is correctly parsed to depend on “love,” but with an incorrect type “dep.” The correct type is “mark.”

The word “love” is correctly parsed to depend on “tired,” but with an incorrect type “dep.” The correct type is “advcl.”

203eng. “Whether you like it or not, you should be here by eight tomorrow morning.”

The word “Whether” is correctly parsed to depend on “like” with an incorrect type “dep.” The correct type is “mark.”

The word “like” is correctly parsed to depend on “be,” but with an incorrect type “dep.” The correct type is “advcl.”

The word “not” is incorrectly parsed to depend on “be” with an incorrect type “dep_or.” The correct parse is that it depends on “like” with the type “neg_or.”

The word “eight” is incorrectly parsed to depend on “tomorrow” with an incorrect type “nn.” The correct parse is that it depends on “be” with the type “prep_by.”

The word “tomorrow” is incorrectly parsed to depend on “be” with an incorrect type “prep_by.” The correct parse is that it depends on “morning” with the type “advmod.”

204eng. “No matter what you wear, you look pretty.”

The word “No” is correctly parsed to depend on “matter,” but with an incorrect type “dep.” The correct type is “neg.”

The word “matter” is incorrectly parsed to depend on “look” with an incorrect type “advmod.” The correct parse is that it depends on “wear” with the type “mark.”

The word “wear” is incorrectly parsed to depend on “matter” with an incorrect type “dep.” The correct parse is that it depends on “look” with the type “advcl.”

205eng. “Whichever candidate we choose, we should not expect too much of him.”

The word “Whichever” is correctly parsed to depend on “candidate,” but with an incorrect type “nsubj.” The correct type is “amod.”

The word “candidate” is incorrectly parsed to depend on “expect” with an incorrect type “ccomp.” The correct parse is that it depends on “choose” with the type “dobj.”

The word “choose” is incorrectly parsed to depend on “candidate” with an incorrect type “ccomp.” The correct parse is that it depends on “expect” with the type “advcl.”

206eng. “No matter how fast I drive, I can’t get to the office in thirty minutes.”

The word “No” is correctly parsed to depend on “matter,” but with an incorrect type “dep.” The correct type is “neg.”

The word “matter” is incorrectly parsed to depend on “get” with an incorrect type “advmod.” The correct parse is that it depends on “drive” with the type “mark.”

The word “fast” is correctly parsed to depend on “drive,” but with an incorrect type “dep.” The correct type is “advmod.”

The word “drive” is incorrectly parsed to depend on “matter” with an incorrect type “dep.” The correct parse is that it depends on “get” with the type “advcl.”

207eng. “I know nothing about him except that he used to be a professional singer.”

The word “that” is correctly parsed to depend on “used,” but with an incorrect type “dep.” The correct type is “mark.”

208eng. “There's a rumor that the movie star is going to get married soon.”

The word “going” is incorrectly parsed to depend on “is” with the type “cccomp.” The correct parse is that it depends on “rumor” with the type “ccomp.”

209eng. “There was little hope that the rescue party would come back safe and sound.”

The word “come” is incorrectly parsed to depend on “was” with the type “cccomp.” The correct parse is that it depends on “hope” with the type “ccomp.”

210eng. “Take your temperature to see if you have a fever.”

The word “see” is incorrectly parsed to depend on “temperature.” The correct parse is that it depends on “Take.”

Relative clauses

214eng. “Do customers who smoke in restaurants bother you?”

The word “Do” is incorrectly parsed to be the root of the sentence. The correct parse is that it depends on “bother” with the type “aux.”

The word “customers” is incorrectly parsed to depend on “Do” with an incorrect type “dobj.” The correct parse is that it depends on “bother” with the type “nsubj.”

The word “in” is incorrectly parsed to depend on “bother” with the type “mark.” The correct parse is that it is collapsed to be the dependency type “prep_in” for the dependency between “smoke” and “restaurant.”

The word “restaurant” is incorrectly parsed to depend on “bother” with an incorrect type “nsubj.” The correct parse is that it depends on “smoke” with the type “prep_in.”

The word “bother” is incorrectly parsed to depend on “smoke” with an incorrect type “advcl.” The correct parse is that it is the root of the sentence.

216eng. “The man who I first thought was the criminal turned out to be a detective.”

The word “man” is incorrectly parsed to depend on “was” with the type “nsubj.” The correct parse is that it depends on “turned” with the type “nsubj.”

The word “criminal” is incorrectly parsed to be the root of the sentence. The correct parse is that it depends on “thought” with the type “ccomp.”

The word “turned” is incorrectly parsed to depend on “criminal” with an incorrect type “partmod.” The correct parse is that it is the root of the sentence.

218eng. “My friend’s sister is a famous singer whose songs many people sing.”

The word “whose” is incorrectly parsed to depend on “people” with the type “poss.” The correct parse is that it depends on “songs” with the type “poss.”

The word “songs” is incorrectly parsed to depend on “people” with an incorrect type “amod.” The correct parse is that it depends on “sing” with the type “dobj.”

219eng. “I may have to work overtime, in which case I’ll call you.”

The word “overtime” is correctly parsed to depend on “work,” but with an incorrect type “dobj.” The correct type is “advmod.”

The word “case” is correctly parsed to depend on “call,” but with an incorrect type “rel.” The correct type is “prep_in.”

224eng. “There seemed to be something he wanted done.”

The word “done” is correctly parsed to depend on “wanted” with an incorrect type “dep.” The correct type is “xcomp.”

231eng. “He often leaves undone what he ought to do.”

The word “undone” is correctly parsed to depend on “leaves,” but with an incorrect type “dep.” The correct type is “xcomp.”

The word “what” is correctly parsed to depend on “ought,” but with an incorrect type “dep.” The correct type is “dobj.”

The word “ought” is incorrectly parsed to depend on “undone” with the type “ccomp.” The correct parse is that it depends on “leaves.”

233eng. “The father gave his son what little money he had saved.”

The word “son” is correctly parsed to depend on “gave,” but with an incorrect type “dobj.” The correct type is “iobj.”

The word “money” is incorrectly parsed to depend on “save” with the type “dobj.” The correct parse is that it depends on “gave” with the type “dobj.”

The word “saved” is incorrectly parsed to depend on “gave” with an incorrect type “dep.” The correct parse is that it depends on “money” with the type “rcmod.”

236eng. “This is how we get rid of things we no longer need.”

The word “get” is correctly parsed to depend on “rid,” but with an incorrect type “dep.” The correct type is “cop.”

240eng. “Choose whichever one of the three methods you think is best.”

The word “whichever” is incorrectly parsed to depend on “best” with an incorrect type “dobj.” The correct parse is that it depends on “one” with the type “amod.”

The word “one” is incorrectly parsed to depend on “best” with an incorrect type “nsubj.” The correct parse is that it depends on “choose” with the type “dobj.”

The word “best” is incorrectly parsed to depend on “choose” with the type “ccomp.” The correct parse is that it depends on “think” with the type “ccomp.”

Comparison:

241eng. “You should read as many different books as you can.”

The word “as” is incorrectly parsed to be a preposition and is collapsed to be the dependency type “prep_as” between “read” and “newspaper.” The correct parse is that it depends on “many” with the type “advmod.”

The word “newspaper” is parsed to depend on “read” with an incorrect type “prep_as.” The correct type is “dobj.”

243eng. “The pain was more than I could bear, so I took some medicine.”

The word “bear” is parsed to depend on “more” with an incorrect type “dep.” The correct type is “advcl.”

The word “so” is incorrectly parsed to depend on “more” with an incorrect type “dep.” The correct parse is that it depends on “took” with the type “mark.”

The word “took” is parsed to depend on “more” with an incorrect type “ccomp.” The correct type is “advcl.”

246eng. “In order to keep in good health, you should smoke fewer cigarettes and drink less.”

The word “In” is incorrectly parsed to depend on “keep” with an incorrect type “mark.” The correct parse is that it is collapsed to be the dependency type “prep_in” between “smoke” and “keep.”

The word “keep” is incorrectly parsed to depend on “smoke” with an incorrect type “dep.” The correct parse is that it depend on “order” with the type “infmod.”

248eng. “My brother plays the guitar much better than he used to.”

The word “guitar” is incorrectly parsed to depend on “better” with an incorrect type “nsubj.” The correct parse is that it depends on “plays” with the type “dobj.”

The word “better” is parsed to depend on “plays” with an incorrect type “xcomp.” The correct type is “advmod.”

The word “used” is incorrectly parsed to depend on “plays” with the type “advcl.” The correct parse is that it depends on “better.”

256eng. “He was the last person I expected to run into in London.”

The word “into” is parsed to depend on “run” with an incorrect type “prep.” The correct type is “prt.”

The word “in” is incorrectly parsed to depend on “into” with an incorrect type “pcomp.”

The correct parse is that it is collapsed to be the dependency type “prep_in” between “run” and “London.”

259eng. “You should know better than to take an examination without preparing for it.”

The word “better” is incorrectly parsed to depend on “than” with an incorrect type “dep.” The correct parse is that it depends on “know” with the type “advmod.”

The word “take” is parsed to depend on “than” with an incorrect type “dep.” The correct type is “infmod.”

260eng. “I didn’t even speak to him, much less talk it over with him.”

The word “less” is incorrectly parsed to depend on “speak” with the type “advmod.” The correct parse is that it depends on “talk” with the type “advmod.”

The word “talk” is parsed to depend on “speak” with an incorrect type “dep.” The correct type is “advcl.”

261eng. “The older we get, the less innocent we become.”

The word “older” is correctly parsed to depend on “get” with an incorrect type “dep.” The correct type is “advmod.”

The word “get” is correctly passed to depend on “become,” with an incorrect type “dep.” The correct type is “advcl.”

The word “innocent” is correctly parsed to depend on “become” with an incorrect type “dep.” The correct type is “advmod.”

264eng. “I can no more play the violin than a baby can.”

The word “no” is parsed to depend on “more” with an incorrect type “dep.” The correct type is “neg.”

266eng. “I'm afraid it'll be difficult for you to find a new job in Tokyo.”

The word “difficult” is correctly parsed to depend on “afraid” with an incorrect type “dep.” The correct type is “ccomp.”

268eng. “I found it very difficult to adjust myself to life in the new school.”

The word “it” is incorrectly parsed to depend on “adjust” with an incorrect type “nsubj.” The correct parse is that it depends on “found” with the type “dobj.”

The word “difficult” is incorrectly parsed to depend on “adjust” with an incorrect type “dep.” The correct parse is that it depends on “found” with the type “acomp.”

The word “adjust” is incorrectly parsed to depend on “found” with the type “xcomp.” The correct parse is that it depends on “difficult.”

270eng. “It costs a huge sum of money to travel around the world by ship.”

The word “sum” is incorrectly parsed to depend on “travel” with the type “nsubj.” The correct parse is that it depends on “cost” with the type “dobj.”

271eng. “I tell you that it’s no good your being angry with me.”

The word “good” is incorrectly parsed to depend on “your” with an incorrect type “advmod.” The correct parse is that it depends on “tell” with the type “ccomp.”

The word “your” is incorrectly parsed to depend on “tell” with an incorrect type “ccomp.” The correct parse is that it depends on “angry” with the type “poss.”

The word “angry” is incorrectly parsed to depend on “your” with an incorrect type “xcomp.” The correct parse is that it depends on “good” with the type “ccomp.”

274eng. “It’s what you do that matters, not the way you do it.”

The word “what” is parsed to depend on the first “do” with an incorrect type “dep.” The correct type is “dobj.”

The word “that” is incorrectly parsed to depend on the second “do” with the type “compln.” The correct parse is that it depends on “matters.”

The word “matters” is incorrectly parsed to depend on the second “do” with an incorrect type “nsubj.” The correct parse is that it depends on the first “do” with the type “rcmod.”

The word “not” is parsed to depend on “way” with an incorrect type “dep.” The correct type is “neg.”

The word “way” is incorrectly parsed to depend on “do” with an incorrect type “dep.” The correct parse is that it depends on “is” with the type “ccomp.”

The second “do” is incorrectly parsed to depend on the first “do” with an incorrect type “ccomp.” The correct parse is that it depends on “way” with the type “rcmod.”

279eng. “How long have you had that car of yours?”

The word “long” is parsed to depend on “had” with an incorrect type “dep.” The correct type is “advmod.”

284eng. “Okay, where shall we go?”

The word “Okay” is parsed to be the root of the sentence. The correct parse is that it depends on “go” with “advcl.”

The word “go” is incorrectly parsed to depend on “Okay” with an incorrect type “rcmod.” The correct parse is that it is the root of the sentence.

285eng. “The bus hasn’t come yet, has it?”

The word “has” is parsed to depend on “come” with an incorrect type “dep.” The correct type is “advcl.”

The word “it” is parsed to depend on “has,” with an incorrect type “dobj.” The correct type is “nsubj.”

286eng. “Oh, no, it hasn’t.”

The word “Oh” is parsed to depend on “not” with an incorrect type “dep.” The correct type is “advcl.”

The word “no” is incorrectly parsed to depend on “Oh” with an incorrect type “dep.” The correct parse is that it depends on “not” with the type “advcl.”

287eng. “Would you mind posting a letter for me on your way home?”

The word “way” is incorrectly parsed to depend on “home” with an incorrect type “nn.” The correct parse is that it depends on “post” with the type “prep_on.”

The word “your” is incorrectly parsed to depend on “home” with the type “poss.” The correct parse is that it depends on “way.”

The word “home” is incorrectly parsed to depend on “post” with an incorrect type “prep_on.” The correct parse is that it depends on “way” with the type “advmod.”

288eng. “Not at all.

The word “at” is not collapsed to be the dependency-type “prep_at” between “Not” and “all.”

290eng. “No, go ahead.”

The word “No” is parsed to depend on “go” with an incorrect type “nsubj.” The correct type is “advcl.”

298eng. “Well, I don’t think I am being arrogant.”

The word “Well” is parsed to depend on “think” with an incorrect type “dep.” The correct type is “advcl.”

300eng. “I don’t know who or what he is, or where he lives.”

The word “who” is incorrectly parsed to depend on “know” with an incorrect type “dep.” The correct parse is that it depends on “is” with the type “dobj.”

The word “is” is parsed to depend on “know” with an incorrect type “dep_or.” The correct type is “ccomp.”

The word “lives” is incorrectly parsed to depend on “who” with an incorrect type “conj_or_or.” The correct parse is that it depends on “know” with the type “ccomp_or.”

303eng. “I can’t walk along this street without running into someone I know.”

The word “walk” is incorrectly parsed to depend on “know” with an incorrect type “dep.” The correct parse is that it is the root of the sentence.

The word “know” is incorrectly parsed to be the root of the sentence. The correct parse is that it depends on “someone” with the type “rcmod.”

314eng. “While at college, they fell in love with each other.”

The word “fell” is incorrectly parsed to depend on “love” with an incorrect type “csubj.” The correct parse is that it is the root of the sentence.

The word “in” is parsed to depend on “fell” with an incorrect type “prt.” This preposition must be collapsed to be the dependency type “prep_in” between “fell” and “love.”

The word “love” is incorrectly parsed to be the root of the sentence. The correct parse is that it depends on “fell” with the type “prep_in.”

315eng. “Many people have cars, but only a few of them use them to go to work.”

The second “them” is incorrectly parsed to depend on “go” with an incorrect type “nsubj.” The correct parse is that it depends on “use” with the type “dobj.”

317eng. “Yes, quite a few.”

The word “Yes” is incorrectly parsed to depend on “quite” with an incorrect type “advmod.” The correct parse is that it depends on “few” with the type “advcl.”

The word “quite” is incorrectly parsed to be the root of the sentence. The correct parse is that it depends on “few” with the type “predet.”

The word “few” is incorrectly parsed to depend on “quite” with the type “dobj.” The correct parse is that it is the root of the sentence.

320eng. “Almost all the passengers on the bus are tourists.”

The word “bus” is incorrectly parsed to depend on “tourists” with the type “prep_on.” The correct parse is that it depends on “passengers” with the type “prep_on.”

323eng. “That comedy isn’t amusing; in fact, it is anything but funny.”

The word “funny” is incorrectly parsed to depend on “amusing” with an incorrect type “parataxis_but.” The correct parse is that it depends on “anything” with the type “prep_but.”

325eng. “One and a half years is a long time to have to wait for a visa.”

The word “One” is incorrectly parsed to depend on “half” with the type “number.” The correct parse is that it depends on “years.”

The word “a” is incorrectly parsed to depend on “year” with an incorrect type “num_and.” The correct parse is that it depends on “half” with the type “det.”

The word “half” is parsed to depend on “year” with an incorrect type “num.” The correct type is “num_and.”

333eng. “My brother and I went skating on the lake yesterday afternoon.”

The word “yesterday” is incorrectly parsed to depend on “skating” with the type “tmod.” The correct parse is that it depends on “went.”

339eng. “Except for a slight fever, Taro doesn’t seem to be very ill.”

The word “for” is incorrectly parsed not to be collapsed to a dependency type. The word “fever” is parsed to depend on “seem” with an incorrect type “pobj.” The correct parse is that it depends on “seem” with the type “prep_except_for.”

Appendix III: The list of incorrect parses for Japanese sentences in Basic 300 (Iida 2010) by KNP

5jpn.

“Kimi-wa	jibun-ga	yat-ta	koto-wo	sugu	tomodachi-ni
You-topic	*yourself-postp*	*do-past*	*thing-postp*	*right.away*	*friend-postp*
ayamat-ta	hou-ga	ii.”			
apologize-past	*side-postp*	*good*			

“You should apologize to your friend right away for what you did.”

The syntactic unit “jibunga (*myself*)” is incorrectly parsed to depend on “ayamatta (*someone apologized for something*)” with the type “postp_ga.” The correct parse is that it depends on “yatta (*someone did something*).”

6jpn.

“Kanja-ga	nakunat-ta-no-wa	naze-ka,	ima-mo	kaimoku
Patirent-postp	*die-past-postp-topic*	*why-int*	*now-focus*	*at.all*
wakara-nai.”				
understand-neg				

“The cause of the patient’s death remains a big mystery.”

The syntactic unit “nakunattanowa (*that something got lost* as a topic)” is incorrectly parsed to depend on “wakaranai (*nobody knows*)” with an incorrect type “advcl.” The correct parse is that it depends on “nazeka” with the type “topic.”

14jpn

“Ano	oto-wa	gaman-nara-nai.”
That	*sound-topic*	*endurance-become-neg*

“I cannot stand that noise.”

The syntactic unit “gamannaranai (*I cannot stand*)” is incorrectly segmented into “gamannara” and “nai.”

21jpn

“Tanaka-san	go-fuufu-ga	konya-no
Tanaka-end	*polite-married.couple-postp*	*tonight-postp*
tanjoukai-ni	kite-kudasaru-to	ii-na.”
birthday.party-postp	*come-polite-postp*	*good-end*

“I hope Mr. and Mrs. Tanaka come to my birthday party tonight.”

The syntactic unit “Tanaka-san (*Mr. Tanaka*)” is parsed to depend on “go-fufu-ga (*wife and husband*)” with an incorrect type “dep.” The correct type is “nn.”

31jpn

“Ano	kata-wo	houmon-suru	mae-ni,	ie-ni	iru
That	*person-postp*	*visit-do*	*before-postp*	*house-postp*	*be.present*

koto-wo	kakunin-shi-ta	hou-ga	ii-desu-yo.”
fact-postp	*confirmation-do-past*	*side-postp*	*good-polite.present-end*

“You should make sure that he is at home before you call on him.”

The syntactic unit “maeni (*before something*)” is incorrectly parsed to depend on “iru (*someone is somewhere*)” with the correct type “postp_ni.” The correct parse is that it depends on “kakuninshita (*someone checked something*).”

32jpn.

“Isha-kara,	sake-mo	tabako-mo	yara-nai	hou-ga
Doctor-postp	*sake-focus*	*tobacco-focus*	*do-neg*	*side-postp*

ii-to	iwa-re-mashi-ta.”
good-ccomp	*say-passive-polite-past*

“The doctor advised me neither to drink nor to smoke.”

The syntactic unit “ishakara (*from a doctor*)” is incorrectly parsed to depend on “iito (*that something is good*)” with the correct type “postp_kara.” The correct parse is that it depends on “iwaremashita (*someone was told*).”

The syntactic unit “sakemo (*alcohol, too*)” is incorrectly parsed to depend on “tabakomo (*tobacco, too*)” with the type “focus.” The correct parse is that it depends on “yaranai (*someone doesn’t do something*)” with the type “focus.”

33jpn.

“Kekkon-shi-te	kudasait-te	iwa-re-ta	toki,	joudan-da-to	omot-ta-wa.”
Marriage-do-infl	*give-infl*	*say-passive-past*	*time*	*joke-be-ccomp*	*think-past-end*

“When he asked me to marry him, I thought he was joking.”

The syntactic unit “kekkonshite (*someone marries someone, and*)” is incorrectly parsed to depend on “iwareta (*someone was told*)” with the correct type “advcl_te.” The correct parse is that it depends on “kudasaitte (*that you give me your favor*).”

34jpn.

“Watashitachi-no	aida-ni-wa,	ikanaru	gokai-mo	atte-wa
We-postp	*between-postp-topic*	*any*	*misunderstanding-focus*	*be-postp*
komaru.”				
troublesome				

“I don’t want there to be any misunderstanding between us.”

The syntactic unit “aidaniwa (*between*)” is incorrectly parsed to depend on “komaru (*it is* *troublesome*)." The correct parse is that it depends on "attewa (*that something is somewhere* as a topic)."

50jpn.

"Watashi-wa	seigo	wazuka	ikkagetu-de	ryoushin-ni	shina-re,
I-topic	*after.birth*	*only*	*one.month-postp*	*parents-postp*	*die-passive,*
sofubo-ni	sodate-rare-mashi-ta."				
grand.parents-postp	*raise-passive-polite-past*				

"My parents died only a month after I was born, and my grandparents brought me up."

The syntactic unit "seigo (*after the birth*)" is incorrectly parsed to depend on "wazuka (*only*)" with an incorrect type "dep." The correct parse is that it depends on "ikkagetsu" with the type "amod."

52jpn.

"Sono	bakuhatsu-jiko-de,	shisha-ga	futari,	juushousha-ga
The	*explosion-accident-postp,*	*victim-postp*	*two.person,*	*severely.wounded-postp*
san-nin	de-ta."			
three-person	*result-past*			

"Two people were killed and three seriously injured in the explosion."

The syntactic unit "shishaga (*victims*)" is incorrectly parsed to depend on "sannin (*three people*)" with the correct type "postp_ga." The correct parse is that it depends on "futari (*two people*)."

The syntactic unit "futari" is incorrectly parsed to depend on "sannin." The correct parse is that it depends on "deta (*something resulted*)."

54jpn.

"Watashi-wa,	suunen-mae	itaria-de	pasupooto-wo
I-topic	*several.years-ago*	*Italy-postp*	*passport-postp*

nusum-are-ta	koto-ga	aru."
steal-passive-past	*event-postp*	*be.present*

"I had my passport stolen while traveling in Italy several years ago."

The syntactic unit "suunenmae (*three years ago*)" is incorrectly segmented into two syntactic units "suunen" and the rest "mae" is adjoined to the next unit "itaria."

The syntactic units "nusumareta (*someone had something stolen*)," "kotoga (*event*)" and "aru (*something has happened*)" are incorrectly analyzed to form one syntactic unit.

The correct parse is that "suunenmae" depends on "nusumareta" with the type "advmod," "nusumareta" depends on "kotoga" with the type rcmod, and "kotoga" depends on "aru" with the type "postp_ga."

72jpn

"Wakai koro	chuugoku-ni	sun-da	koto-ga	arimasu-ga,
Young time	*China-postp*	*live-past*	*thing-postp*	*be.polite.present-postp,*

chuugokugo-wa	sappari-desu."
Chinese-topic	*not.at.all-polite.present*

"I lived in China abour five years when I was young, but I cannot speak Chinese at all."

The syntactic units "sunda," "kotoga" and "arimasuga" are incorrectly parsed to be one syntactic unit. The correct parse is that "sunda" depends on "kotoga" with the type "rcmod," "kotoga" on "arimasuga" with the type "postp_ga," and "arimasuga" on "sapparidesu" with the type "advcl_ga."

77jpn.

“Tadachi-ni sono hito-da-to wakari-mashi-ta, terebi-de

Immediate-postp the person-be-postp notice-polite-past, TV-postp

nando-mo mite-i-mashita-kara.”

many.times-focus watch-perfect-polite-end

“I recognized him right away because I had seen him on TV quite a few times.”

The syntactic unit “wakarimashita” is incorrectly parsed to depend on “terebide” with an incorrect type “rcmod.” The correct parse is that it is the root of the sentence.

The syntactic unit “miteimashitakara” is incorrectly parsed to be the root of the sentence. The correct parse is that it depends on “wakarimashita” with the type “advcl_kara.”

81jpn.

“Toko-ni tsuku-ka tsuka-nai-ka-no uchi-ni

Bed-postp get.in-int get.in-neg-int-postp while-postp

neitte-shimai-mashi-ta.”

fall.asleep-perfect-polite-past

“My son had hardly lain down on the bed when he fell asleep.”

The syntactic units “tsukuka” and “tsukanaikano” are incorrectly segmented as follows: “tsukukatsu” “kanai” and “kano.” The correct parse is that the unit “tsukuka” depends on “tsukanaikano” with the type “advcl_ka,” and “tsukanaikano” depends on “uchi-ni” with the type “advcl.”

85jpn.

“Rainen, chichi-wa kono kaisha-de kinzoku

Next.year, my.father-topic this company-postp keeping.on.working

sanjuunen-to	iu-koto-ni	nari-masu.”
thirty.years-postp	*say-thing-postp*	*become-polite.present*

“Next year, my father will have been working for this company for thirty years.”

The syntactic unit “kinzoku” is correctly parsed to depend on “sanjunen,” but with an incorrect type “dep.” The correct type is “nn.”

The syntactic units “iukotoni” and “narimasu” are incorrectly parsed to be one syntactic unit.

87jpn.

“Ohiru-wo	tabe-ta-ra	shibuya-e	kaimono-ni	iku-wayo,	anata.”
Lunch-postp	*eat-past-infl*	*Shibuya-postp*	*shopping-postp*	*go-end*	*darling*

“When we have had lunch, we’ll go shopping in Shibuya, darling.”

The syntactic unit “tabetara” is incorrectly parsed to depend on “anata.” The correct parse is that it depends on “ikuwayo.”

The syntactic unit “kaimononi” is incorrectly parsed to depend on “anata.” The correct parsed is that it depends on “ikuwayo.”

The syntactic unit “ikuwayo” is incorrectly parsed to depend on “anata.” The correct parse is that this unit depends on the root of the sentence with the type “root.”

The syntactic unit “anata” is incorrectly parsed to depend on the root of the sentence. The correct parse is that it depends on “ikuwayo” with the type “appos.”

89jpn.

“Neru-mae-ni	doa-ni	kagi-wo	kake-nasait-te	it-ta-desho.”
Sleep-before-postp	*door-postp*	*key-postp*	*lock-imp.polite-infl*	*say-past-end*

"Didn't I tell you to lock the door before you go to bed?"

The syntactic unit "maeni," "doani," and "kagiwo" are incorrectly parsed to depend on "ittadesho." They must depend on "kakenasaitte," which is an allophone of "kakenasaito."

The syntactic unit "kakenasaitte" is incorrectly segmented in the parse output as follows: "kakena" and "saitte." The correct parse is that it depends on "ittadesho" with the type "ccomp."

97jpn.

"Watashi-ga	hanako-san-nara,	anna	sakeguse-no	warui
I-topic	*Hanako-Ms.-be.infl*	*such*	*drinking.behavior-postp*	*bad*
otoko-to	kekkon-shi-nakatta-wa."			
man-postp	*marry-do-neg.past-end*			

"If I were Hanako, I would not have married such a terrible alcoholic."

The syntactic unit "watakushiga" is incorrectly parsed to depend on "kekkon-shi-nakatta-wa." The correct parse is that it depends on "hanakosannara."

112jpn.

"Kinou	chichi-wa	isha-kara,	sake-to	tabako-wo
Yesterday	*my.father-topic*	*doctor-postp*	*sake-postp*	*tabacco-postp*
yame-te-wa	dou-ka-to	iwa-re-mashi-ta."		
stop-infl-focus	*how-int-ccomp*	*say-passive-polite-past*		

"Yesterday, the doctor recommended that my father give up smoking and drinking."

The syntactic units "kinou," "chichiwa," and "ishakara" are incorrectly parsed to depend on "yametewa;" the correct parse is that they depend on "iwaremashita."

119jpn.

“Izen	watashitachi-wa	hataraku	mise-ga	onaji-de,	yoku
Before	*we-topic*	*work*	*shop-postp*	*same-postp,*	*often*

issho-ni	koohii-wo	non-da-monodesu.”
together-postp	*coffee-postp*	*drink-past-end*

“We used to work in the sampe shop and would often have coffee together.”

The syntactic unit “onajide” is incorrectly parsed to depend on “yoku.” The correct parse is that it depends on “nonda.”

122jpn.

“Kyou	bokura-wa	toukou-shinakute-iin-da,	saijitsu-dakara-ne.”
Today	*we-topic*	*go.to.school-neg-good-be*,	*holiday-postp-end*

“We don’t have to go to school today because it’s a holiday.”

The syntactic units “kyou” and “bokurawa” are incorrectly parsed to depend on “saijitsudakarane.” The correct parse is that they depend on “toukoushinakuteiinda.”

The syntactic unit “toukoushinakuteiinda” is incorrectly parsed to depend on “saijitsudakarane” with the type “advmod.” The correct parse is that it is the root of the sentence.

The syntactic unit “saijitsudakarane” is incorrectly parsed to be the root of the sentence. The correct parse is that it depends on “toukoushinakuteiinda” with the type “advcl_kara.”

123jpn.

“Omaewa,	yakan-ni	sonna	bussou-na	tokoro-wo
You-topic	*during.the.night-postp*	*such*	*dangerous-postp*	*place-postp*

arukimawara-nai	hou-ga	ii.”

walk.around-neg side-postp good

“You had better not walk around in such a dangerous place at night.”

The syntactic unit “yakanni” is incorrectly parsed to depend on “ii.” The correct parse is that it depends on “arukimawaranai.”

124jpn.

“Kooto-wa koko-ni oiteiku hou-ga ii, acchi-wa

Coat-topic here-postp leave side-postp good, there-topic

motto atsui-kara.”

more hot-postp

“You should leave your coat here as it is warmer there.”

The syntactic unit “kootowa” is incorrectly parsed to depend on “atsuikara.” The correct parse is that it depends on “oiteiku.”

The syntactic unit “ii” is incorrectly parsed to depend on the unit “atsuikara” with an incorrect type “advcl.” The correct parse is that it is the root of the sentence.

The syntactic unit “atsuikara” is incorrectly parsed to be the root of the sentence. The correct parse is that it depends on the unit “ii” with the type “advcl_kara.”

129jpn.

“Ano onna-no hanashi-ga hontou-no hazu-ga nai, itsumo

That woman-postp story-postp true-postp case-postp neg, always

uso-bakari tsui-te-iru hito-dakara.”

lie-postp tell-progressive-present person-postp

“What she says can’t be true, because she’s always telling lies.”

The three syntactic units “hontouno,” “hazuga,” and “nai” are incorrectly parsed to be one syntactic unit. The unit “hontouno” depends on “hazuga” with the type “postp_no,” “hazuga” depends on “nai” with the type “postp_ga,” and “nai” is the root of the sentence. Because of these corrections, the syntactic unit “hanashiga” must depend on “hontouno” with the type “postp_ga.”

The syntactic unit “hitodakara” is incorrectly parsed to depend on “hontounohazuganai” with the type “advcl.” The correct parse is that it depends on “nai” with the type “advcl_kara.”

135jpn.

“Watashi-ga	rusu-no	aida,	wasure-zu	mikka-goto-ni
I-postp	*away-postp*	*during,*	*forget-neg*	*three.days-each-postp*
kingyo-ni	esa-wo	yat-te	choudai-ne.”	
goldfish-postp	*feed-postp*	*give-infl*	*imp.polite-end*	

“Please remember to feed the goldfish every three days when I am away.”

The syntactic unit “aida” is incorrectly parsed to depend on “choudaine.” The correct parse is that it depends on “yatte.”

The syntactic unit “wasurezu” is incorrectly parsed to depend on “choudaine” with an incorrect type “dep.” The correct parse is that it depends on “yatte” with the type “advcl.”

The syntactic unit “mikkagotoni” is incorrectly parsed to depend on “choudaine.” The correct parse is that it depends on “yatte.”

136jpn.

“Gaikoku-ryokou-no	tabi-ni,	wakai	koro	motto	eigo-wo
Foreign.countries.trip-postp	*time-postp,*	*young*	*age*	*more*	*English-postp*

benkyou-sure-ba	yokat-ta-to	koukai-suru.”
study-do-infl	*good-past-ccomp*	*regret-do.present*

"Every time I travel abroad, I regret not studying English harder when I was young."

The syntactic unit "tabini" is incorrectly parsed to depend on "benkyousurebayokattato." The correct parse is that it depends on "koukaisuru."

The two syntactic units "benkyousureba" and "yokattato" are incorrectly parsed to be one syntactic unit.

139jpn.

"Anata-wa	osara-wo	arai-sae-sure-ba	ii-noyo."
You-topic	*dish-postp*	*wash-only-do-infl*	*good-end*

"All you have to do is wash the dishes."

The syntactic unit "araisaesurebaiinoyo" must be segmented into two syntactic units "araisaesureba" and "iinoyo." The unit "araisaesureba" depends on "iinoyo."

145jpn.

"Watashi-wa	sono	bin-ni	noriokure-nai-you-ni	kuukou-e
I-topic	*the*	*flight-postp*	*miss-neg-so.that-postp*	*airport-postp*

isoi-da-nodesu-ga,	ososugi-mashi-ta."
hurry-past-end-postp,	*too.late-polite-past*

"I hurried to the airport so as not to miss the flight, but it was too late."

The syntactic unit "noriokurenaiyouni" is correctly parsed to depend on "isoidanodesuga," yet with an incorrect type "dep." The correct parse is "advcl_youni."

146jpn.

"Kaze-wo	hika-nai-you-ni	ki-wo	tsuke-nasai,	raishuu-wa
Cold-postp	*catch-neg-so.that-postp*	*care-postp*	*take-int.polite,*	*next.week-topic*

shuushoku-no	mensetsu-nandakara.”
getting.job-postp	*interview-end*

“Be careful not to catch a cold because you have a job interview next week.”

The syntactic unit “hikanaiyouni” is correctly parsed to depend on “tsukenasai,” but with an incorrect type “ccomp.” The correct type is “advcl_youni.”

The syntactic unit “tsukenasai” is incorrectly parsed to depend on “mensetsunandakara” with an incorrect type “dep.” The correct parse is that it is the root of the sentence.

The syntactic unit “mensetsunandakara” is incorrectly parsed to be the root of the sentence. The correct parse is that it depends on “tsukenasai” with the type “advcl_kara.”

149jpn.

“Sono	hito-ga	watashi-no	chichioya-demo	okashiku-nai	nenrei-to
The	*person-postp*	*I-postp*	*father-focus*	*strange-neg*	*age-ccomp*
kii-te,	watashi-wa	shokku-deshi-ta.”			
hear-infl,	*I-topic*	*shock-polite-past.*			

“I was shocked to hear that he was old enough to be my father.”

The syntactic unit “hitoga” is incorrectly parsed to depend on “kiite.” The correct parse is that it depends on “chichioyademo.”

154jpn.

“Kono	purinta-wa,	shuuri-shi-nakute-wa	ike-mase-n.”
This	*printer-postp,*	*fix-do-neg-postp*	*do-polite-neg*

“This printer needs repairing.”

The syntactic unit “shuurishinakutewaikemasen” in the parsed output must be segmented into two syntactic units “shuurishnakutewa” and “ikemasen,” and the former depends on the latter with the type “advcl_tewa.”

156jpn.

“Kono	eiga-wa	hontou-ni	meisaku-dakara,	nando-mo
This	*movie-postp*	*real-postp*	*masterpiece-postp,*	*many.times-focus*
miru	kachi-ga	aru-to	omou.”	
Watch	*value-postp*	*be-ccomp*	*think.present*	

“This film is a real masterpiece; I think it’s worth watching many times.”

The syntactic unit “eigawa” is incorrectly parsed to depend on “aruto.” The correct parse is that it depends on “meisakudakara.”

The syntactic unit “meisakudakara” is correctly parsed to depend on “aruto,” but with an incorrect type “advcl.” The correct type is “advcl_kara.”

161jpn.

“Shibaraku	at-tei-nakat-ta-node,	tounin-da-to
For.a.long.time	*meet-perfect-neg-past-postp,*	*the.person-be-ccomp*
wakara-nakat-ta.”		
recognize-neg-past		

“Not having seen him for a long time, I failed to recognize him.”

The syntactic unit “tounindato” is correctly parsed to depend on “wakaranakatta,” but with an incorrect type “advcl.” The correct type is “ccomp.”

164jpn.

“Manin-deshi-ta-kara,	watashi-wa eiga-wo
Every.seat.was.taken-polite-past-postp,	*I-topic movie-postp*

tachimi-shimashi-ta."

watch.while.standing-do-past

"I watched the movie standing because every seat was taken."

The syntactic unit "tachimishimashita" is incorrectly parsed to be two syntactic units. This syntactic unit is the dependency head of all the other syntactic units.

168jpn.

"Me-wo	toji-nagara-de-wa,	massugu	aruku	koto-wa
Eye-postp	*close-with-postp-topic*	*straight*	*walk*	*thing-topic*
deki-nai-to	omou."			
able-neg-postp	*think.present*			

"I don't think you can walk in a straight line with you eyes closed."

The syntactic units "aruku," "koto-wa" and "deki-nai-to" are incorrectly parsed to be one syntactic unit. The correct parse is that the unit "aruku" depends on "koto-wa" with the type "rcmod," the unit "koto-wa" depends on "deki-nai-to" with the type "topic," and the unit "deki-nai-to" depends on "omou" with the type "ccomp."

169jpn.

"Sakuya-wa	hidoku	atsukat-ta-node,	eakon-wo	tsuke-ta-mama
Last.night-topic	*terribly*	*hot-past-postp,*	*air.conditioner-postp*	*turn.on-past-postp*
nema-shi-ta."				
sleep-polite-past				

"It was so hot last night that I slept with the air-conditioner on."

The syntactic unit "tsuketamama" is correctly parsed to depend on "nemashita," but with an incorrect type "dep." The correct type is "advcl_mama."

173jpn.

“Amari	chikayora-nai-de,	watashi-no	kaze-ga	utsuru-wayo.”
Too	*come.close-neg-imp,*	*I-postp*	*cold-postp*	*catch-end*

“Don’t come too close, or you’ll catch my cold.”

The syntactic unit “chikayoranaide” is correctly parsed to depend on “utsuruwayo,” but with an incorrect type “dep;” The correct type is “advcl.”

174jpn.

“Watashi-wa	tsukarete-iru	toki-wa	dekakeru	kini-wa
I-topic	*be.tired-present*	*time-focus*	*go.out*	*feeling-focus*
nare-nai.”				
become-neg				

“I can’t bring myself to go out when tired.”

The syntactic unit “watashiwa” is incorrectly parsed to depend on “dekakeru.” The correct parse is that it depends on “narenai.”

178jpn.

“Watashi-wa	ano	hito-ga	kimei-suru-no-wo	miru-made,
I-topic	*that*	*person-postp*	*sign-do-that-postp*	*see-postp,*
hidarikiki-ni		kizuka-nakat-ta.”		
left.handed-postp		*realize-neg-past*		

“I hadn’t realized he was left-handed until I saw him sign his name.”

The syntactic unit “hitoga” is incorrectly parsed to depend on “mirumade.” The correct parse is that it depends on “kimeisuru.”

179jpn.

"Yuubinkyoku-wa,	kimi-ga	tsuku-madeni-wa	shimatte-iru-darou."
Post.office-topic,	*you-postp*	*arrive-postp-focus*	*close-perfect-end*

"The post office will have already closed by the time you get there"

The syntactic unit "yuubinkyokuwa" is incorrectly parsed to depend on "tsukumadeniha." The correct parse is that it depends on "shimatteirudarou."

180jpn.

"Teokure-ni	nara-nai	uchi-ni,	isha-ni	mite-morau
Too.late-postp	*become-neg*	*before-postp,*	*doctor-postp*	*see-receive.present*

hou-ga ii."
side-postp good

"You should see your doctor before it is too late."

The syntactic units "teokureni" and "naranai" are incorrectly parsed to be one syntactic unit "teokureninaranai."

The syntactic unit "uchini" is incorrectly parsed to depend on "ii." The correct parse is that it depends on "mitemorau."

181jpn.

"Ano	ko-wa	kesa	mata	chikoku-shi-mashi-ta."
That	*child-topic*	*this.morning*	*again*	*be.late-do-polite-past*

"He was late again this morning."

The syntactic unit in the parsed output "kesamata" must be segmented into two syntactic units: "kesa" and "mata." These units depend on "chikokushimashita" with the type "advmod."

184jpn.

“Ano	kata-wa	uchi-ni	ko-rareru	toki-wa	kanarazu,
That	*person-topic*	*house-postp*	*come-honorific*	*time-focus*	*every.time,*
musume-no	tame-ni	omiyage-wo	jisan-nasaimasu.”		
daughter-postp	*sake-postp*	*souvenir-postp*	*bring-honorific*		

“Every time he visits us, he brings some present for our daughter.”

The syntactic unit “uchini” is incorrectly parsed to depend on “jisannasaimasu.” The correct parse is such that it depends on “korareru.”

186jpn.

“Ima-wa	kodomotachi-mo	ookiku	natta-kara,	watashitachi-wa	nandemo
Now-topic	*children-focus*	*grown*	*become-postp,*	*we-topic*	*anything*
sukina	koto-ga	deki-ru.”			
like	*thing-postp*	*able-present*			

“Now that our children are grown up, we can do anything we like.”

The syntactic unit “sukinakotogadekiru” in the parsed output must be segmented into three units: “sukina” “kotoga” and “dekiru.” The unit “sukina” depends on “kotoga” with the type “amod;” the unit “kotoga” depends on “dekiru” with the type “postp;” the unit “dekiru” is the root of the sentence.

189jpn.

“Kankyou-ni	junnou-shi-nai	kagiri,	ikinobi-ru	koto-ga
Environment-postp	*adjustment-do-neg*	*unless,*	*survive-present*	*thing-postp*
deki-ru	doubutsu-wa	i-nai.”		
able-present	*animal-focus*	*be-neg.*		

“No animal can survive unless it adjusts to its environment.”

The syntactic unit “ikinobirukotogadekiru” in the parse unit must be segmented into three units: “ikinobiru,” “kotoga,” and “dekiru.” The unit “ikinobiru” depends on “kotoga” with the type “rcmod;” the unit “kotoga” depends on “dekiru” with the type “postp;” the unit “dekiru” is the root of the sentence.

194jpn.

“Uchi-ni-wa	okane-ga	nai-noda-kara,	kuruma-wo	kaikae-ru
Home-postp-topic	*money-postp*	*neg-aux-postp,*	*car-postp*	*buy-present*

koto-wa	deki-nai.”
thing-postp	*able-neg*

“Since we don’t have enough money, we can’t buy a new car.”

The syntactic unit “kaikaerukotowadekinai” in the parsed output must be segmented into three syntactic units: “kaikaeru,” “kotowa,” and “dekinai.” The unit “kaikaeru” depends on “kotoga” with the type “rcmod;” the unit “kotoga” depends on “dekinai” with the type “postp;” the unit “dekinai” is the root of the sentence.

196jpn.

“Mazushii-to	iu-dake-de	hito-wo	mikudasu	koto-ga	nai
Poor-ccomp	*say-focus-postp*	*person-postp*	*look.down.upon*	*thing-postp*	*neg*

you-ni	shi-nasai.”
so.that-postp	*do-imp.polite*

“Don’t look down upon a person just because he or she is poor.”

The syntactic unit “mikudasukotoganaiyounishinasai” in the parsed output must be segmented into five syntactic units: “mikudasu,” “kotoga,” “nai,” “youni,” and “shinasai.” The unit “mikudasu” depends on “kotoga” with the type “rcmod;” the unit “kotoga” depends on “nai” with the type “postp_ga;” the unit “nai” depends on “youni” with the type “rcmod;” the unit “youni” depends on “shinasai” with the type “postp_ni.”

197jpn.

“Asa	motto	hayaku-kara	benkyo-dekiru	you-ni,	goji-ni
Morning	*much*	*earlier-postp*	*study-able*	*so.that-postp,*	*five.o’clock-postp*
mezamashi-wo	setto-shi-mashi-ta.”				
alarm-postp	*set-do-polite-past*				

“I set the alarm for five in the morning so that I could study earlier.”

The syntactic unit “hayakukarabenkyoudekiruyouni” in the parsed output must be segmented into three syntactic units: “hayakukara,” “benkyoudekiru,” and “youni.” The unit “hayakukara” depends on “benkyoudekiru” with the type “postp_kara;” the unit “benkyoudekiru” depends on “youni” with the type “rcmod;” the unit “youni” depends on “settoshimashita” with the type “postp_ni.”

198jpn.

“Nureru-to	ikenai-kara	reinkooto-wo	ki-nasai.”
Get.wet-postp	*bad-postp*	*raincoat-postp*	*wear-imp*

“Wear your raincoat so that you won’t get wet.”

The syntactic unit “nurerutoikenaikara” in the parsed output must be segmented into “nureruto” and “ikenaikara.” The unit “nureroto” depends on “ikenaikara” with the type “advcl.”

199jpn.

“Tochu-de onaka-ga suku-to ikenai-kara, kono

On.the.way-postp	*belly-postp*	*empty-postp*	*bad-postp,*	*this*
sandouicchi-wo	motte-itte-kudasai.”			
sandwiches-postp	*bring-go-imp.polite*			

“Take these sandwiches with you in case you get hungry on the way.”

The syntactic unit “sukutoikenaikara” in the parsed output must be segmented into two syntactic units “sukuto” and “ikenaikara.”

200jpn.

“Tachigiki-sareru-to	ikenai-kara,	watashi-wa	koe-wo
Overhear-passive-postp	*bad-postp,*	*I-topic*	*voice-postp*
hikuku-shi-mashi-ta.”			
lower-do-polite-past			

“I lowered my voice for fear I might be overheard.”

The syntactic unit “tachigikisarerutoikenaikara” in the parsed output must be segmented into two syntactic units “tachigikisareruto” and “ikenaikara.” The unit “tachigikisareruto” depends on “ikenaikara” with the type “advcl.”

201jpn.

“Kono	koohii-wa	kosugi-te,	shoujiki watashi-ni-wa	nome-nai.”
This	*coffee-topic*	*too.strong-postp,*	*honestly I-postp-focus*	*drink-neg*

“This coffee is so strong that I really can’t drink it.”

The syntactic unit “shoujikiwatashiniwa” in the parsed output must be segmented into two syntactic units “shoujiki” and “watashiniwa.” The unit “shoujiki” depends on “nomenai” with the type “advmod.”

207jpn.

“Ano	hito-no	koto-wa,	mukashi	puro-no
That	*person-postp*	*thing-topic,*	*past*	*professional-postp*

kashu-datta	koto-igai-wa	nanimo	shiri-mase-n.”
singer-past	*thing-else-focus*	*nothing*	*know-polite-neg*

“I know nothing about him except that he used to be a professional singer.”

The syntactic unit “mukashi” is incorrectly parsed to depend on “shirimasen.” The correct parse is that it depends on “kashudatta.”

209jpn.

“Kyujo-tai-ga	buji	seikan-suru	mikomi-wa	hotondo	nakatta.”
Rescue-party-postp	*safe*	*come.back-do*	*hope-topic*	*almost*	*neg.past*

“There was little hope that the rescue party would come back safe and sound.”

The syntactic unit “bujiseikansuru” in the parsed output must be segmented into two syntactic units “buji” and “seikansuru.” The unit “buji” depends on “seikansuru” with the type “advmod.”

212jpn.

“Ame-no	naka-de	mata-sare-tsudukeru	koto-ga	donna
Rain-postp	*in-postp*	*wait-passive-keep*	*thing-postp*	*how*

koto-ka,	souzou-shite-mite-hoshii.”
thing-int	*imagine-do-try-want*

“Imagine what it is like to be kept waiting in the rain.”

The syntactic unit “nakade” is incorrectly parsed to depend on “souzousitemitehoshii.” The correct parse is that it depends on “matasaretsuzukeru.”

221jpn.

“Watashi-wa	motto	kaiteki-na	ie-ni	hikkoseru	hi-ga
I-topic	*more*	*comfortable-postp*	*house-postp*	*move*	*day-postp*
kure-ba	ii-to	omou.”			
come-infl	*good-postp*	*think*			

“I hope the day will come when we can move into a more comfortable house.”

The syntactic units “hik” and “koseru” in the parsed output must be one syntactic unit “hikkoseru.”

The syntactic unit “ieni” in the parsed output incorrectly depends on “hik.” The correct parse is that it depends on “hikkoseru” with the type “postp.”

The syntactic unit “kurebaiito” in the parsed output must be segmented into two syntactic units: “kureba” and “iito.” The unit “kureba” depends on “iito” with the type “advcl.”

231jpn.

“Ano	hito-wa,	yara-nakute-wa	ikenai	koto-wo	yara-zu-ni
That	*person-topic,*	*do-neg-focus*	*bad*	*thing-postp*	*do-neg-postp*
iru	koto-ga	ooi.”			
be.present	*thing-postp*	*often*			

“He often leaves undone what he ought to do.”

The syntactic unit “yaranakutewaikenai” in the parsed output must be segmented into “yaranakutewa” and “ikenai.” The unit “yaranakutewa” depends on “ikenai” with the type “topic.”

The syntactic unit “yarazuni” is incorrectly parsed to depend on “ooi.” The correct parse is that it depends on “iru” with the type “advcl_ni.”

245jpn.

“Hillary-wa	shinki-saiyou-shita	hoka-no	dare-yori-mo
Hillary-topic	*newly-hire-past*	*other-postp*	*anyone-more-focus*
seiryoku-teki-desu.”			
energetic-suf-polite.present			

“Hillary is more energetic than any other newly hired employee.”

The syntactic unit “shinkisaiyoushita” is incorrectly parsed to depend on “hokano.” The correct parse is that it depends on “dareyorimo.”

253jpn.

“Fudan chichi-wa,	hitsuyou-gaku-wo	koeru	o-kane-wo
Usually father-topic,	*need-amount-postp*	*surpass*	*prefix-money-postp*
mochiaruki-mase-n.”			
carry-polite-neg			

“My father usually does not carry more money than he needs.”

The syntactic unit “fudanchichiwa” in the parsed output must be segmented into two syntactic units “fudan” and “chichiwa.” Both units depend on “mochiarukimasen.” The type of the former is “advmod,” and the type of the latter is “topic.”

254jpn.

“Watashi-wa,	konnani	oishii	orenji-juusu-wo	nonda
I-topic,	*such*	*delicious*	*orange-juice-postp*	*drink.past*
koto-ga	ari-mase-n.”			
thing-postp	*be-polite-neg*			

“This is the most delicious orange juice I have ever drunk.”

The syntactic unit “nondakotogaarimasen” in the parsed output must be segmented into three syntactic units “nonda”, “kotoga”, and “arimasen.” The unit “nonda” depends on “kotoga” with the type “rcmod;” the unit “kotoga” depends on “arimasen” with the type “postp_ga;” the unit “arimasen” is the root of the sentence.

255jpn.

“Sore-wa	watashitachi-ni-totte	gojuu-nen-buri-no	oo-jishin-deshi-ta.”
That-topic	*we-postp-postp*	*50-year-in-postp*	*big-earthquake-polite-past*

“That was the biggest earthquake we had had in the past fifty years.”

The syntactic unit “watashitachinitotte” is incorrectly segmented into two syntactic units “watashitachini” and “totte.”

260jpn.

“Watashi-wa	ano	hito-ni	hanashi-kake-mo	si-nakat-ta-nodesu,	
I-topic	*that*	*person-postp*	*speak-to-focus*	*do-neg-past-polite,*	
mashiteya	sono	koto-wo	hanashiau	koto-ga	nakat-ta-no-wa
even	*the*	*thing-postp*	*talk.over*	*thing-postp*	*neg-past-no-topic*

mochiron-desu.”

matter.of.fact-polite.present

“I didn’t even speak to him, much less talk it over with him.”

The syntactic units “hanashikakemo” and “shinakattanodesu” are incorrectly parsed to be one single syntactic unit, which is incorrectly parsed to depend on “mashiteya.” The correct parse is that “hanashikakemo” depends on “shinakattanodesu” with the type “focus”, and “shinakattanodesu” is the root of the sentence.

The syntactic units “hanashiau”, “kotoga” and “nakattanowa” are incorrectly parsed to be one single syntactic unit. The unit “hanashiau” depends on “kotoga” with the type “rcmod”, “kotoga” depends on “nakattanowa” with the type “postp_ga”, and “nakattanowa” depends on “mochirondesu” with the type “topic.”

263jpn.

“Kanja-wa,	kusuri-wo	nonda	kai-mo	naku,
Patient-topic,	*medicine-postp*	*take.past*	*effect-focus*	*neg,*
sukoshimo	yoku-nara-nakatta.”			
at.all	*better-get-neg.past*			

“The patient was none the better for taking the medicine.”

The syntactic unit “kanjawa” is incorrectly parsed to depend on “naku.” The correct parse is that it depends on “yokunaranakatta.”

265jpn.

“Tashikani	ai-wa	taisetsudearu-ga,	o-kane-mo	sore-ni
For.sure	*love-topic*	*important-postp,*	*prefix-money-focus*	*it-postp*
otora-zu	taisetsudearu.”			
less-neg	*important*			

“It is true that love is important, but still, money is no less important.”

The syntactic unit “otorazu” is correctly parsed to depend on “taisetsudearu”, but with an incorrect type “dep.” The correct type is “advcl.”

270jpn.

“Sekai	isshu-no	funatabi-ni-wa,	bakudaina	o-kane-ga
World	*around-postp*	*ship.travel-postp-topic,*	*a.huge.sume*	*prefix-money-postp*

kakaru.”

cost.present

“It costs a huge sum of money to travel around the world by ship.”

The syntactic unit “sekai” is correctly parsed to depend on “isshuu”, but with an incorrect type “dep.” The correct type is “nn.”

300jpn.

“watashi-ni-wa,	sono	otoko-ga	doko-no	dare-nanoka-mo	sumai-ga
I-postp-topic,	*the*	*person-postp*	*where-postp*	*who-int-focus*	*house-postp*
doko-nanoka-mo		wakari-mase-n.”			
where-int-focus		*know-polite-neg*			

“I don’t know who or what he is, or where he lives.”

The syntactic unit “dokonanokamowakarimasen” in the parsed output should be segmented into two units “dokonanokamo” and “wakarimasen.” The unit “dokonanokamo” depends on “wakarimasen” with the type “focus.”

304jpn.

“Nietagiru	o-yu-wa	nome-nai,	sonna	koto-wo
Boiling	*prefix-hot.water-topic*	*drink-neg,*	*such*	*thing-postp*
shi-tara	yakedo-suru.”			
do-conditional	*burn-do.present*			

“You cannot drink boiling water; if you do, you will burn yourself.”

The syntactic unit “oyuwa” is incorrectly parsed to depend on “yakedosuru.” The correct parse is that it depends on “nomenai.”

The syntactic unit “nomenai” is incorrectly parsed to depend on “kotowo” with an incorrect type “rcmod.” The correct parse is that it is a root of the sentence.

308jpn.

“Imoto-wa,	saakasu-de	miru-made	zou-wo	mi-ta
Younger.sister-topic,	*circus-postp*	*see-postp*	*elephant-postp*	*see-past*

koto-ga	nakattanodesu.”
thing-postp	*neg.polite*

“My sister had never seen an elephant until she saw one at the circus.”

The syntactic unit “mitakotoganakattanodesu” in the parsed output should be segmented into two syntactic units “mita”, “kotoga” and “nakattanodesu.” The unit “mita” depends on “kotoga” with the type “rcmod.” The unit “kotoga” depends on “nakattanodesu” with the type “postp_ga.” The unit “nakattanodesu” is the root of the sentence.

312jpn.

“Watashi-ga	oshie-te-iru	seito-ni-wa,	suugaku-ga	suki-na
I-postp	*teach-progressive-present*	*student-postp-topic,*	*math-postp*	*like*

mono-mo	ireba,	soudenai	mono-mo iru.”
one-focus	*be,*	*not.so*	*one-focus be.*

“Some of my students like math, and others don’t”

The syntactic unit “soudenai” is correctly parsed to depend on “monomo”, but with an incorrect type “dep.” The correct type is “rcmod.”

319jpn

“Watashi-ga	kanyuu-shite-iru	kurabu-no	gen-kaiin-wa,
I-postp	*belong-do-present*	*club-postp*	*prefix-member-topic,*

subete	dansei-desu.”
all	*male-polite.present*

“The present members of the club I belong to are all men.”

The syntactic unit “watashiga” is incorrectly parsed to depend on “danseidesu.” The correct parse is that it depends on “kanyushiteiru.”

321jpn.

“Watashitachi-wa,	seito-ga	nan-nin-ka	kure-ba	ii-to
We-topic,	*student-postp*	*some-people-int*	*come-infl*	*good*

omot-ta-ga,	jissaiwa	hitori-mo	ko-nakat-ta.”
think-past-postp,	*actually*	*one-focus*	*come-neg-past*

“We had hoped some students would come, but actually, none did.”

The syntactic unit “watashiatchiwa” is incorrectly parsed to depend on “konakatta.” The correct parse is that it depends on “omottaga.”

The syntactic unit “kurebaiito” in the parsed output must be segmented into “kureba” and “iito.” The unit “kureba” depends on “iito” with the type “advcl.” The unit “iito” depends on “omottaga” with the type “ccomp.”

323jpn

“Ano	komedi-wa	omosiroku-nai-dokoroka,	sukoshimo	okashiku-nai.”
That	*comedy-postp*	*amusing-neg-postp,*	*at.all*	*funny-neg*

“That comedy isn't amusing; in fact, it's anything but funny.”

The syntactic unit “omoshirokunaidokoroka” is correctly parsed to depend on “okashikunai”, but with an incorrect type “dep.” The correct type is “advcl_dokoroka.”

337jpn

“Jikan-ga	amari	nakat-ta-node,	watashi-wa	choushoku-wo
Time-postp	*much*	*neg-past-postp,*	*I-topic*	*breakfast-postp*
tabe-nai-de	ie-wo	de-mashi-ta.”		
eat-neg-postp	*house-postp*	*leave-polite-past*		

“I left home without having breakfast because I didn't have much time.”

The syntactic unit “tabenaide” is correctly parsed to depend on “demashita”, but with an incorrect type “dep.” The correct type is “advcl_de.”

339jpn

“Sukoshi	netsu-ga	aru	koto-wo	betsu-ni	sure-ba,
A.little	*fever-postp*	*be.present*	*thing-postp*	*except-postp*	*do-infl,*
Tarou-no	byouki-wa	taishita	koto-wa	na-sasou-desu.”	
Tarou-postp	*illness-topic*	*very*	*thing-focus*	*neg-seem-polite.present*	

“Except for a slight fever, Taro doesn't seem to be very ill.”

The syntactic unit “betsunisureba” in the parsed output should be segmented into two units “betsuni” and “sureba.” The unit “betsuni” depends on “sureba” with the type “postp_ni,” and the unit “sureba” depends on “nasasoudesu” with the type “advcl_ba.”

Appendix IV: The Ruby script for converting Stanford Parser output into Pajek .net files

```
#! ruby -Ks


triple = File.read("#{ARGV[0]}")
folder = "#{ARGV[0]}".gsub(/¥.txt/,"")
if !File.exist?(folder)
Dir.mkdir(folder)
end
id = 0
all = Array.new(0)
triple.split(/¥n¥n/).each{|i|
      vertices = ""
      arcs = "*Arcs¥n"
      outgoing = Array.new(0)
      incoming = Array.new(0)
      num = Array.new(0)
      snt_id = 1
      i.to_s.split(/¥n/).each{|j|
            if j.to_s.match(/^[A-Za-z]+(_[A-Za-z]+)*¥(.+-/)

      j.to_s.match(/^(.+?)¥((.+?)-([0-9]+),¥s(.+?)-([0-9]+)¥)$/)
                  if $3!=nil && $5!=nil
                  outgoing.push("#{$2}-#{$3.to_i+1}")
                  incoming.push("#{$4}-#{$5.to_i+1}")
                  vertices << "#{$5.to_i+1}¥s¥"#{$4}¥"¥n"
                  arcs                                            <<
"#{$3.to_i+1}¥s#{$5.to_i+1}¥s1¥sl¥s¥"#{$1}¥"¥n"
                  last = $5.to_i
```

```
                    num.push("#{$3}".to_i)
                    num.push("#{$5}".to_i)
                        end
              end
       }
       (outgoing.uniq                                            -
incoming.uniq).to_s.match(/^(.+?)-([0-9]+)/)
        if (outgoing.uniq - incoming.uniq).to_s=="ROOT-1"
           vertices = "*Vertices¥s#{num.max+1}¥n#{$2}¥s¥"#{$1}¥"¥n"
+ vertices if num.max!=nil&&vertices!=nil
           all.push(vertices+arcs)
        else
           vertices = "*Vertices¥s#{num.max+1}¥n#{$2}¥s¥"#{$1}¥"¥n"
+ vertices if num.max!=nil&&vertices!=nil
           all.push(vertices+arcs)
           puts "#{i}¥n¥n"
           puts id
        end
       id += 1
}
index = 0
all.each{|a|
   File.open("#{folder}/#{index}.net", "w"){|file|
      file.puts(a.to_s)
   }
   index += 1
}
```

Appendix V: The Ruby script for converting KNP output into Stanford-Parser-style typed-dependency triples

```
#! ruby -Ks

sentences = File.read("#{ARGV[0]}").split(/EOS/)
results=""
sentences.each{|a|
  i = -1
  units = Array.new(0)
  a.to_s.split(/¥n¥*/).each{|b|
    lex = ""
    cs = ""
    b.to_s.split(/¥n/).each{|c|
      if c.to_s.match(/^¥s/)
      elsif c.to_s.match(/^¥+/)
        if c.to_s.index("格解析結果")
          c.to_s.match(/格解析結果:(.+)/)
          $1.to_s.match(/(ガ.+)/)
          $1.to_s.split(/;/).each{|d|
            cs << "#{d.to_s}|" if d.to_s.index("ガ
/")||d.to_s.index("ヲ/")||d.to_s.index("ニ/")
          }
        end
      elsif c.to_s.match(/^#/)
      else
        c.to_s.match(/(.+?)¥s/)
        lex << "#{$1.to_s}|" if $1!=nil
      end
```

```
     }
     puts "#{lex}¥s#{cs}"
     prt = ""
     typ=""
     b.to_s.match(/^¥s(.*?)[D|P|A|I]/)
     h = $1.to_i+1
     if h==0
       typ="root"
     else
       b.to_s.match(/(.+?)¥n/)
       if $1.to_s.index("¥s<用言")||$1.to_s.index("否定表現><用言
")||$1.to_s.index("可能表現><用言")
         if $1.to_s.index("<連体節")
           if $1.to_s.index("用言:動")
             typ="rcmod"
           elsif $1.to_s.index("用言:形")
             typ="amod"
           else
             typ="dep"
           end
         elsif $1.to_s.index("<連用節")
           if $1.to_s.index("動詞連用")
             typ="advcl"
           else
             t = $1.to_s
             t.match(/ID:(.+?)>/)
             if $1.to_s.index("（も）")
               typ="advcl_ba"
```

```
elsif $1.to_s.index("ば")
   typ="advcl_ba"
elsif $1.to_s.index("て（用言）")
   typ="advcl_te"
elsif $1.to_s.index("ては")
   typ="advcl_tewa"
elsif $1.to_s.index("くても")
   typ="advcl_kutemo"
elsif $1.to_s.index("くて")
   typ="advcl_kute"
elsif $1.to_s.index("ても")
   typ="advcl_temo"
elsif $1.to_s.index("けれども")
   typ="advcl_keredomo"
elsif $1.to_s.index("～が")
   typ="advcl_ga"
elsif $1.to_s.index("～と")
   typ="advcl_to"
elsif $1.to_s.index("～ので")
   typ="advcl_node"
elsif $1.to_s.index("～から")
   typ="advcl_kara"
elsif $1.to_s.index("～たら")
   typ="advcl_tara"
elsif $1.to_s.index("～て")
   typ="advcl_te"
elsif $1.to_s.index("～ながら")
   typ="advcl_nagara"
```

```
        elsif $1.to_s.index("～のだから")
          typ="advcl_nodakara"
        elsif $1.to_s.index("～であって")
          typ="advcl_deatte"
        elsif $1.to_s.index("～で")
          typ="advcl_de"
        elsif $1.to_s.index("～より")
          typ="advcl_yori"
        elsif $1.to_s.index("～まで")
          typ="advcl_made"
        elsif $1.to_s.index("～く")
          typ="advcl_ku"
        elsif $1.to_s.index("～に")
          typ="advcl_ni"
        elsif $1.to_s.index("～または")
          typ="advcl_matawa"
        elsif $1.to_s.index("～こうが")
          typ="advcl_kouga"
        else
          typ="dep"
        end
      end
    else
      if $1.to_s.index("ト格")
        typ="ccomp"
      elsif $1.to_s.index("係:複合辞連用")
        typ="postp_nituite_niyotte"
      elsif $1.to_s.index("係:連用")
```

```
  typ="advmod"
elsif $1.to_s.index("係:連体")
  typ="rcmod"
elsif $1.to_s.index("ガ格")
  typ="postp_ga"
elsif $1.to_s.index("ヲ格")
  typ="postp_wo"
elsif $1.to_s.index("ニ格")
  typ="postp_ni"
elsif $1.to_s.index("ノ格")
  typ="postp_no"
elsif $1.to_s.index("デ格")
  typ="postp_de"
elsif $1.to_s.index("カラ格")
  typ="postp_kara"
elsif $1.to_s.index("ヨリ格")
  typ="postp_yori"
elsif $1.to_s.index("ト格")
  typ="postp_to"
elsif $1.to_s.index("ヘ格")
  typ="postp_he"
elsif $1.to_s.index("無格")
  typ="advmod"
elsif $1.to_s.index("未格")
  if b.to_s.index("は 助詞")
    typ = "topic"
  else
    typ = "focus"
```

```
          end
        elsif $1.to_s.index("隣")
          typ="dep"
        elsif $1.to_s.index("連体:ヤ")
          typ="conj"
        elsif $1.to_s.index("同格連体")
          typ="appos"
        elsif $1.to_s.index("連体")
          typ="advmod"
        elsif $1.to_s.index("NONE")
          typ="advmod"
        else
          typ="dep"
        end
      end
    elsif $1.to_s.index("¥s<体言")||$1.to_s.index("否定表現><体
言")||$1.to_s.index("可能表現><体言")
      if $1.to_s.index("<用言:判>")
        if $1.to_s.index("ト格")
          typ="ccomp"
        elsif $1.to_s.index("連用")
          typ="advcl"
        elsif $1.to_s.index("連体")
          typ="rcmod"
        else
          typ="focus"
        end
      elsif $1.to_s.index("<係:連用>")
```

```
  typ="advmod"
else
  if $1.to_s.index("ガ格")
    typ="postp_ga"
  elsif $1.to_s.index("ヲ格")
    typ="postp_wo"
  elsif $1.to_s.index("ニ格")
    typ="postp_ni"
  elsif $1.to_s.index("ノ格")
    typ="postp_no"
  elsif $1.to_s.index("デ格")
    typ="postp_de"
  elsif $1.to_s.index("カラ格")
    typ="postp_kara"
  elsif $1.to_s.index("ヨリ格")
    typ="postp_yori"
  elsif $1.to_s.index("ト格")
    typ="postp_to"
  elsif $1.to_s.index("ヘ格")
    typ="postp_he"
  elsif $1.to_s.index("無格")
    typ="advmod"
  elsif $1.to_s.index("未格")
    if b.to_s.index("は 助詞")
      typ = "topic"
    else
      typ = "focus"
    end
```

```
        elsif $1.to_s.index("隣")
          typ="dep"
        elsif $1.to_s.index("連体:ヤ")
          typ="conj"
        elsif $1.to_s.index("同格連体")
          typ="appos"
        elsif $1.to_s.index("連体")
          typ="advmod"
        else
          typ="dep"
        end
      end
    else
      if $1.to_s.index("連体")
        typ="det"
      else
        typ="advmod"
      end
    end
  end
  i+=1
units.push("#{typ}¥s#{h}¥s#{lex}-#{i}") if lex!=""
}
triple=""
units.each{|a|
 # puts a
  h = ""
  a.to_s.match(/(.+?)¥s(.+?)¥s(.+)/)
```

```
      typ=$1.to_s
      hnum=$2.to_s
      tail=$3.to_s
      if typ!="root"
      units.each{|b|
        b.to_s.match(/(.+?)¥s(.+?)¥s(.+)/)
        h=$3.to_s
        h.match(/-(.*)/)
        if $1.to_s==hnum
          results << "#{typ}(#{h},¥s#{tail})¥n"
        end
      }
      else
        results << "root(ROOT-0,¥s#{tail})¥n"
      end
    }
    results << "¥n"
}
File.open("#{ARGV[1]}", "w"){|file|
       file.puts("#{results}")
}
```

Appendix VI: The Ruby script for calculating the precision, recall, and f-score of each dependency type

```
require "find"

before = Hash.new(0)
Find.find(File.expand_path("F:/data/Basic300_2012/NetKNPConjFixK
NP")){|path|
  next unless File.file?(path)
  path.to_s.match(/([0-9]+)eng¥.net/)
  id = $1.to_s if $1!=nil
  net = File.read(path)
  before.store(id, net.split(/¥*Arcs/)[1])
}

after = Hash.new(0)
Find.find(File.expand_path("F:/data/Basic300_2012/Fixed")){|path
|
  next unless File.file?(path)
  path.to_s.match(/([0-9]+)eng¥.net/)
  id = $1.to_s  if $1!=nil
  net = File.read(path)
  after.store(id, net.split(/¥*Arcs/)[1])
}

wrongType = Hash.new(0)
failedType = Hash.new(0)
correctType = Hash.new(0)
```

```
list="id¥tbefore¥tafter¥n"
listc="id¥tbefore¥tafter¥n"
num=0.0
num2=0.0
before.each{|k1,v1|
  if k1!=nil
  tails1=Hash.new(0)
  tails2=Hash.new(0)
  v1.each{|a|
    a.to_s.match(/^[0-9]+¥s([0-9]+)¥s/)
    if $1!=nil
      tails1.store($1.to_s, a.to_s)
    end
  }
  after[k1].each{|b|
    b.to_s.match(/^[0-9]+¥s([0-9]+)¥s/)
    if $1!=nil
      tails2.store($1.to_s, b.to_s)
      num+=1
    end
  }

  tails1.each{|k2,v2|
    if !tails2.include?(k2)
      list<<"#{k1}¥t#{v2.gsub(/¥n/,"")}¥tnone¥n"
    elsif v2!=tails2[k2]

list<<"#{k1}¥t#{v2.gsub(/¥n/,"")}¥t#{tails2[k2].gsub(/¥n/,"")}¥n
"
```

```
      else

listc<<"#{k1}¥t#{v2.gsub(/¥n/,"")}¥t#{tails2[k2].gsub(/¥n/,"")}¥
n"
      end
   }

   tails2.each{|k3,v3|
      if !tails1.include?(k3)
         list<<"#{k1}¥tnone¥t#{tails2[k3].gsub(/¥n/,"")}¥n"
      end
   }
   end
}

listNew=""

list.each{|a|
   if a!=nil
   listNew<<"#{a.to_s.gsub(/¥n/,"¥t")}"
   a.to_s.match(/^[0-9]+¥t(.+)¥t(.+)/)
   before = $1.to_s
   after = $2.to_s
   before.match(/^(.+)(".+")/)
   bnum=$1
   btyp=$2
   after.match(/^(.+)(".+")/)
   anum=$1
   atyp=$2
```

```
  puts atyp
  if !wrongType.include?(btyp)
    wrongType.store(btyp, 1)
  else
    i=wrongType[btyp]+1
    wrongType.store(btyp,i)
  end
  if !failedType.include?(atyp)
    failedType.store(atyp,1)
  else
    i=failedType[atyp]+1
    failedType.store(atyp,i)
  end
  if btyp!=atyp
    if bnum!=anum
       listNew<<"wt,wr"
    else
       listNew<<"wt"
    end
  else
    listNew<<"wr"
  end
  listNew<<"¥n"
  end
}

File.open("F:/data/Basic300_2012/wrongRelType.txt", "w"){|file|
  file.puts(listNew)
```

```
}

listc.each{|a|
  if a!=nil
    a.to_s.match(/(".+?")/)
    if !correctType.include?($1)
      correctType.store($1,1)
    else
      i=correctType[$1]+1
      correctType.store($1,i)
    end
  end
}
precrec="type¥tcorrect¥tincorrect¥tfailed¥tP¥tR¥tF¥n"
correctType.each{|k,v|
  precision=v.to_f/(v.to_f+wrongType[k].to_f)
  recall=v.to_f/(v.to_f+failedType[k].to_f)
  fscore=2*precision*recall/(precision+recall)

precrec<<"#{k}¥t#{v}¥t#{wrongType[k]}¥t#{failedType[k]}¥t#{preci
sion}¥t#{recall}¥t#{fscore}¥n" if k!=nil
}

failedType.each{|k,v|
  if !correctType.include?(k)

precrec<<"#{k}¥t0¥t#{wrongType[k]}¥t#{failedType[k]}¥t0¥t0¥t0¥n"
  end
}
```

```
File.open("F:/data/Basic300_2012/depTypePrecRecNew.txt",
"w"){|file|
   file.puts(precrec)
}
```

Appendix VII: The Ruby script for calculating the degree centrality, the closeness centrality, and the average dependency distance from the Pajek-style .net files of the typed-dependency trees

```
require "find"
folder = "#{ARGV[0]}".gsub(/¥.stp/,"")
if !File.exist?(folder)
  Dir.mkdir(folder)
end
class Array
#http://d.hatena.ne.jp/sesejun/20070502/p1
  def sum_with_number
    s = 0.0
    n = 0
    self.each do |v|
      next if v.nil?
      s += v.to_f
      n += 1
    end
    [s, n]
  end

  def sum
    s, n = self.sum_with_number
    s
  end

  def avg
    s, n = self.sum_with_number
    s / n
  end
  alias mean avg

  def var
    c = 0
    while self[c].nil?
      c += 1
    end
    mean = self[c].to_f
    sum = 0.0
    n = 1
    (c+1).upto(self.size-1) do |i|
      next if self[i].nil?
      sweep = n.to_f / (n + 1.0)
      delta = self[i].to_f - mean
      sum += delta * delta * sweep
      mean += delta / (n + 1.0)
      n += 1
    end
    sum / n.to_f
  end

  def stddev
    Math.sqrt(self.var)
  end
```

```
def corrcoef(y)
  raise "Invalid Argument Array Size" unless self.size == y.size
  sum_sq_x = 0.0
  sum_sq_y = 0.0
  sum_coproduct = 0.0
  c = 0
  while self[c].nil? || y[c].nil?
    c += 1
  end
  mean_x = self[c].to_f
  mean_y = y[c].to_f
  n = 1
  (c+1).upto(self.size-1) do |i|
    next if self[i].nil? || y[i].nil?
    sweep = n.to_f / (n + 1.0)
    delta_x = self[i].to_f - mean_x
    delta_y = y[i].to_f - mean_y
    sum_sq_x += delta_x * delta_x * sweep
    sum_sq_y += delta_y * delta_y * sweep
    sum_coproduct += delta_x * delta_y * sweep
    mean_x += delta_x / (n + 1.0)
    mean_y += delta_y / (n + 1.0)
    n += 1
  end
  pop_sd_x = Math.sqrt(sum_sq_x / n.to_f)
  pop_sd_y = Math.sqrt(sum_sq_y / n.to_f)
  cov_x_y = sum_coproduct / n.to_f
  cov_x_y / (pop_sd_x * pop_sd_y)
  end

end

class Degree
   def initialize(net)
      @net = net
      @num = 0
   end

   def numget
      i=0
      @net.split(/¥*Arcs/)[0].split(/¥n/).each{|a|
         i+=1
      }
      @num = i-1
      return @num
   end

   def set
      v = Hash.new(0)
      @net.split(/¥*Arcs/)[1].split(/¥n/).each{|a|
         a.to_s.match(/^([0-9]+)¥s([0-9]+)¥s/)
         if $1!=nil
            if v.include?($1.to_s)
               i = v[$1.to_s]
               v.store($1.to_s, i+1)
            else
```

```
          v.store($1.to_s, 1)
        end
        if v.include?($2.to_s)
          i = v[$2.to_s]
          v.store($2.to_s, i+1)
        else
          v.store($2.to_s, 1)
        end
      end
    }
    difSum=0.0
    maxDegSum=0.0
    maxDeg=v.max{|key,value|key[1]<=>value[1]}[1]
    v.each{|key,value|
      difSum+=maxDeg.to_f - value.to_f
    }
    return difSum / ((v.length-2)*(v.length-1)).to_f
  end
end

class Closeness
  def initialize(net)
    @net = net
    @num = 0
  end

  def set
    @left = ""
    @right = ""
    @edges = @net.split(/¥*Arcs/)[1].split(/¥n/)
    sum = 0.0
    @edges.each{|a|
      a.to_s.match(/^([0-9]+)¥s([0-9]+)¥s/)
      @left = $1.to_s
      @right = $2.to_s
      i = 1
      j = 1
      begin
        @edges.each{|b|
          if b.to_s.match(/^[0-9]+¥s#{@left}¥s/)
            b.to_s.match(/^([0-9]+)¥s#{@left}¥s/)
            @left = $1.to_s
            j += 1
          else
            next
          end
        }
        i += 1
      end while i < @edges.length + 1
       sum += j
    }
    return @edges.length / sum
  end
end
```

#the first argument：the path of the folder containing the parsed

```
output file named *.net
sntnum = 0
sntdg = 0.0
dccc = "¥tdc¥twc¥tcc¥twc¥n"
depDist="filename¥tedges¥taverage¥n"
eachDeptypeAll = "type¥toccurrence¥taverage¥tSD¥n"
all = Array.new(0)
id = 1
depType2 = Hash.new(0)
depType3 = Hash.new(0)
depTypeAll = Hash.new(0)
Find.find(File.expand_path("#{ARGV[0]}")){|path|
  next unless File.file?(path)
    d = Degree.new(File.read(path))
    c = Closeness.new(File.read(path))
    n = d.numget
    i = d.set
    j= c.set
    dccc << "#{path.to_s}¥t#{i}¥t#{n}¥t#{j}¥t#{n}¥n"

    edges = 0.0
    sum = 0.0
    File.read(path).split(/¥*Arcs/)[1].split(/¥n/).each{|a|
      head = 0
      tail = 0
      a.to_s.match(/^([0-9]+)¥s([0-9]+)¥s1¥sl¥s¥"(.+)¥"/)
      if $1!=nil&&$2!=nil&&$3!=nil
        edges+=1
```

```
head=$1.to_f
tail=$2.to_f
type=$3.to_s
sum += (head-tail).abs
if j.to_f < 0.4
  if !depType2.include?(type)
    number = Array.new(0)
    number.push((head - tail).abs)
    depType2.store(type,number)
  else
    number = depType2[type].push((head - tail).abs)
    depType2.store(type, number)
  end
else
  if !depType3.include?(type)
    number = Array.new(0)
    number.push((head - tail).abs)
    depType3.store(type,number)
  else
    number = depType3[type].push((head - tail).abs)
    depType3.store(type, number)
  end
end
if !depTypeAll.include?(type)
  number = Array.new(0)
  number.push((head - tail).abs)
  depTypeAll.store(type,number)
else
```

```
          number = depTypeAll[type].push((head - tail).abs)
          depTypeAll.store(type, number)
        end
      end
    }
  depDist << "#{path}¥t#{edges}¥t#{(sum / edges)}¥n"
}
#degree centralities and closeness centralities
File.open("#{folder}/DCCC.txt", "w"){|file|
  file.puts("#{dccc}")
}

#average dependency distance
File.open("#{folder}/depDistance.txt", "w"){|file|
     file.puts("#{depDist}")
}

depTypeAll.sort.each{|k, v|
  eachDeptypeAll << "#{k}¥t#{v.length}¥t#{v.avg}¥t#{v.stddev}¥n"
}

File.open("#{folder}/depDistEachTypeAll.txt", "w"){|file|
  file.puts("#{eachDeptypeAll}")
}
```

Appendix VIII: The degree centralities and closeness centralities in each section of MASC 500K

Blog section

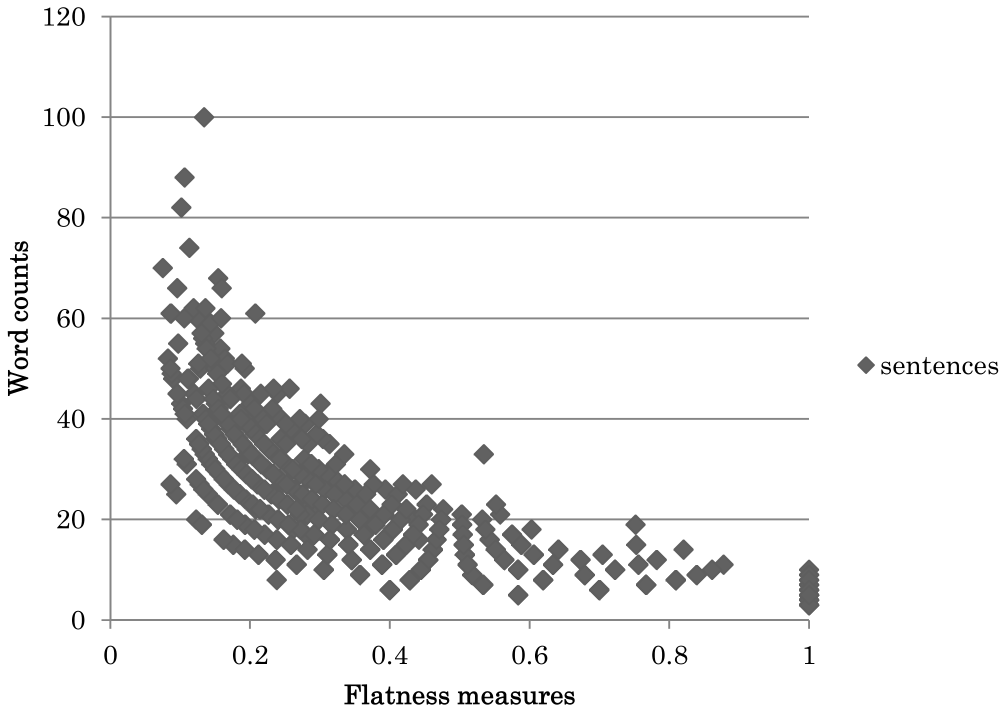


Figure VIII.1. The distribution of sentences in the blog section (n=1524) in terms of their flatness measures and word counts

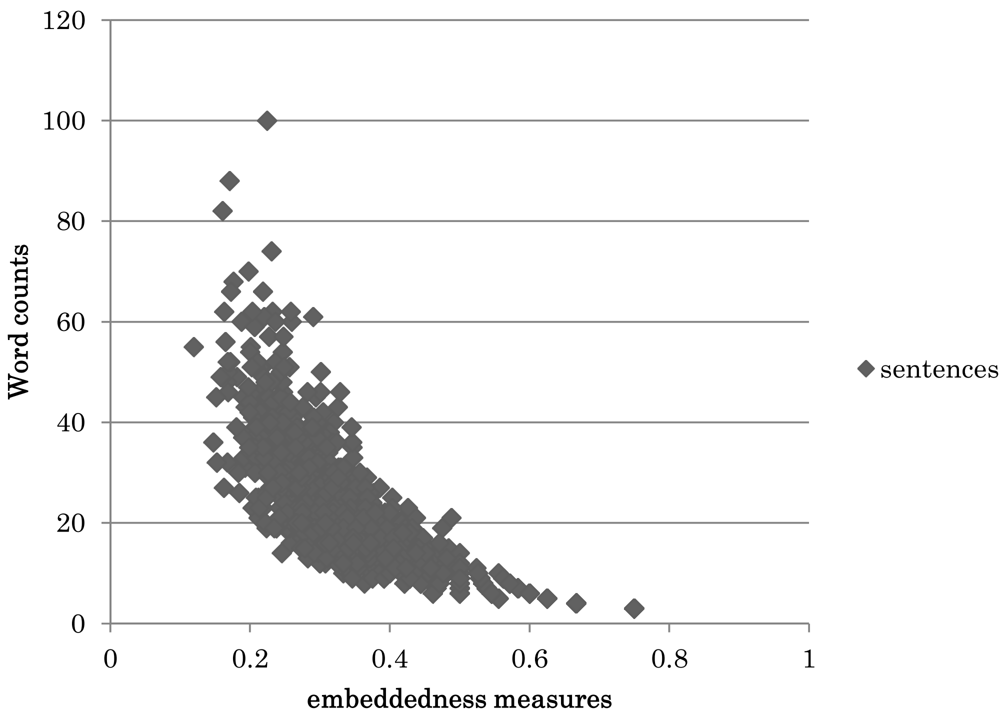


Figure VIII.2. The distribution of sentences in the blog section (n=1524) in terms of their embeddedness measures and word counts

Essay section

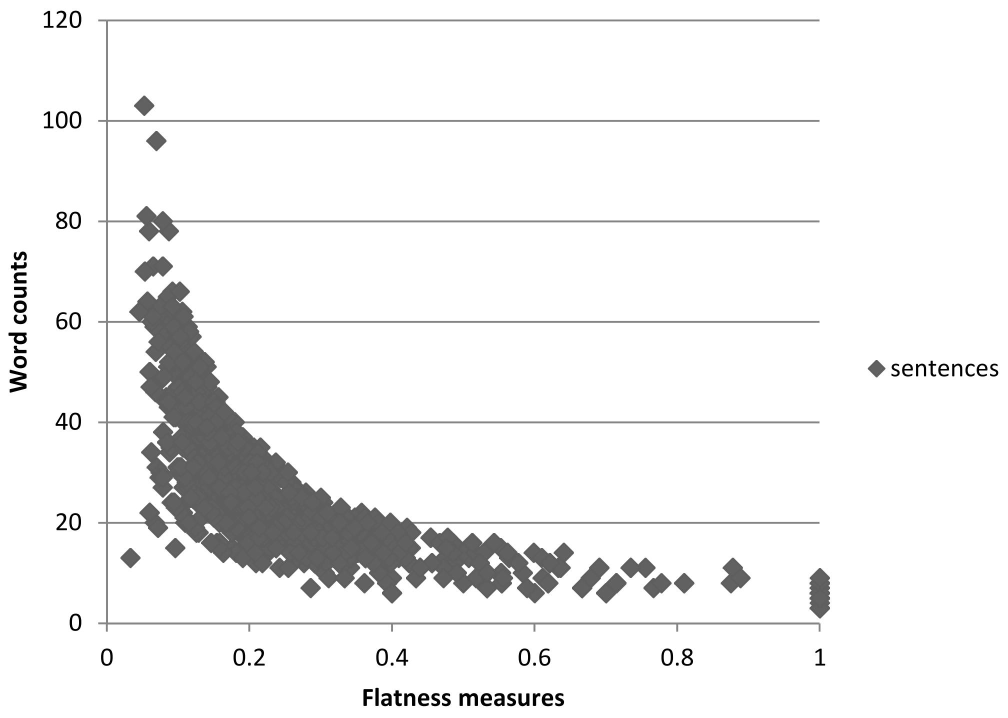


Figure VIII.3. The distribution of sentences in the essay section (n=1072) in terms of their flatness measures and word counts

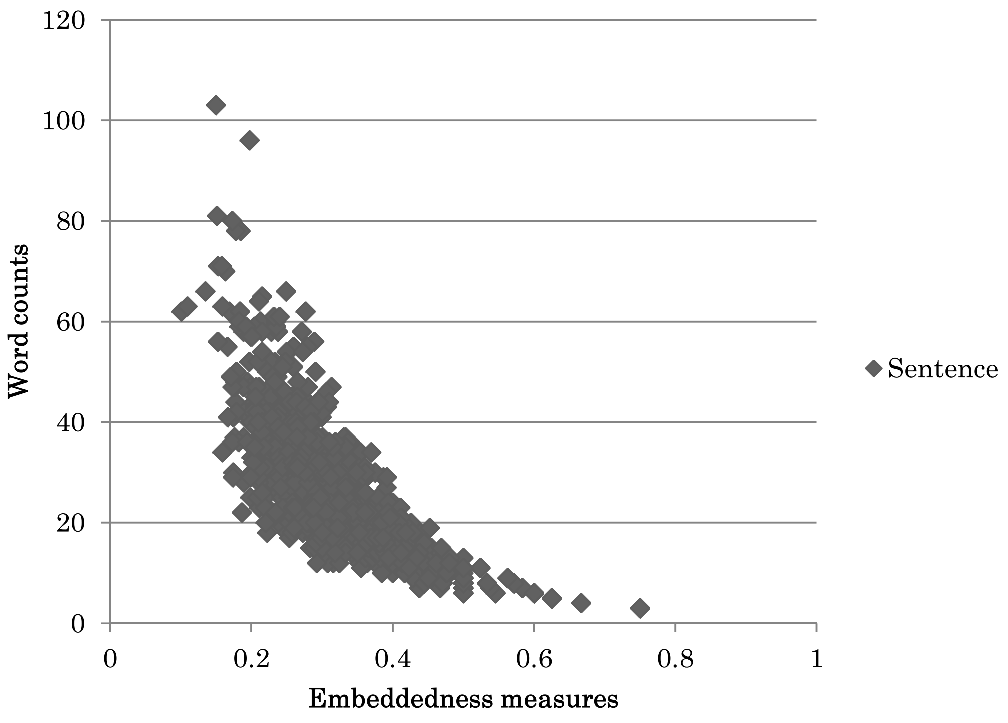


Figure VIII.4. The distribution of sentences in the essay section (n=1072) in terms of their embeddedness measures and word counts.

Ficlets section

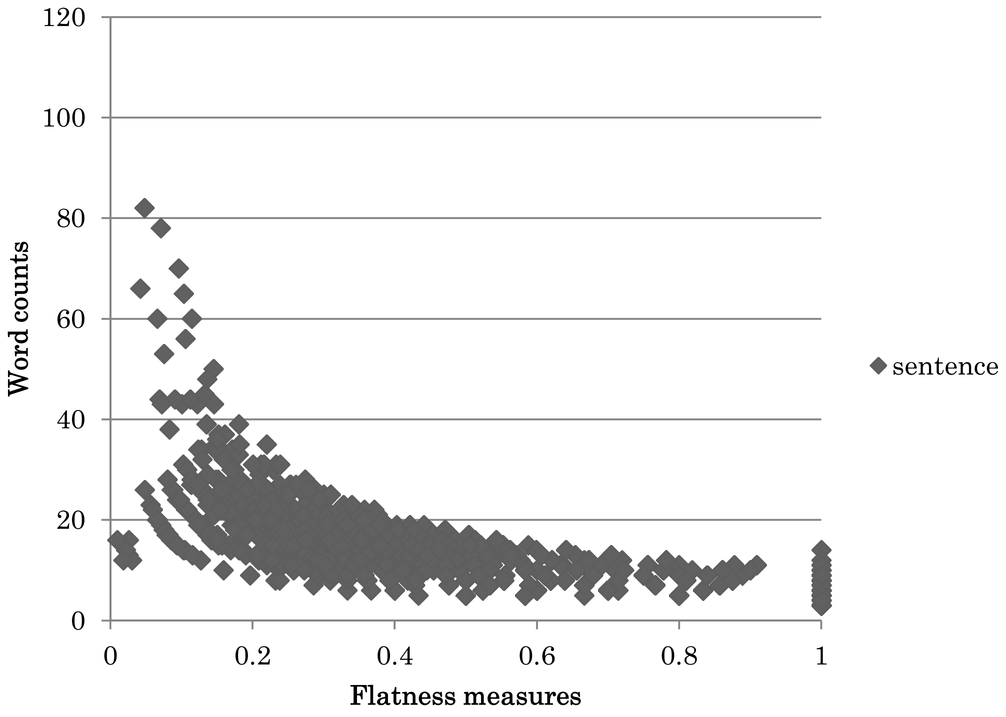


Figure VIII.5. The distribution of sentences in the Ficlet section (n=2645) in terms of their flatness measures and word counts

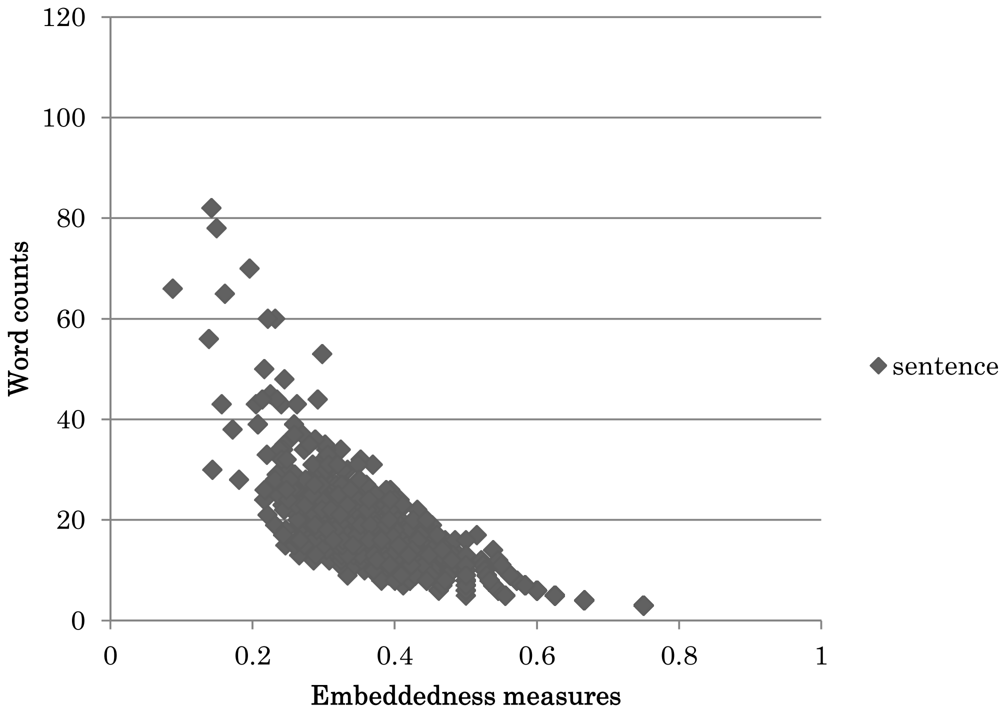


Figure VIII.6. The distribution of sentences in the Ficlet section (n=2645) in terms of their embeddedness measures and word counts

Fiction section

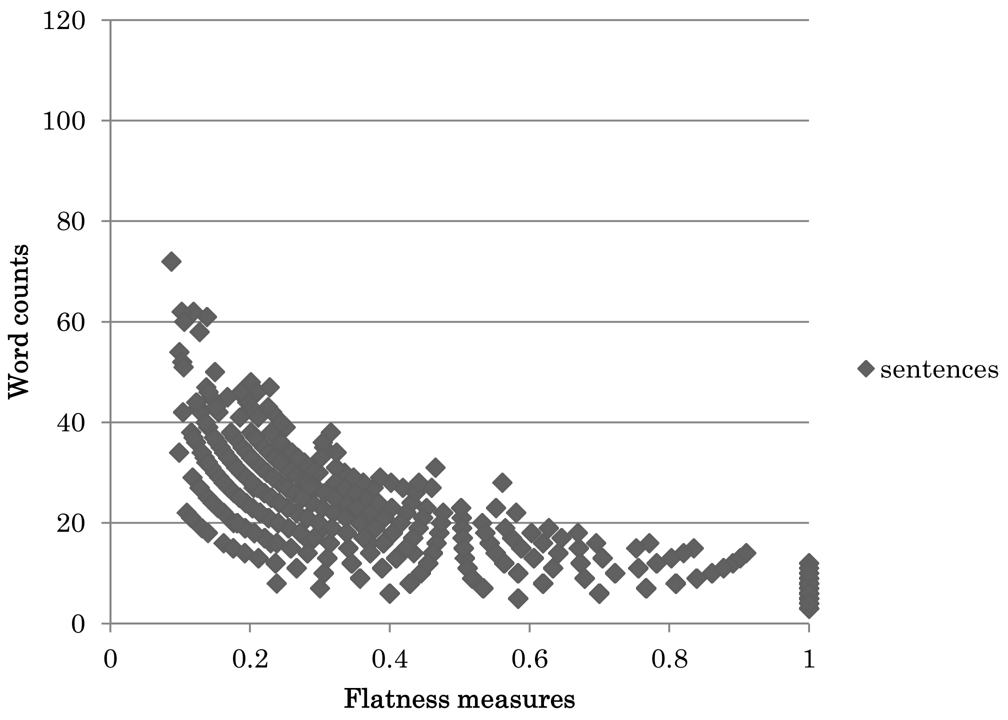


Figure VIII.7. The distribution of sentences in the Fiction section (n=2639) in terms of their flatness measures and word counts

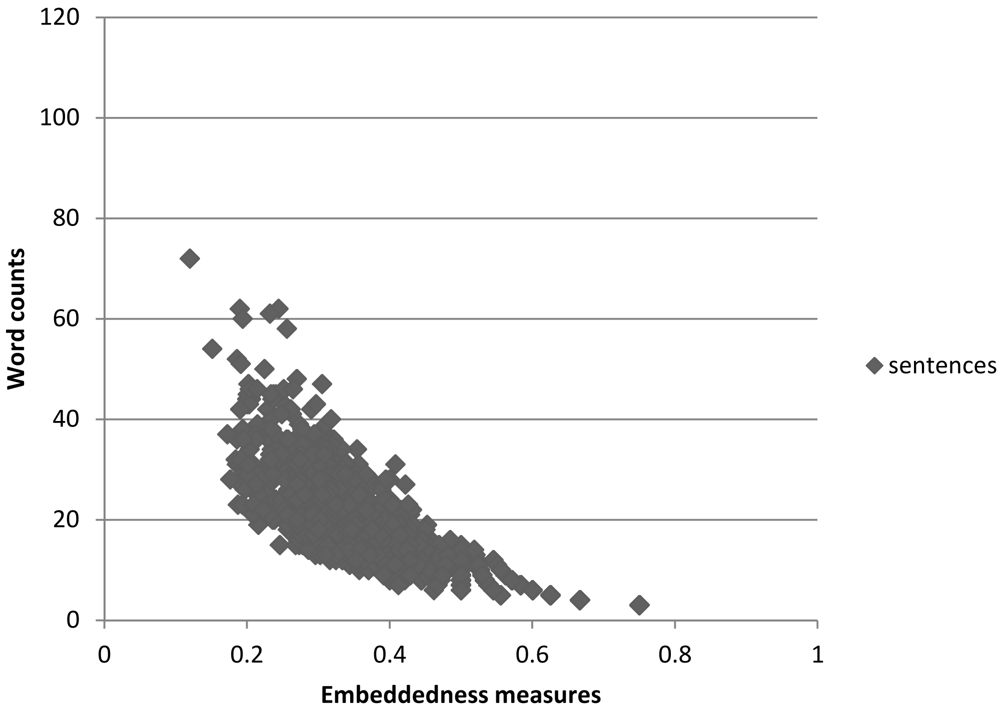


Figure VIII.8. The distribution of sentences in the Fiction section (n=2639) in terms of their embeddedness measures and word counts

Government document section

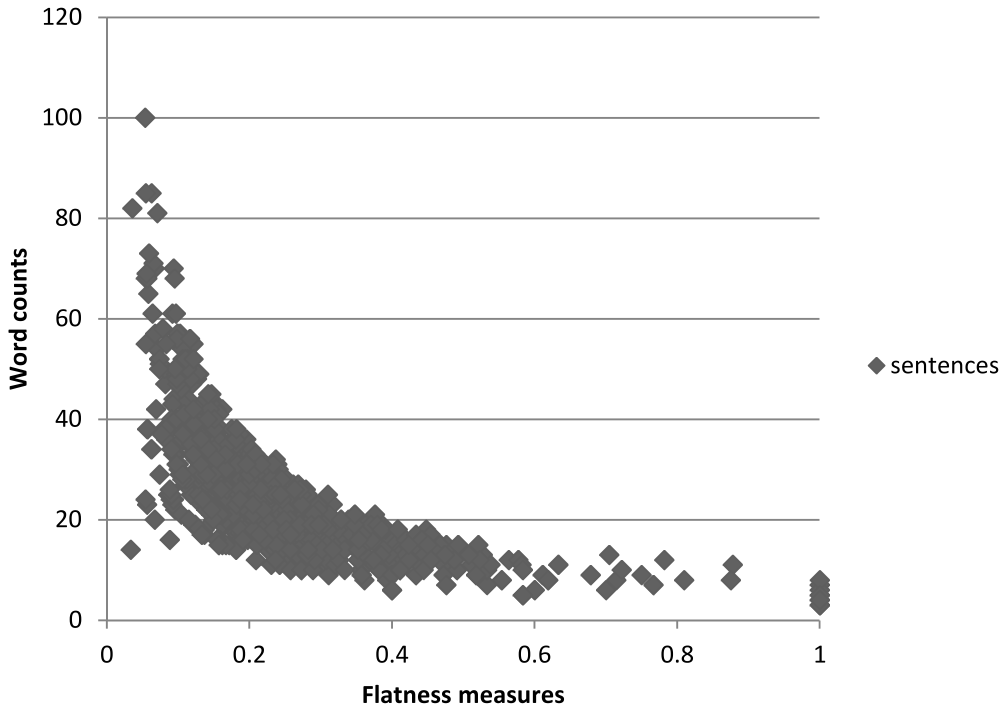


Figure VIII.9. The distribution of sentences in the Government document section (n=1028) in terms of their flatness measures and word counts

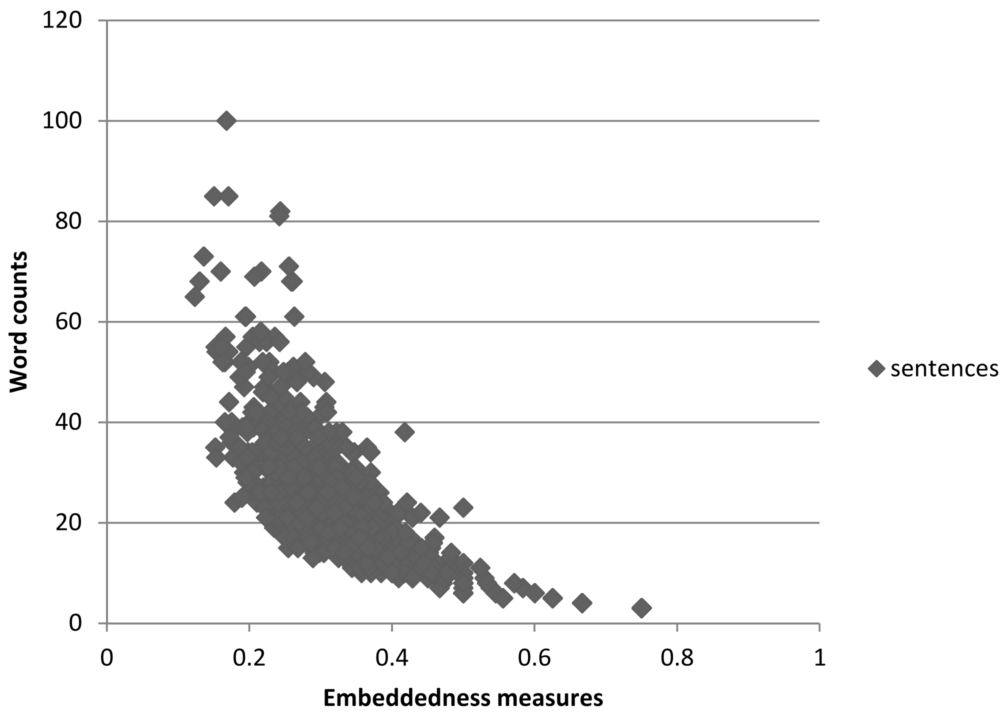


Figure VIII.10. The distribution of sentences in the Government document section (n=1028) in terms of their embeddedness measures and word counts

Joke section

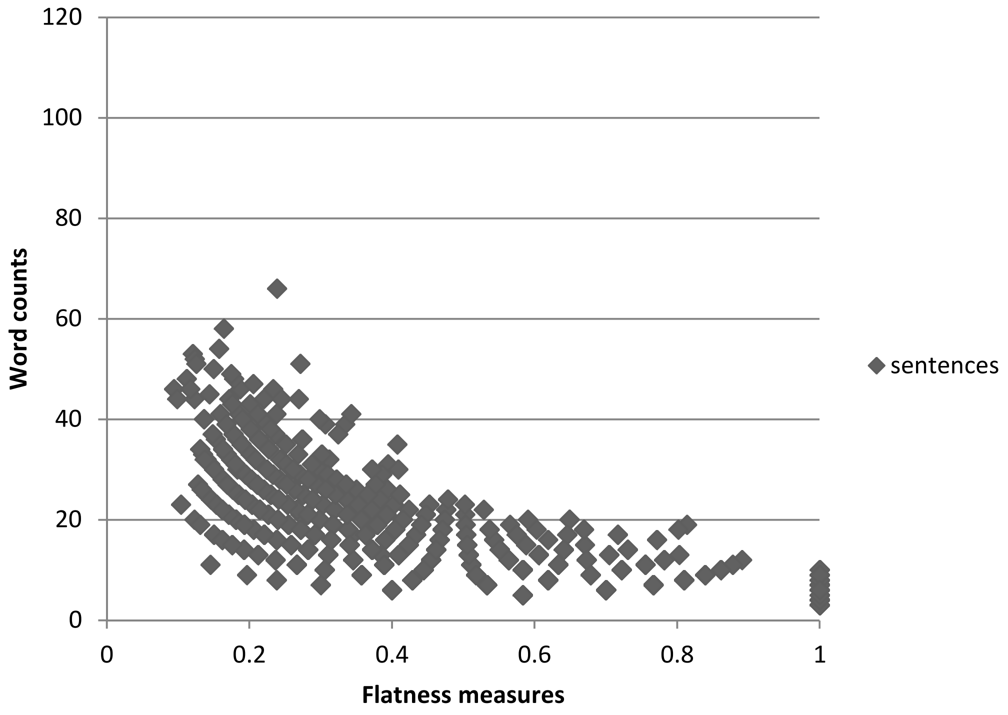


Figure VIII.11. The distribution of sentences in the Joke section (n=2254) in terms of their flatness measures and word counts

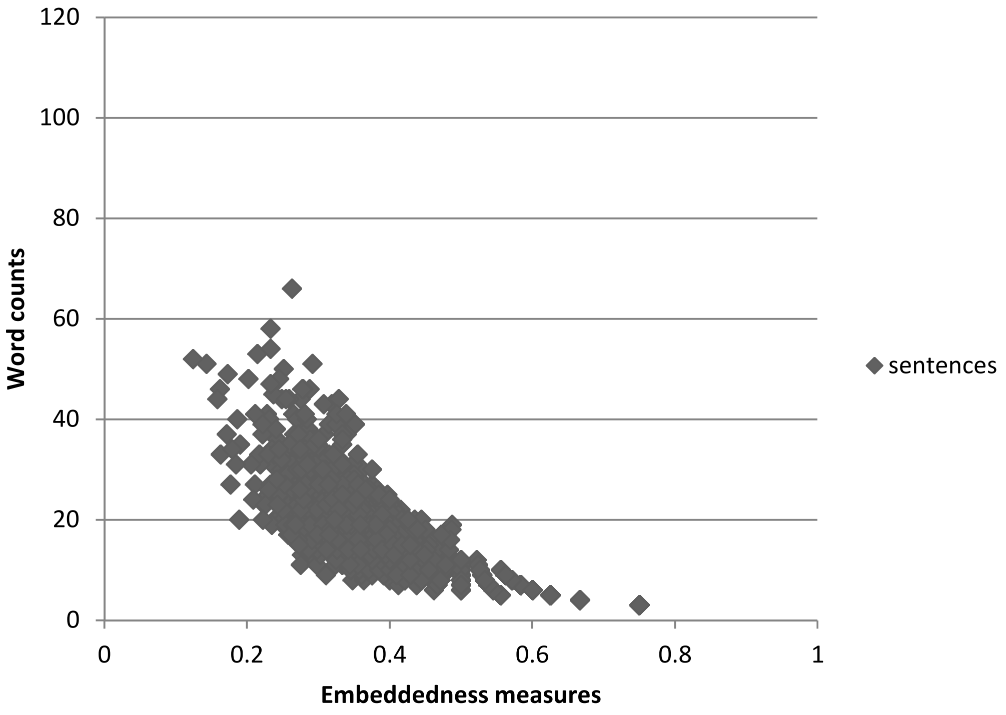


Figure VIII.12. The distribution of sentences in the Joke section (n=2254) in terms of their embeddedness measures and word counts

Journal section

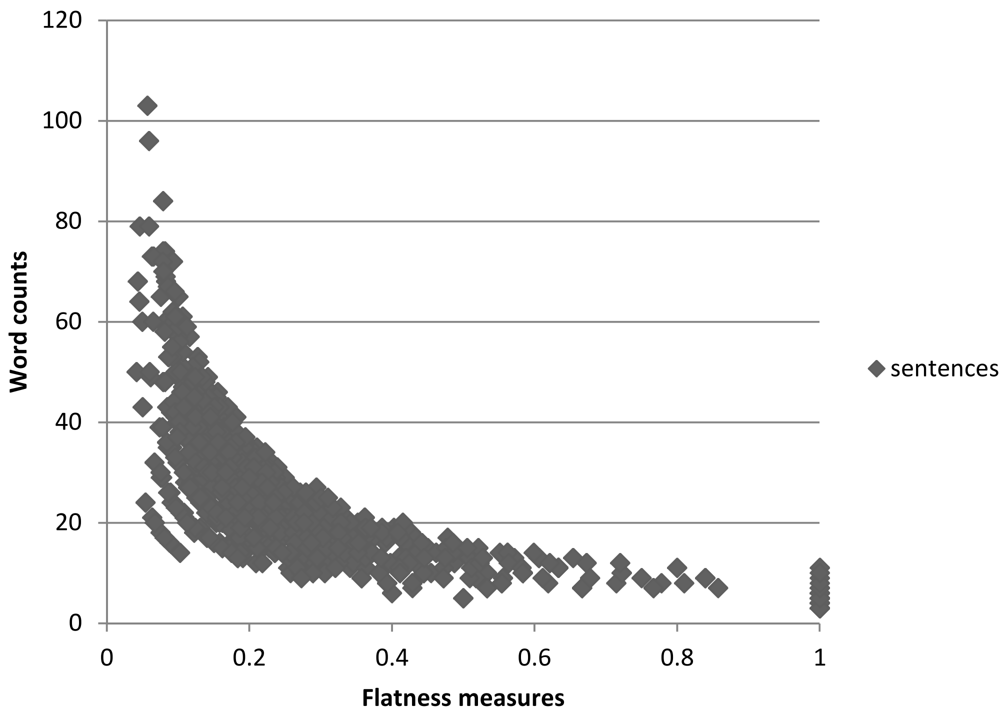


Figure VIII. 13. The distribution of sentences in the Journal section (n=867) in terms of their flatness measures and word counts

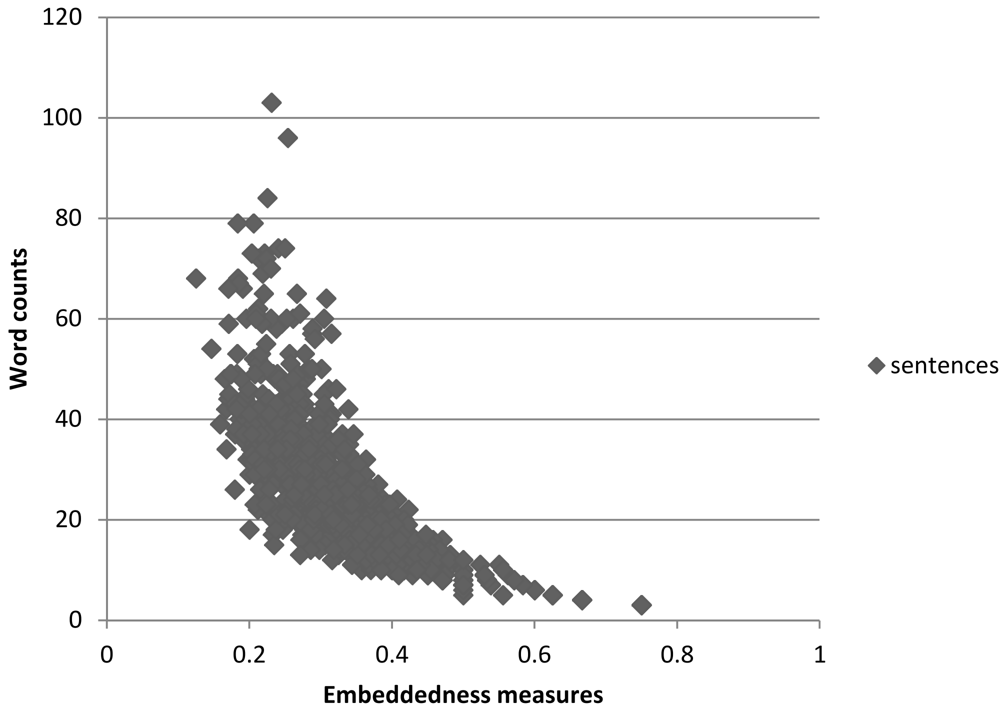


Figure VIII.14. The distribution of sentences in the Journal section (n=867) in terms of their embeddedness measures and word counts

News section

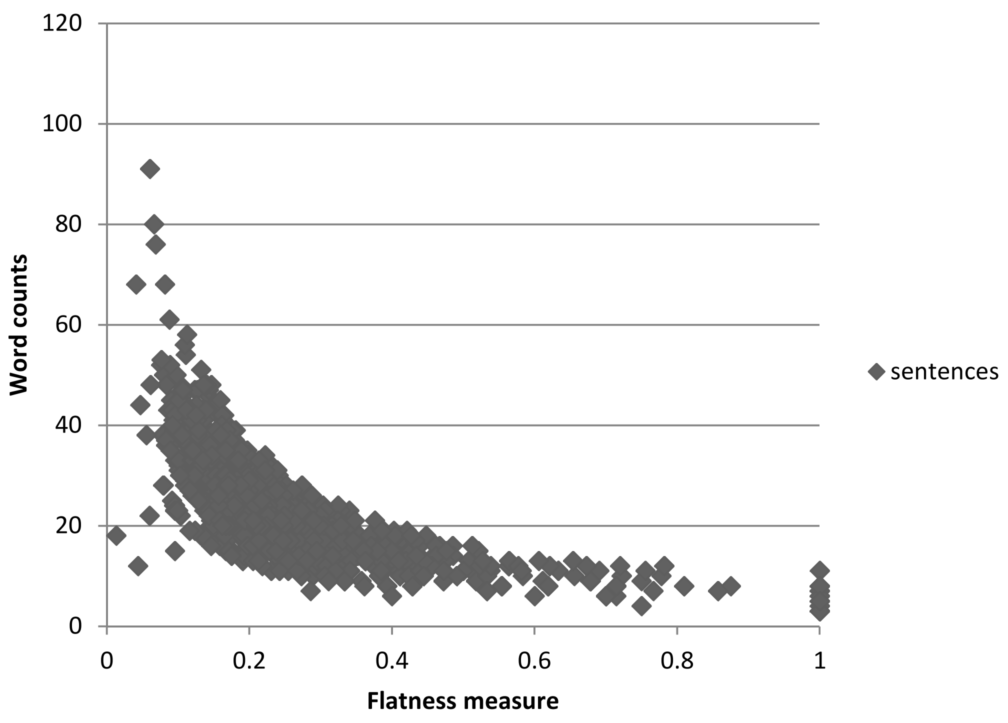


Figure VIII.15. The distribution of sentences in the News section (n=1196) in terms of their flatness measures and word counts

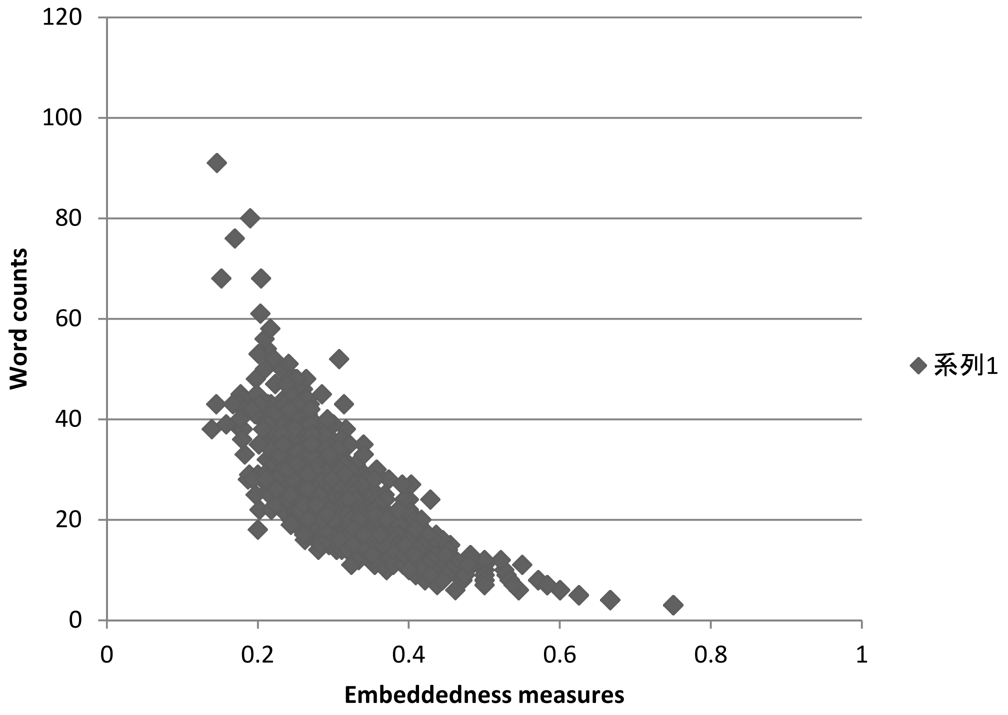


Figure VIII.16. The distribution of sentences in the News section (n=1196) in terms of their embeddedness measures and word counts

Non-fiction section

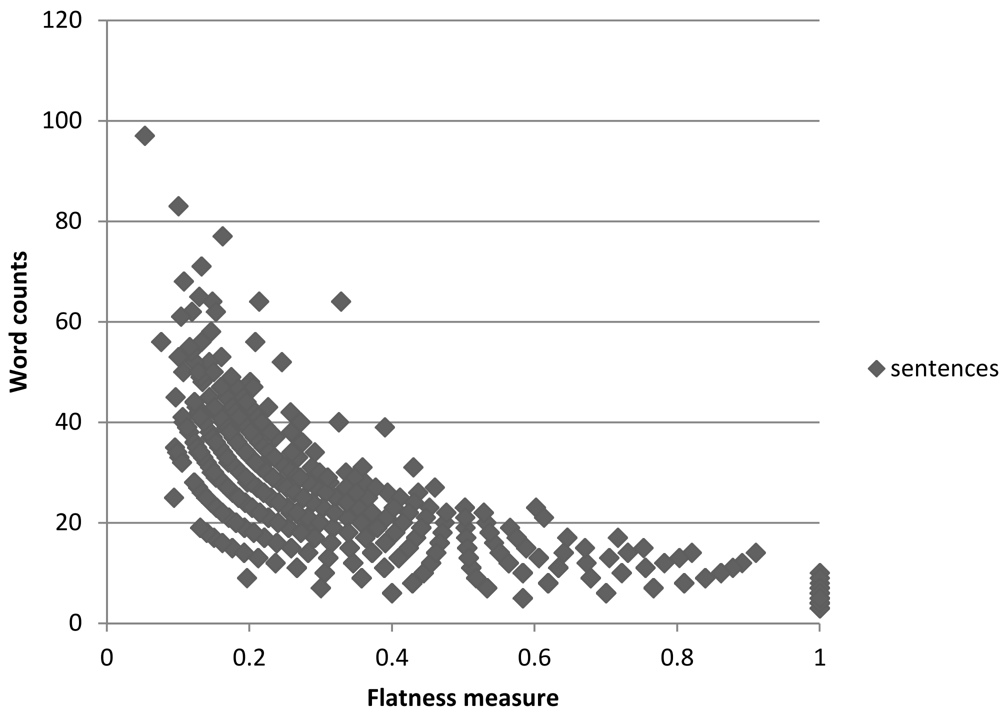


Figure VIII.17. The distribution of sentences in the Non-Fiction section (n=1278) in terms of their flatness measures and word counts

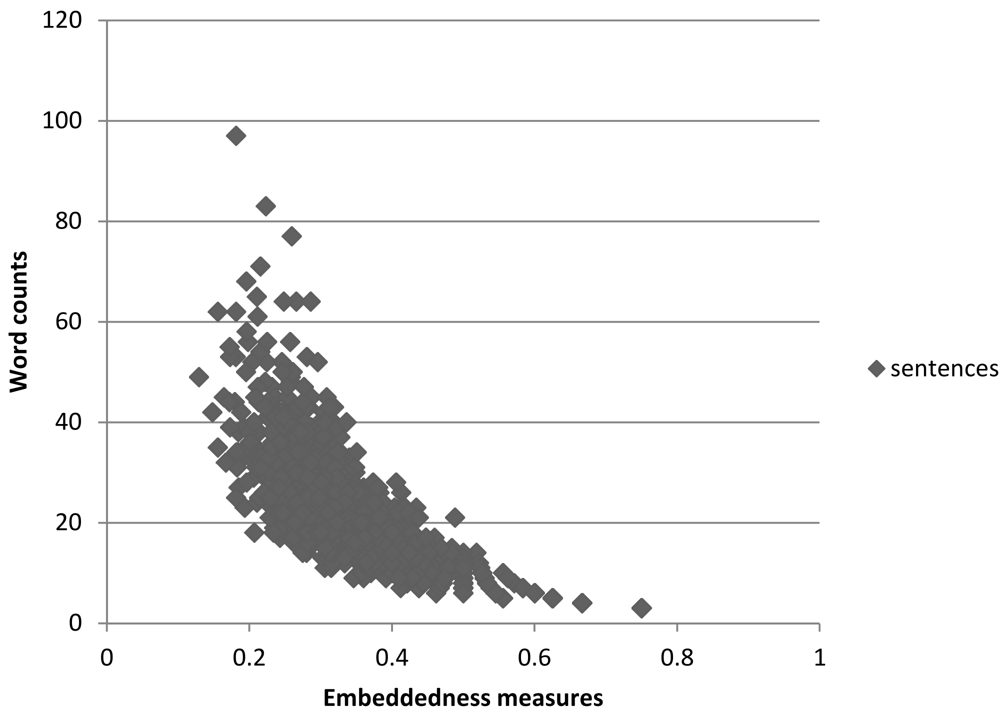


Figure VIII.18. The distribution of sentences in the Non-Fiction section (n=1278) in terms of their embeddedness measures and word counts

Technical report section

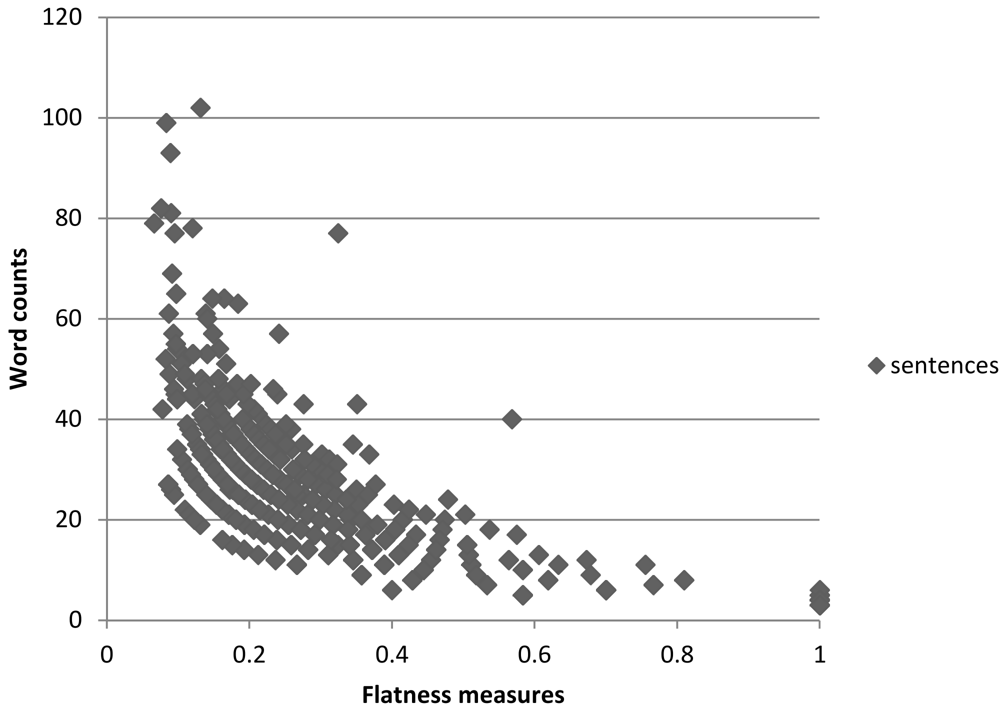


Figure VIII.19. The distribution of sentences in the Technical report section (n=825) in terms of their flatness measures and word counts

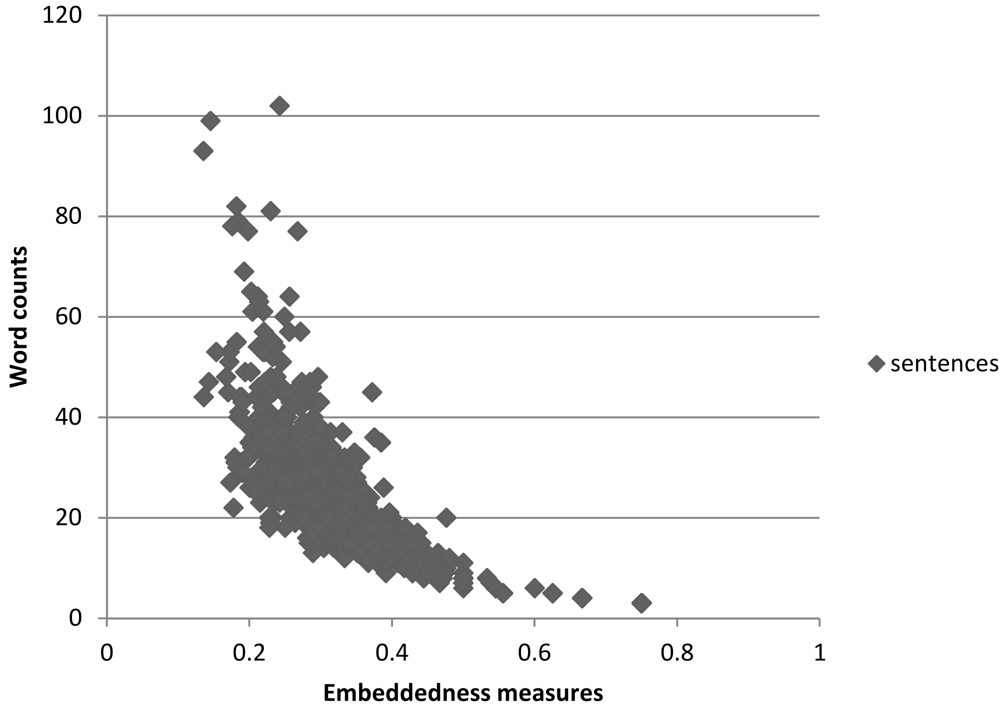


Figure VIII.20. The distribution of sentences in the Technical report section (n=825) in terms of their embeddedness measures and word counts

Travel guide section

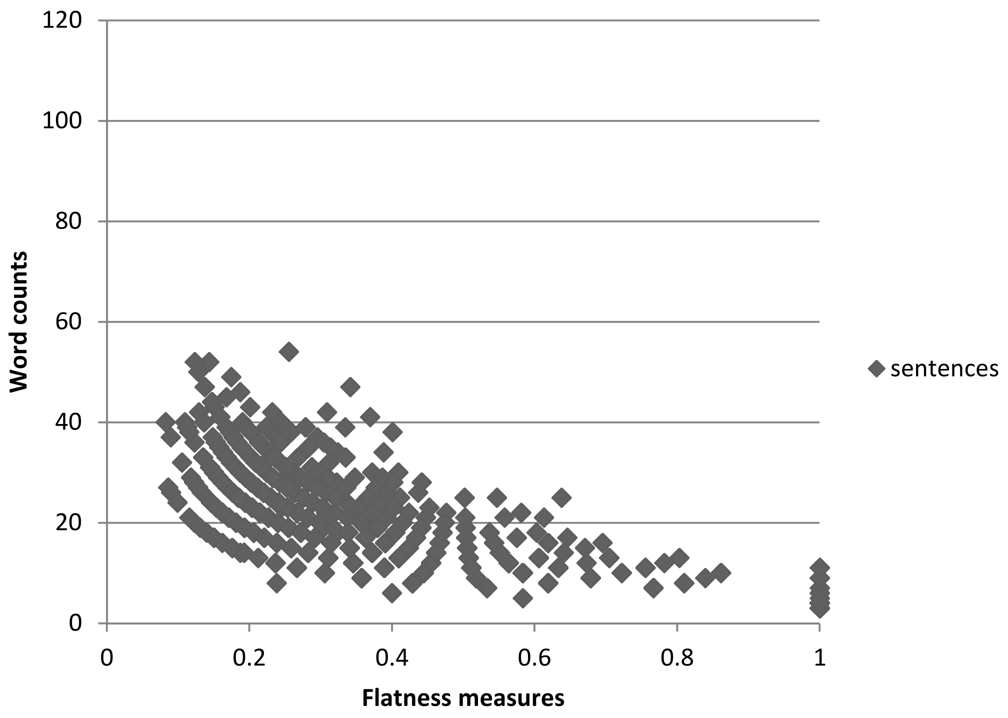


Figure VIII.21. The distribution of sentences in the Travel guide section (n=1196) in terms of their flatness measures and word counts

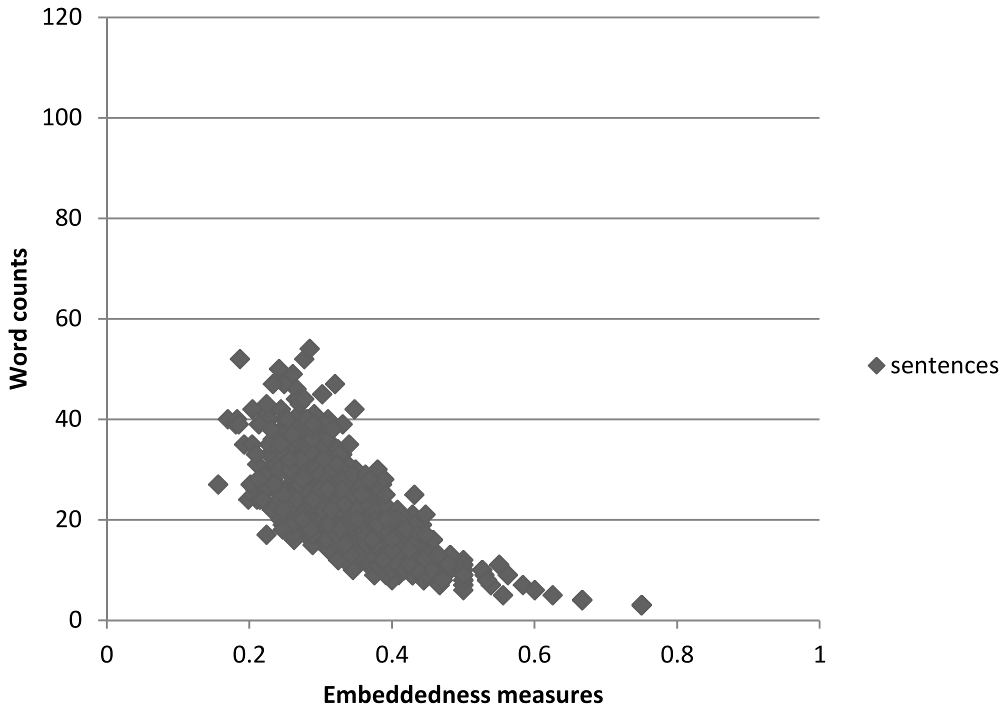


Figure VIII.22. The distribution of sentences in the Travel guide section (n=1196) in terms of their embeddedness measures and word counts