An English Dependency Treebank à la Tesnière

Federico Sangati
University of Amsterdam
f.sangati@uva.nl

Chiara Mazza
University of Pisa
chiara.mazza@gmail.com

Abstract

During the last decade, the Computational Linguistics community has shown an increased interest in Dependency Treebanks. Several groups have developed new annotated corpora using dependency representation, while other people have proposed several automatic conversion algorithms to transform available Phrase Structure (PS) treebanks into Dependency Structure (DS) notation. Such projects typically refer to Tesnière as the father of dependency syntax, but little attempt has been made to explain how the chosen representation relates to the original work. A careful comparison reveals substantial differences: modern DS annotations discard some relevant features characterizing Tesnière's model. This paper is presenting our attempt to go back to the roots of dependency theory, and show how it is possible to transform a PS English treebank to a DS notation that is closer to the one proposed by Tesnière, which we will refer to as TDS. We will show how this representation can incorporate all main advantages of modern DS, while avoiding well known problems concerning the choice of heads, and better representing common linguistic phenomena such as coordination.

1 Introduction

Although the tradition of using syntactic models in linguistics can be dated back to Panini's work (4th century BC), the discussion about which model linguists should use is still open. Corpus-based Computational Linguistics (CL) was born in the middle of last century, an important period of change in linguistic theory. In continental Europe, the French structuralist Lucien Tesnière, was developing a general theory of syntax, which was published posthumously in 1959 [15], and would became the foundation of Dependency Structure (DS) theory. In 1957 Noam Chomsky published his own work on Syntactic Structures [3], which would become the main reference for Phrase Structure (PS) theory.

Corpus-based CL has been strongly influenced by PS notation, due to the strong worldwide position of Chomskyan theory, and still nowadays most linguistic resources conform to this representation. Only in the last decade, more and more interest was placed on dependency theory, thanks to several research groups that have developed new annotated treebanks using DS formalisms (e.g., [1], [6]). In the same period other people have proposed several automatic conversion algorithms to transform available PS treebanks into DS (e.g., [17], [5], [9]). Such projects typically refer to Tesnière as the father of dependency syntax, but little attempt has been made to explain how the chosen dependency representation relates to the original work. A careful comparison reveals substantial differences: modern DS retains only the main idea proposed by Tesnière, namely the relation of dependency between words (section 2.1), while other operations and features of the original model are discarded or not overtly represented.

In the current paper we present an ongoing project of converting the English Penn Wall Street Journal (WSJ) Treebank [11] into a DS notation that is closer to the one proposed by Tesnière, which we will refer to as TDS. In particular we will reintroduce three key concepts: the division of a sentence into *blocks* of words, which act as intermediate linguistic units (section 2.2), the *junction* operation, to handle coordination and other types of conjoined structures (section 2.3), and the operation of *transference*, to generalize over the categories of the linguistic elements (section 2.4)[1].

Our work is not purely driven by historical concerns; in section 3 we will in fact give empirical evidence that shows how this representation can incorporate all main advantages of modern DS, while avoiding well known problems concerning the choice of heads, and better representing common linguistic phenomena such as coordination. Finally, in section 4 we will give more details about the conversion procedure, and the generated structures.

2 Dependency Structures à la Tesnière

2.1 The dependency relation

The main idea behind Tesnière's model is the notion of *dependency*, which identifies the syntactic relation existing between two elements within a sentence, one of them taking the role of governor (or head) and the other of dependent (*régissant* and *subordonné* in the original terminology). Tesnière schematizes this syntactic relation using a TDS *stemma* as in figure 1, putting the governors above the dependents. On the right side of the figure we present the same sentence using our notation, incorporating all the main features introduced by Tesnière, which we will explain in the following sections.

[1] See in [15] part I ch. 22 on *nucléus*, part II on *jonction*, and part III on *translation*. We choose *transference* for the original French word *translation* to avoid any misunderstanding with the other meaning of the word *translation* in English. Unfortunately, 50 years after its publication, there is still no official English translation of the author's work.

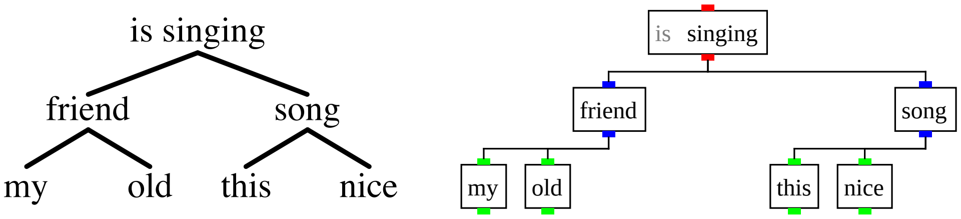

Figure 1: TDS of the sentence "*My old friend is singing this nice song*", in Tesnière notation (left) and in our TDS representation (right).

2.2 Words, blocks and categories

In TDS, all words are divided into two classes: full content words (e.g., nouns, verbs, adjectives, etc.), and empty functional words (e.g., determiners, prepositions, etc.). Each full word forms a *block*[2] which may additionally include one or more empty words, and it is on blocks that operations are applied. Tesnière distinguishes four *block categories* (or functional labels[3]), here listed together with the original single letter notation, and the color reported in our diagrams: *nouns* (O, blue), *adjectives* (A, green), *verbs* (I, red), and *adverbs* (E, yellow).

The verb represents the process expressed by the clause, and all its arguments, representing the participants in the process, have the functional labels of nouns, and are determined by the *valence* of the verb. On the other hand the verb's adjuncts (or circonstants), representing the circumstances under which the process is taking place, i.e., time, manner, location, etc., have the functional labels of adverbs. We will now introduces two operations, *junction* and *transference*, by means of which it is possible to construct more complex clauses from simple ones.

2.3 Junction

The first operation is the *junction*. It is employed to group blocks which lie at the same level, the *conjuncts*, into a unique entity which has the status of a block. The conjuncts are horizontally connected in the TDS, belong to the same category, and are possibly (and not always) connected by means of empty words, the *conjunctions*. Figure 2 displays three coordinated structures[4].

[2] Tesnière in [15, ch. 22] uses the term *nucléus*, and explains that it is important in order to define, among other things, the *transference* operation (see section 2.4). In our diagrams blocks are represented as black boxes, and empty words are written in grey to distinguish them from full words.

[3] We will use both terms interchangeably. Categories can be roughly seen as a simplification of both PoS tags and dependency relations in DS's. See also section 3.2.

[4] Tesnière uses the junction operation to represent coordinated structures and other particular joined structures, such as the apposition (e.g., *[the US president], [Obama]*). Although our implementation includes all types of junction, in this paper we will particularly focus on *coordination* since it constitutes the majority of the cases, and yet the most problematic ones.

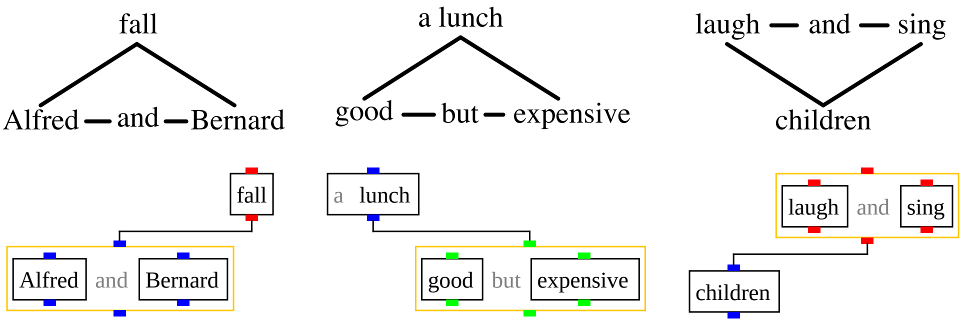

Figure 2: Examples of coordination. Tesnière's original notation is on top, and our notation at the bottom (we represent the junction with a yellow box).

2.4 Transference

The other operation is named *transference*. There are two types of transference. The first degree transference is a shifting process which makes a block change from the original category of the content word, to another[5]. This process often occurs by means of one or more empty words belonging to the same block, called *transferrers*. Figure 3 (left) shows an example of first degree transference. The word *Peter* is transferred from the word class *noun* and takes the functional label of an *adjective* via the possessive clitic *'s* which acts as a transferrer. In our representation (bottom), every block has two little colored boxes: the one at the bottom indicates the original category of the content word, and the one at the top indicates the category of the block after all transferences are applied.

The second degree transference occurs when a simple clause becomes an argument or an adjunct of another clause[6], maintaining all its previous lower connections, but changing its functional label within the main clause. The sentences below represent some examples of second degree transference:

(1) She believes *that he knows*

(2) The man *I saw yesterday* is here today

(3) You will see him *when he comes*

[5] A somehow similar operation has been formulated in the framework of Combinatory Categorial Grammar (cf. [8]), and goes under the name of type raising.

[6] In other words, the verb of the embedded clause becomes a dependent of an other verb. This should not be confused with the case of compound verbs, which are represented as a single block, where auxiliaries are labeled as empty words (see for instance the TDS in figure 1).

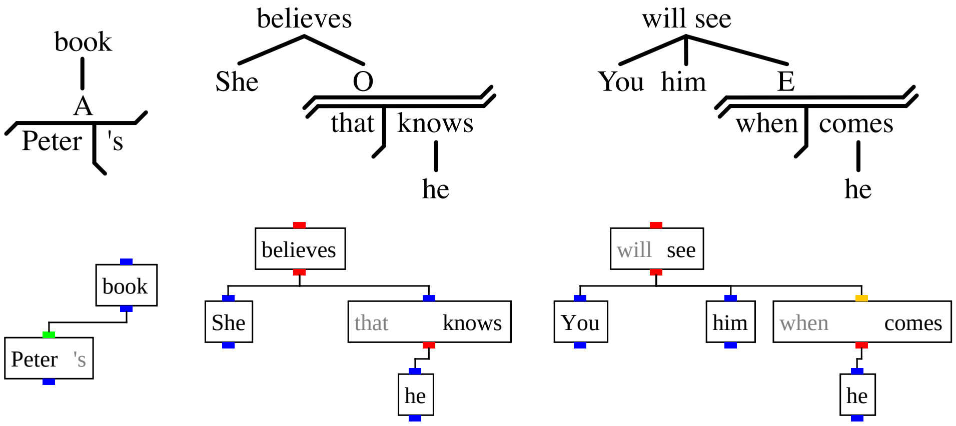

Figure 3: An example of *first degree transference* of the phrase "*Peter's book*" (left), and two examples of *second degree transference* of the sentence "*She believes that he knows*" (center) and the sentence "*You will see him when he comes*" (right).

In the first sentence, we have a transference verb≫ noun by means of the transferrer *that*. The embedded clause in italics takes the functional label of a noun, and becomes the object of the verb. Figure 3 (center) shows the corresponding TDS. The embedded clause in the second example has the functional label of an adjective. It is a transference verb≫ adjective without any transferrer. The third sentence is an example of transference verb≫ adverb: the clause in italics has the functional label of a temporal adverb through the transferrer *when*. Figure 3 (right) shows the corresponding TDS.

3 Advantages of TDS over DS

In this section we will describe three main advantages of using TDS notation as an alternative of currently available DS representations. In particular we will discuss the issue of choosing the linguistic heads in PS trees (section 3.1), compare how the two models categorize dependency relations (section 3.2), and how they treat the phenomenon of coordination (section 3.3).

In order to compare the different representations, Figure 4 illustrates three structures of an English sentence: the original Penn WSJ PS tree, the same structure converted to DS as in [9], and the TDS our conversion algorithm generates.

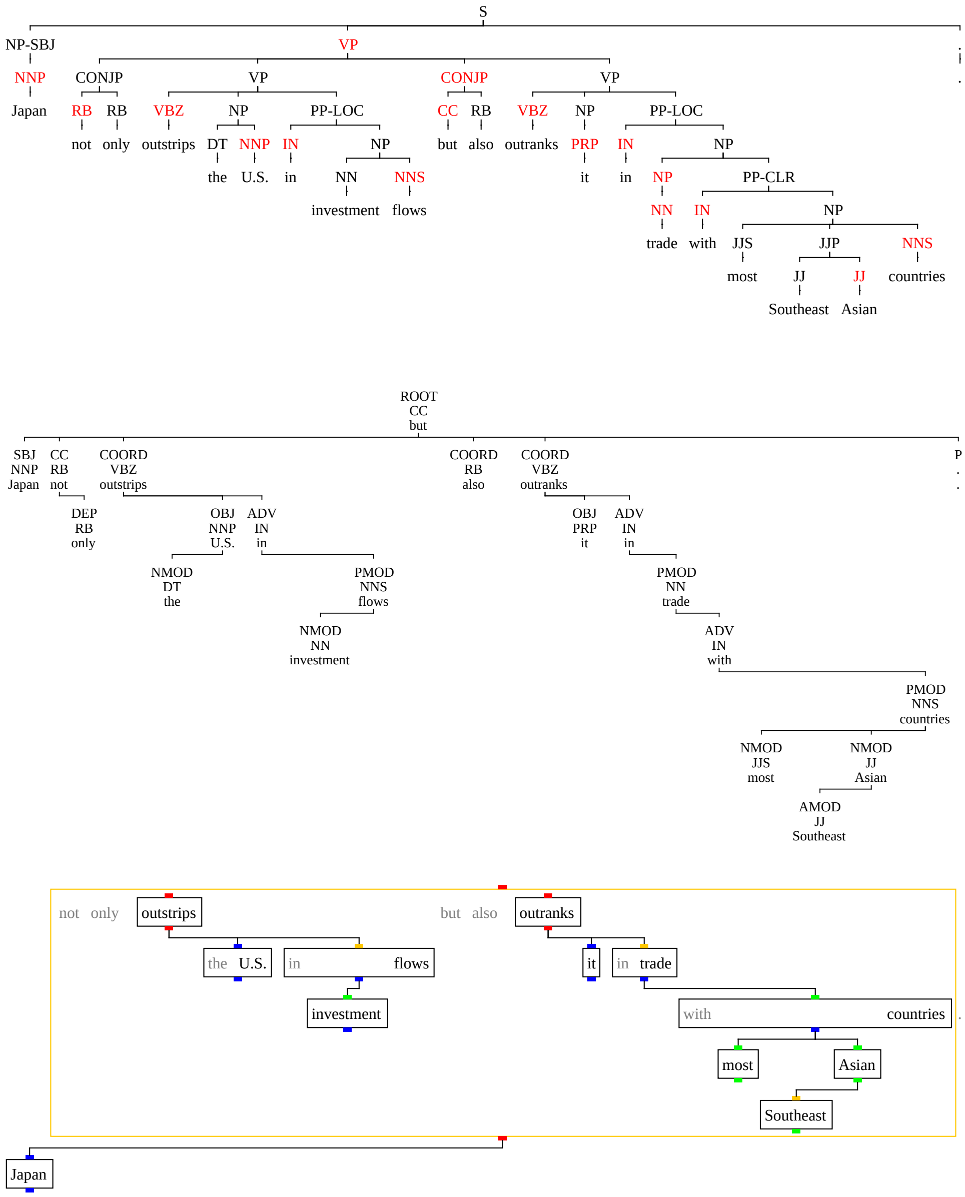

Figure 4: Comparison between different representations of an English sentence. **Top**: original WSJ PS taken from the WSJ sec-00 (#666). Null productions and traces have been removed. The red labels are the heads according to the DS below. **Center**: DS according to [9] using the `pennconverter` script in `conll2007` mode. Every word in the DS is presented together with its PoS and the label of the dependency relation with its governor. **Bottom**: TDS our conversion algorithm generates.

3.1 Choosing the correct heads

The first step in order to perform a PS-to-DS conversion is to annotate the starting PS with head labels. This procedure has been initially proposed in Natural Language Processing (NLP) by Magerman [10] and then slightly modified by others (e.g., [17], [9]). If exactly one unique head is chosen for every constituent of the PS, the enriched tree can be shown to be homomorphic to a single projective DS (cf. [7], [13]).

The choice of heads assignment is a critical one: although much linguistic literature is present on this issue (cf. [4]), in NLP there have been only few attempts to empirically evaluate different heads assignments (i.e., [2], [14]). While certain choices are less disputed (e.g., the verb is unequivocally the head of simple clauses), most of the remaining ones are contended between empty and full words. The most frequent cases are listed here:

- Determiner vs. noun in nominal phrases (e.g., *the man*).
- Preposition vs. noun in prepositional phrases (e.g., *on paper*).
- Complementizer vs. verb in sub-clauses (e.g., *I believe that it is raining*).

In TDS, all these choices become irrelevant: since every empty word is included in the block together with the content word it belongs to, no preference is needed[7].

3.2 Categories and Blocks

Currently used DS representations make use of labels to identify the dependencies between words. For example *SBJ* and *OBJ* are used to mark the relation between a verb and its subject and direct object respectively. These labels are closely related to the four categories proposed by Tesnière. The main difference is in their number: while DS uses around a dozen of different labels, TDS uses only four. This turns out to be beneficial for a more simplified and yet generalized analysis.

The other difference is more subtle. In DS every word is a node, and therefore, for every node (except for the root) we need to identify the label of the dependency relation with its governor. The problem here is related to the above discussion about the choice of heads. If we take the example in figure 3 (center), one has to choose whether the complementizer or the verb is the direct object of the main verb. TDS better represents these cases, by including both elements in the same block. This choice is justified by the fact that both elements contribute to make the node an argument or an adjunct of the verb.

[7] In TDS, heads assignment remains essential when two or more full words are sister nodes of the same constituent, such as in “the woman who I like”. In this example the verb should be the head.

3.3 Coordination

Coordination represents one of the major problems in currently used DS representations (cf. [13]). If dependency[8] is the only operation available to relate words, two main strategies are adopted:

1. One conjunction (or conjunct) is the head of the other elements.

2. Each element (conjunction or conjunct) is the head of the adjacent element which follows.

The first solution is the one which is more commonly adopted in current PS-to-DS conversions. The second one is proposed by Mel'čuk in [12]. Both solutions are problematic in circumstances such as the one of figure 4. If the coordination includes multiple conjunctions, assigning the head to either one of the conjuncts or one of the conjunctions, leads to a strong asymmetry in the structure: either the conjuncts are not all at the same level, or the set of dependents includes both conjunctions and conjuncts. Moreover, if the coordination phrase is coordinating verbs at the top of the sentence structure, other potential blocks, e.g., the subject *Japan* in the example, will also appear in the set of dependents, at the same level with the verbs they depend on[9]. Finally the *conjunction phrase*, i.e., a group of words forming a single conjunction (e.g., *not only* in the example), is also poorly represented in DS representations, since it is not grouped into a unique entity.

Tesnière's choice of adding a special operation to handle coordination is justified if we consider how well it represents all the cases DS fails to represent consistently. Coordination in TDS can be seen as a borrowing of the notion of constituency from PS notation: the different blocks being conjoined have equal status, they govern all the blocks being dominated by the coordination block, and are dependents on all blocks the coordination structure depends on.

4 Converting the Penn WSJ in TDS notation

In this section we will present the current state of the project of converting the Penn WSJ treebank [11] into TDS notation. In section 4.1 we will introduce the elements composing each generated TDS, in section 4.2 we will describe the conversion procedure, and in section 4.3 we will provide some error analysis on the generated structures.

[8] We only consider the case of single headed DS, i.e., each word should have exactly one governor.

[9] The labels of the dependency relations, such as the ones in the DS of figure 4, can often help to differentiate between dependents which have the same head, but differ in their functional labels. However they cannot be considered an optimal solution, since they don't eliminate the structural asymmetry.

4.1 Elements of a TDS

Figure 5 illustrates the main elements, introduced in section 2, which we need to define in order to construct our TDS's. Words are divided into full and empty words[10], and blocks are either standard or junction blocks. A generic block contains a list of empty words, and a list of dependent blocks. In addition a standard block has to contain a unique full word, while a junction block needs to specify a list of conjunction words and a list of conjunct blocks.

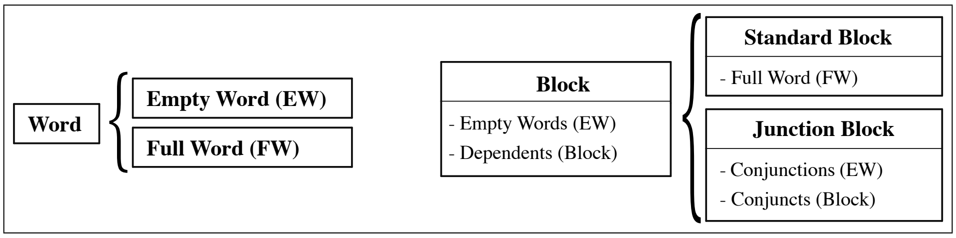

Figure 5: Word and block types.

4.2 The conversion procedure

In order to generate TDS's from the Penn WSJ, we have decided to start from the PS original annotation, instead of using already converted DS's. The main reason for this choice is that PS annotation of the WSJ is richer than currently available DS representations. This concerns in particular coordination structures, which would be much harder to reconstruct from DS notation (see section 3.3).

Each PS in the corpus is preprocessed using the procedure described in [16], in order to add a more refined bracketing structure to noun and adjectival phrases. Moreover, we remove null productions and traces from all trees, and enrich them with head labels[11].

The pseudocode reported in algorithm 1, contains the procedure which is applied to each PS of the corpus, in order to generate the respective TDS. The algorithm recursively traverses from top to bottom each node of a PS, and outputs either a junction block[12] (left) or a standard block (right).

[10] A word is empty if its PoS is one of the following: punctuation marks, CC, DT, EX, IN, MD, POS, RP, SYM, TO, WDT, WRB. Moreover special pairs of words are marked as empty (e.g., more like, more than, even though, such as, many of, most of, rather than).

[11] This operation is done using the `treep` script [2], and a rule set which is a modification of the one used in [10]. This modification mainly consists in prioritizing conjunctions' PoS within several phrase structures (e.g., NP, VP, QP). Moreover we have added few rules for the non-terminals (i.e., "NML", "JJP") introduced by the procedure described in [16].

[12] A constituent is roughly identified as a junction structure when it presents conjunctions elements (i.e., CC, CONJP), or when it is composed of subconstituents with the same labels, such as in the cases of apposition (see note 4).

```
Algorithm: $Convert(N_{PS})$
Input: A node $N_{PS}$ of a PS tree
Output: A block $N_{TDS}$ of a TDS tree
begin
    instantiate $N_{TDS}$ as a generic block
    if $N_{PS}$ is a junction then
        instantiate $N_{TDS}$ as a junction block
        foreach node $D$ in children of $N_{PS}$ do
            if $D$ is a conjunct then
                $D_{TDS} \leftarrow Convert(D)$
                add $D_{TDS}$ as a conjunct block in $N_{TDS}$
            else
                $D_{lex} \leftarrow$ lexical yield of $D$
                if $D_{lex}$ is a conjunction then
                    add $D_{lex}$ as a conjunction in $N_{TDS}$
                else
                    add $D_{lex}$ as empty word(s) in $N_{TDS}$
    else
        $N_h \leftarrow$ head daughter node of $N_{PS}$
        if $N_h$ yield only one word $w_h$ then instantiate $N_{TDS}$
        as a standard block with $w_h$ as its full word
        else $N_{TDS} \leftarrow Convert(N_h)$
        foreach node $D$ in children of $N_{PS}$ do
            if $D == N_h$ then continue
            $D_{lex} \leftarrow$ lexical yield of $D$
            if $D_{lex}$ are only empty words then
                add $D_{lex}$ as empty word(s) in $N_{TDS}$
            else
                $D_{TDS} \leftarrow Convert(D)$
                add $D_{TDS}$ as a dependent of $N_{TDS}$
    return $N_{TDS}$
end
```

Algorithm 1: Pseudocode of the conversion algorithm from PS to TDS.

For each TDS the algorithm generates, several post-processing steps are applied:

1. Join together all *compound verbs* into a unique block[13].

2. Unify in a unique standard block all *contiguous proper nouns*.

3. Define the *original category*[14] of each block.

4. Define the *derived category*[15] after transferences are applied.

The conversion procedure just described has been successfully employed to generate the first TDS version of the Penn WSJ treebank. The conversion and visualization tool, together with its technical documentation, is freely available at `http://staff.science.uva.nl/~fsangati/TDS`.

[13] E.g., [is eating], [has been running]. All verbs preceding the main one, are marked as empty words. This procedure doesn't apply to junction structures, see also section 4.3.

[14] This category is specified by the PoS of its full word if it is a standard block, and by the original category of the first conjunct block, if it is a junction structure.

[15] This category is specified by the original category of the governing block (if the current block is the root of the structure the category coincides with its original category). If the governing block is a noun or an adjective, the current block is an adjective or an adverb, respectively. If the governing block is a verb, the current block is either a noun or an adverb. This last decision depends on whether the original PS node, from which the current block derives, has a circumstantial label, i.e., it contains one of the following tags: ADVP, PP, PRN, RB, RBR, RBS, ADV, BNF, CLR, DIR, EXT, LOC, MNR, PRP, TMP, VOC.

4.3 Error analysis

At this stage it is impossible to give a meaningful quantitative analysis of the overall accuracy of the conversion, since no gold corpus annotation is available for the target format. However, a manual analysis on a small sample of the corpus reveals that most of the mistakes relate to coordinated structures[16], and wrongly assigned categories (mostly arguments/adjuncts).

Moreover, in a limited number of cases, we found two possible inadequacies of the current TDS notation, when dealing with specific linguistic phenomena. The first issue concerns junction structures: while in our model the junction operation can only join blocks, several linguistic constructions show the necessity of defining it also on full (or empty) words within a block[17]. Two examples are the coordination of compound verbs (e.g., *He was [eating and drinking]*), and the coordination of empty words (e.g., *The indicator fell steadily [up to and through] the crash*).

The second case regards the transferrers: in our implementation they must be always empty words. There are however few cases in which a full words can function as a transferrer, such as *likely* in "*A forum likely to bring attention*". We will take into consideration these modifications for future updates of the model.

5 Conclusion

In this paper we have described an ongoing project of converting the Penn Wall Street Journal treebank from PS to TDS representation, inspired by the work of Tesnière [15]. Corpus-based Computational Linguistics has often valued a good compromise between adequacy and simplicity in the choice of linguistic representation. The transition from PS to DS notation has been seen as a useful simplification, but many people have argued against its adequacy in representing frequent linguistic phenomena such as coordination. The TDS conversion presented in this paper, reintroduces several key features from Tesnière's work: on one hand the operation of junction enriches the model with a more adequate system to handle conjoined structures (e.g., coordination); on the other, the blocks, the transference operation, and the category system further simplify and generalize the model.

We are currently working on a probabilistic extension of our framework. The idea is to define a language model which generates and parses sentences using the new representation. In particular, for what concerns junction structures, our intuition is that the model should generate them differently with respect to standard blocks. If our intuition is correct, the new probabilistic model could be better at modeling and predicting language structures.

[16] In certain complex coordinated structures, where conjuncts and modifiers of the coordination are put at the same level, dependent blocks are wrongly identified as conjuncts. In other cases the coordination is not detected because conjunction words are missing.

[17] This means that we would need to define an other element, in addition to the word-types of figure 5 (left), which we could call *junction word*. This new entity would have to specify a list of full (or empty) *conjunct words* and a list of (empty) *conjunction words*.

Acknowledgments

The authors are particularly thankful to Igor Mel’čuk, Willem Zuidema, Rens Bod, Yoav Seginer, Sophie Arnoult, and three anonymous reviewers for their valuable comments. FS gratefully acknowledge funding by the NWO (grant 277.70.006).

References

[1] Cristina Bosco, Vincenzo Lombardo, Daniela Vassallo, and Leonardo Lesmo. Building a Treebank for Italian: a Data-driven Annotation Schema. In *Proceedings of the Second International Conference on Language Resources and Evaluation*, pages 99–105, 2000.

[2] David Chiang and Daniel M. Bikel. Recovering latent information in treebanks. In *Proceedings of the 19th international conf. on Comput. Linguist.* , pages 1–7, Morristown, NJ, USA, 2002.

[3] Noam Chomsky. *Syntactic structures*. Mouton, Den Haag, 1957.

[4] Greville G. Corbett, Norman M. Fraser, and Scott McGlashan, editors. *Heads in Grammatical Theory*. Cambridge University Press, New York, 2006.

[5] Martin Forst, Núria Bertomeu, Berthold Crysmann, Frederik Fouvry, Silvia Hansen-Schirra, Valia Kordoni. Towards a dependency-based gold standard for German parsers - The TiGer Dependency Bank, 2004.

[6] Jan Hajič, Eva Hajičová, Petr Pajas, Jarmila Panevová, Petr Sgall, and Barbora Vidová Hladká. Prague Dependency Treebank 1.0, 2001.

[7] David G. Hays. Grouping and dependency theory. In *National Symposium on Machine Translation*, pages 258–266, Englewood Cliffs, NY, USA, 1960.

[8] Julia Hockenmaier and Mark Steedman. CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. *Comput. Linguist.* , 33(3):355–396, 2007.

[9] Richard Johansson and Pierre Nugues. Extended Constituent-to-Dependency Conversion for English. In *Proceedings of NODALIDA 2007*, Tartu, Estonia, May 2007.

[10] David M. Magerman. *Natural Language Parsing as Statistical Pattern Recognition*. PhD thesis, Stanford University, 1994.

[11] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a Large Annotated Corpus of English: The Penn Treebank. *Comput. Linguist.* , 19(2):313–330, 1993.

[12] Igor Mel’čuk. *Dependency Syntax: Theory and Practice*. State Univ. of New York Press, 1988.

[13] Joakim Nivre. *Inductive Dependency Parsing (Text, Speech and Language Technology)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[14] Federico Sangati and Willem Zuidema. Unsupervised Methods for Head Assignments. In *Proceedings of the 12th Conference of the European Chapter of the ACL*, pages 701–709, Athens, Greece, March 2009.

[15] Lucien Tesnière. *Eléments de syntaxe structurale*. Editions Klincksieck, Paris, 1959.

[16] David Vadas and James Curran. Adding Noun Phrase Structure to the Penn Treebank. In *Proceedings of the 45th Annual Meeting of the Association of Comput. Linguist.* , pages 240–247, Prague, Czech Republic, June 2007.

[17] Hiroyasu Yamada and Yuji Matsumoto. Statistical Dependency Analysis with Support Vector Machines. In *Proceedings of IWPT*, pages 195–206, 2003.