A grammatical framework for the computational parsing of written Afrikaans sentences

by

Johannes Jacobus Swarts

Dissertation presented for the degree of PhD (Afrikaans and Dutch) in the Faculty of Arts and Social Sciences at Stellenbosch University

Supervisor:	Prof. R. H. Gouws
Co-supervisors:	Dr. J. Oosthuizen
	Dr. G-J van Rooyen

December 2019

Stellenbosch University https://scholar.sun.ac.za

Declaration

By submitting this dissertation electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: December 2019

Copyright © 2019 Stellenbosch University
All rights reserved.

Stellenbosch University https://scholar.sun.ac.za

Abstract

This dissertation investigates which grammatical framework is best suited to computationally represent and parse written Afrikaans sentences. This knowledge is necessary to build a large scale Afrikaans treebank – a resource which does not yet exist, but is a critical prerequisite for advanced endeavours in Afrikaans natural language processing. To gain this knowledge, we formally describe the building blocks of written Afrikaans from the perspectives of two major grammatical frameworks: constituency grammar and dependency grammar. Using these formal descriptions, we construct the first linguistically motivated treebank for Afrikaans, annotated with both constituency and dependency graphs. We perform k-fold cross-validation on multiple variations of this treebank with four state of the art sentence parsers, and fine-comb the results. Combining insights from the formal descriptions of written Afrikaans with the data obtained during parser evaluation, we conclude that dependency grammar outperforms constituency grammar at computationally representing the syntactic structure of written Afrikaans sentences under the conditions tested.

Stellenbosch University https://scholar.sun.ac.za

Opsomming

Hierdie proefskrif ondersoek watter grammatikale raamwerk meer geskik is vir die rekenaarmatige voorstelling en ontleding van geskrewe Afrikaanse sinne. Hierdie kennis is nodig om 'n grootskaalse Afrikaanse boombank te bou – 'n hulpbron wat tans ontbreek, maar 'n kritiese voorvereiste is vir gevorderde Afrikaanse natuurlike taalverwerking. Ten einde hierdie kennis te verwerf, beskryf ons die boublokke van geskrewe Afrikaans formeel vanuit die perspektiewe van twee dominante grammatikale raamwerke: samestellingsgrammatiek ("constituency grammar") en afhanklikheidsgrammatiek ("dependency grammar"). Hierdie formele beskrywings word ingespan om die eerste taalkundig gemotiveerde Afrikaanse boombank te bou wat annotasies vanuit beide grammatikale raamwerke bevat. Met verskeie variasies van hierdie boombank voer ons dan k-voudige kruisvalidering uit met vier toonaangewende sinsontleders en fynkam hul resultate. Aan die hand van hierdie resultate, sowel as die teoretiese insigte verkry tydens die formele beskrywings van geskrewe Afrikaans, lei ons af dat afhanklikheidsgrammatiek samestellingsgrammatiek oortref vir die rekenaarmatige voorstelling van die sintaktiese struktuur van geskrewe Afrikaanse sinne binne die getoetsde toestande.

Stellenbosch University https://scholar.sun.ac.za

Acknowledgements

I would like to express my sincere gratitude to the following people, without whom this dissertation would have been impossible:

- My wife, Jomari, for her unflinching support throughout this study – even after getting twins!
- Professor Rufus Gouws, Dr Johan Oosthuizen and Dr Gert-Jan van Rooyen (*Stellenbosch University*) for expert supervisory advice and a never-ending willingness to engage academically
- Professor Docter Uwe Quasthoff (*Leipzig University*) for sharing the Leipzig Corpora collection
- Professor Timothy John Osborne (*Zhejiang University*) for insights and advice about dependency grammar

Stellenbosch University https://scholar.sun.ac.za

Contents

Stellenbosch University https://scholar.sun.ac.za

Stellenbosch University https://scholar.sun.ac.za

Chapter 1

Introduction

Afrikaans is currently precluded from serious academic and commercial endeavours in natural language processing (NLP) due to a lack of resources for syntactic sentence parsing. The language is still considered resource scarce by many in the South African academic community. An audit performed in 2011 indicated that the available language resources are "of a very basic and exploratory nature" and that developments in NLP are "by and large uncoordinated" [43, p. 2847]. The mantra of resource scarcity has been repeated as preface to many academic publications on Afrikaans NLP [5] [6] [27] [30] [38] [63] until as recently as 2018 [87].

The key resource required to enable sentence parsing for any natural language is a treebank: a collection of sentences annotated with syntactic structures that describe the different components constituting each sentence (subjects, objects, clauses et cetera). These annotations are necessary for a sentence parser to learn from these structures and predictively reconstruct them for unseen sentences. Building an accurate and linguistically sound treebank is a time intensive but important step towards any advanced endeavour in NLP, including question answering, information extraction, speech recognition, machine translation and text summarization.

At the onset of this study, many large scale corpora for Afrikaans existed, but no treebank has been built from them. During the course of this study, a team at North West University (NWU) built a small treebank consisting of government texts which they dubbed "Afribooms". While Afribooms is a laudable first step towards a richer set of NLP resources for Afrikaans, the treebank contains a number of errors and theoretical problems which make

Stellenbosch University https://scholar.sun.ac.za

it impractical for further work in NLP. (An overview of these problems is provided in Appendix A.1.)

The lack of a linguistically sound treebank for Afrikaans is compounded by the fact that half of the theoretical underpinnings necessary to build it has not been developed. While there are many types of grammars, the majority of them fall into one of two theoretical frameworks: constituency grammar, or dependency grammar. (We explore the differences between these frameworks in detail in Chapter 3). While volumes about the syntactic structure of Afrikaans have been written from the perspective of various constituency grammars (the generative tradition is especially strong among Afrikaans syntacticians), very little exists in terms of dependency grammars. Dependency parsers built from dependency-based treebanks have been commonplace in NLP research for at least two decades, yet nothing presently exists to theoretically guide the development of similar datasets for Afrikaans. This is problematic since it is not evident from the available literature which framework is best suited to represent the structure of written Afrikaans for the purpose of computational sentence parsing.

The general trend in comparisons between constituency and dependency parsers, as noted by Kübler [57, p. 1], "is that the dependency parser performs slightly worse". This is evident from work on languages for which high volumes of data are available, such as English [74] and Chinese [20] [74]. Yet other languages with comparably high volumes of data such as German [57] and Swedish [44] do not fit into this trend, performing worse with constituency parsers. (We list the available data on the matter in Chapter 2.)

Since Afrikaans does not currently have a linguistically coherent treebank, and since constructing a treebank is expensive, knowing which framework to use for syntactic annotations is an important problem to solve. This study aims to offer a solution by providing an answer to the following research question:

What is an optimal grammatical framework for the computational representation and parsing of Afrikaans sentences?

We attempt to answer the question theoretically and empirically, by completing the following research objectives:

Stellenbosch University https://scholar.sun.ac.za

1. *Formally describe written Afrikaans from the perspective of constituency and dependency grammars.*

2. *Build a small but linguistically sound treebank of Afrikaans sentences with annotations from both frameworks.*

3. *Evaluate constituency and dependency sentence parsers on the treebank.*

4. *Answer the research question by combining the knowledge gained from the formal descriptions and the empirical experiments.*

In the following chapters, we guide the reader through our attempts to complete these objectives. We start this process with an overview of similar investigations done for other languages in Chapter 2, which allows us to define complementing null and research hypotheses for this study. This is followed by a theoretical section (Chapters 3 and 4), an empirical section (Chapters 5 and 6) and a conclusion in Chapter 7 that dependency grammar is the preferred grammatical framework to represent written Afrikaans sentences.

Stellenbosch University https://scholar.sun.ac.za

Chapter 2

Literature Study

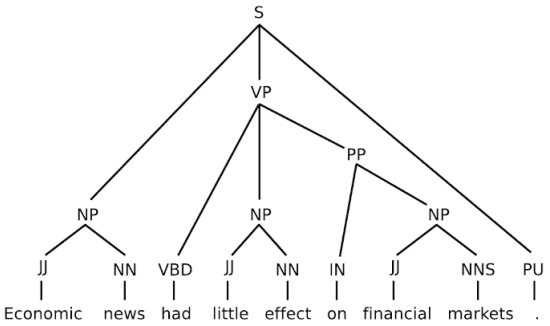
Sentence parsers are computer programs that extract linguistic information from digitised sentences. They assign hierarchical structure to linear input[1] by means of supervised machine learning: an approach in which a grammatical model is deduced from annotated training data (usually in the form of a treebank), and then applied on unseen input data by predicting the most probable parse that describes it. The predictions made by sentence parsers take on the form of graphs in which usually either the nodes or the edges represent the key building blocks of sentences: subjects, objects, verbs, adjectives, determiners and so forth. These elements serve as building blocks from which to generate more useful information further up in an NLP pipeline, allowing developers to identify things, attributes, locations, mood, manner, direction and many other aspects found in human language. Accurate sentence parsers are the building blocks for much of the human language technology we rely on in the 21st century – from translating between languages to dictating to smart devices what they should do for us.

A detailed history of sentence parsers is beyond the scope of this study, as are the various types of machine learning employed by different parsers. What is important for our purposes is the distinction between two major types of sentence parsers: constituency parsers and dependency parsers. Each type of parser is named after the grammatical framework on which they are based, and thus the type of parse trees they predict. We delve into the detail of each grammatical framework in Chapter 3. For now, it is

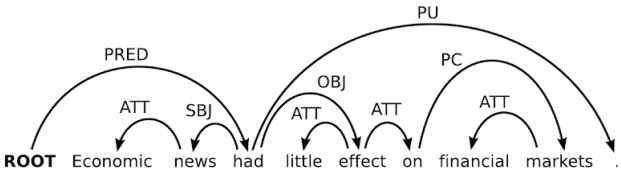
[1]This input is usually text, although recent advances in parsing audio with neural networks have been made. See Tran et al. in this regard [99].

Stellenbosch University https://scholar.sun.ac.za

sufficient to state that a constituency parser generates undirected acyclical graphs, and maps syntactic categories onto graph nodes, as illustrated in this example from Kübler et al. [58, p. 2]:



while a dependency parser generates directed acyclical graphs, and maps syntactic categories onto graph edges, as illustrated in Kübler et al. [58, p. 3]:



In order to hypothesise which grammatical framework – and by extension which type of parser – suits Afrikaans best, we can inspect other studies that empirically evaluated constituency and dependency parsers using the same treebank. In researching this topic, we learned that these types of comparative empirical studies are rare. Treebank annotation is usually done either from within a constituency or a dependency framework – often for historical reasons.[2] Treebanks containing annotations for both frameworks do exist, but are not commonplace. We find mention in the literature of two

[2]For example: the Penn Treebank for English (PTB) has been built within a constituency framework by American-based researchers who had little awareness of the work of Tesnière (the father of modern dependency grammar). They were influenced by the work of Chomsky, whose ideas have fully taken hold of Syntax when work on PTB started [78, p. 29]. The Penn Treebank eventually influenced other research teams to build similar treebanks for other languages, including Arabic, Chinese and Korean [62] [104] [45]. Likewise the Prague Dependency Treebank (PDT) for Czech was built by researchers influenced by Czech syntactician Šmilauer, whose work was heavily influenced by Tesnière [61]. The Prague Dependency Treebank eventually influenced the creation of treebanks for other Slavic languages, including Croatian and Serbian [7] [50]. In each of the aforementioned cases, the suitability of a grammatical framework was not something to bother about, as each framework were to the linguists using it like water to a fish.

Stellenbosch University https://scholar.sun.ac.za

treebanks that have purposefully been built with syntactic annotations in both frameworks:

- The **German** Tiger treebank [15]
- The **Swedish** Talbanken05 [77][3]

Other treebanks that are published with both constituency and dependency annotations exist, but were built by annotating sentences with graphs only in one framework, and later heuristically converting them to the other framework.[4] Such is the case for:

- The **French** Sequoia treebank [17]
- The **Italian** VIT and TUT treebanks [14] [31]
- The **Polish** Składnica frazowa [103]
- The **Portuguese** Floresta Sintá(c)tica [88]
- The **Spanish** Cast3LB [24]
- The **Urdu** KON treebank [1]
- The **Vietnamese** treebanks [72]

Of the aforementioned, all datasets except TUT have been created as constituency treebanks, and augmented with dependency annotations through rule based conversion. (TUT has been built as a dependency treebank, and converted to a constituency treebank.)

Investigations into which framework yields better scores when evaluated on sentence parsers are equally rare. We find three investigations into the impact of grammatical frameworks on parser outputs in the literature:

1. Kübler and Prokić [57] notes that LoPar (constituency) and MaltParser (dependency) achieve significantly differing scores when evaluated on the German Tüba-D/Z treebank (the dependency parser achieves

[3] Talbanken05 is a modernised version of Talbanken76 [36] [37] which we do not count as a fourth example.

[4] Within the context of this study, "heuristically" refers to rule based pattern matching (as opposed to prediction based classification).

Stellenbosch University https://scholar.sun.ac.za

83.4%, while the constituency parser achieves 75.3%.) They hypothesise that this is because the dependency framework handles coordination and long-distance dependencies better, and show that this assumption is valid.

2. Hall et al. built a parser that is able to produce constituency trees with dependency relations mapped onto their edges, and vice versa, as seen in this example from [44, p. 286]:

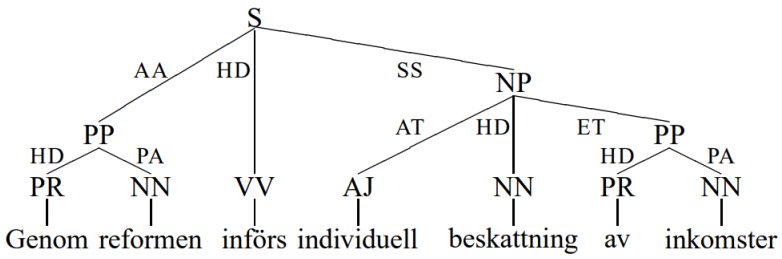


Figure 3: Hybrid representation

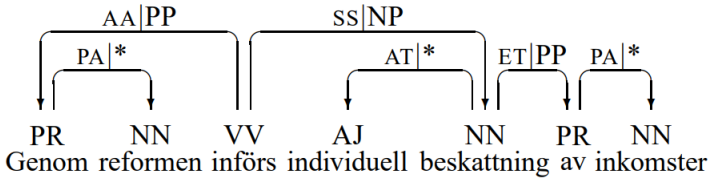


Figure 4: Dependency encoding of hybrid representation

Using Maltparser to evaluate their schema on Talbanken05, Hall et al. found that dependency relations were predicted with higher precision than constituency trees (82% as opposed to 75%).

3. Nivre et al. [76] investigated the scores obtained by MaltParser trained on treebanks for ten languages. For each language, they compare MaltParser's best score with existing state of the art scores. In three of these cases, the state of the art score is from a constituency parser. This provides us with three comparisons of constituency and dependency parsers, summarized in the table below. For all three languages (Bulgarian, Chinese, English), the state of the art constituency score is superior to the best the dependency parser can offer.

Stellenbosch University https://scholar.sun.ac.za

Figure 2.1: Constituency versus dependency scores in Nivre et al.

Language	Constituency	Dependency
Bulgarian	73.6% [96]	62.6% [76]
Chinese	81.8% [20] [19]	79.8% [76]
English	92.1% [105]	86.3% [76]

Besides the three studies mentioned above, we have also been able to find separate, unrelated experiments performed with different parsers on the constituency and dependency versions of the treebanks which obtained their annotations in a second grammatical framework heuristically. We identify examples for four languages in the literature, namely French, Italian, Polish and Vietnamese:

1. Lavelli et al. evaluated MaltParser on the dependency based TUT treebank of Italian, achieving an LAS of 91.23% [60]. Lavelli converted the same treebank to constituency based graphs, evaluated the Berkely parser on it and achieved an F1 score of 83.83% [59]. Here, the dependency framework yielded better results.

2. Woliński and Rogozinska evaluated the Świgra parser on the treebank Składnica frazowa, and attained an F1 score of 94.1% [102]. This is several percentage points higher than the best dependency score – an LAS of 88.8% – obtained by Wróblewska and Woliński [103].

3. Nguyen et al. obtained an F1 score of 78% for sentences with less than 40 words from the Vietnamese Treebank, using an unnamed constituency parser [71]. This is several percentage points higher than Nguyen et al.'s LAS score of 73.53% when evaluating a converted version of the Vietnamese Treebank with the BistG parser built by Kiperwasser and Goldberg [70] [53].

Taking into account every experiment we can find in the literature, we summarize the grammatical framework that yielded the highest experimental scores for each language as follows:

Stellenbosch University https://scholar.sun.ac.za

Table 2.1 Grammatical framework yielding highest parser evaluation score for language evaluated

Best framework	Languages
Constituency	Bulgarian, Chinese, English, Polish, Vietnamese
Dependency	French, German, Italian, Swedish

While it does not seem obvious from these results which framework is best suited for Afrikaans, the German and Swedish results hint at dependency grammar. This is because, as Germanic languages, they share traits with Afrikaans (more so than English which is also Germanic, but contains significant Romance influences). We therefore put forward the following hypotheses for this study:

Null Hypothesis (H0). *Types of grammatical frameworks do not impact the efficacy of representing and parsing Afrikaans sentences.*

Research Hypothesis (H1). *Written Afrikaans sentences are most accurately represented and parsed by dependency parsers.*

In the next chapter we delve into the two grammatical frameworks on which most modern day sentence parsers are based, and explore how they should be used for computational purposes.

Stellenbosch University https://scholar.sun.ac.za

Chapter 3

Grammar

This chapter lays the theoretical foundation for this study by discussing and contextualising the following topics:

- Principles to which grammars of natural languages used for computational purposes should adhere (Section 3.1)
- The merits of investigating two grammatical frameworks (Section 3.2)
- Formal properties of constituency grammar (Section 3.3)
- Formal properties of dependency grammar (Section 3.4)

3.1 Principles

"Del rigor en la sciencia", a story by Argentinian author Jorge Luis Borges, recounts the tale of a kingdom where the science of cartography became impractically intricate. The English translation of the story – in its entirety – reads [12]:

> In that Empire, the Art of Cartography attained such Perfection that the map of a single Province occupied the entirety of a City, and the map of the Empire, the entirety of a Province. In time, those Unconscionable Maps no longer satisfied, and the Cartographers Guilds struck a Map of the Empire whose size was that of the Empire, and which coincided point for point with it. The following Generations, who were not so fond of the Study of

Stellenbosch University https://scholar.sun.ac.za

> Cartography as their Forebears had been, saw that that vast map was Useless, and not without some Pitilessness was it, that they delivered it up to the Inclemencies of Sun and Winters. In the Deserts of the West, still today, there are Tattered Ruins of that Map, inhabited by Animals and Beggars; in all the Land there is no other Relic of the Disciplines of Geography.

The story, often employed by professors to convince their students that modern human societies live in Baudrillardian simulacra, contains a lesson in frugality that is relevant for our purposes.

The purpose of our research is to discover new theoretical knowledge by means of computational techniques. To this extent, these techniques are limited to practicalities. Although today's computers are powerful machines, they remain units with finite resources. The essence of a computer, we hold, is not simply to compute, but to compute useful results within a practical time frame. If this is not an attainable goal, developing computational systems become futile. For example: building infrastructure to download high resolution cinematic films to the Curiosity Rover on Mars is technically possible, but practically and financially infeasible. Likewise, building a password cracking system to crack 30-character alphanumeric passwords is entirely possible, but with current technology will likely require trillions of centuries to brute force a single password.

It would by no means be a sensible endeavour if our methods to derive theoretical conclusions turn out to be impractical. Should the software not yield sensible results or require unnecessary amounts of computational resources and time, it would not serve any useful purpose, and any attempts to make deductions about the feasibility of grammatical models involved would be pointless.

When building a grammatical model for computational purposes two competing factors come into play, and need to be planned for. The first factor is the model's ability to describe the language in question; the second is the computational feasibility of the model.

In order to yield useful results, our models should describe the syntax of (written) Afrikaans sentences to **a sufficient degree of accuracy**. Should it fail to do so, the resultant parser built from it will fail to properly recognise useful sentential structures. The results drawn from its output will have

Stellenbosch University https://scholar.sun.ac.za

little to no scientific relevance, and any time spent crafting a dataset based on this model for training purposes would be wasted effort. To avoid this, the models should contain enough information to describe the syntactic structures found in written Afrikaans.

The second factor is the models' **computational feasibility**. In the context of this study, this factor plays a role in two domains: corpus annotation as well as actual parsing. Work and results in both areas will suffer from a cumbersome model, which will incur too much processing overhead for humans (who have to annotate training data in a finite amount of time) and computers (that have limited resources, and could take impractically long to learn and apply the model). In this context, an ideal grammar would also be one that avoids unnecessary information, that is to say: a grammar where rules that increase processing time or storage requirements without contributing any significant value to the final output must be avoided.

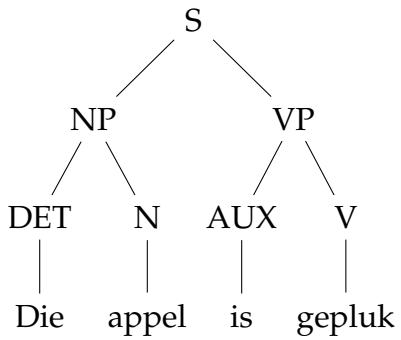
Complexity must therefore be balanced with accuracy. If either one weighs too heavily in a grammatical model, the resultant output will not be useful. Consequently, although it might be desirable in certain branches of Linguistics to describe natural language by means of notationally detailed and complex frameworks, it does not follow that this level of detail is suitable for developing sentence parsers.

A currently popular paradigm amongst many linguists, for example, is "the Minimalist Program". The program is an approach to analysing language centred around the notion of Universal Grammar (UG) – the idea that humans are born with an innate "language faculty" or "language organ", which enables them to rapidly acquire any language to which they are exposed at an early age. Popularised by Chomsky [23], grammars defined within this program contain the implicit assumption that, although natural languages contain differing syntactic phenomena, their structures can be explained with reference to a single, universal framework. Because of this, Minimalist analyses of very short sentences often result in elaborate syntactic structures that become difficult to render on paper. Consider, for instance, the following Afrikaans sentence:

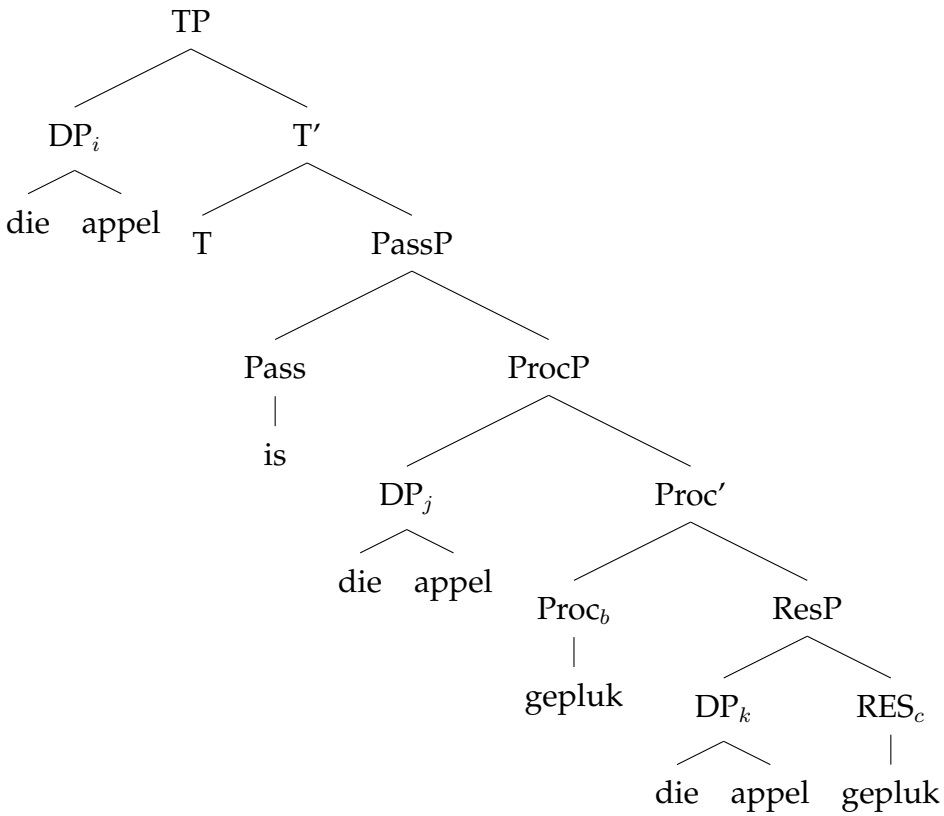
(1) die appel is gepluk
the apple PAST-be picked
'the apple was picked'

Stellenbosch University https://scholar.sun.ac.za

An analysis of this sentence within a simple phrase structure grammar could look like the following:



Alternatively, if described in terms of the devices associated with the Minimalist approach, the analysis could be represented as follows:[1]



The two analyses above have different goals. The former classifies different syntactic constituents and orders them hierarchically, thereby serving as a sort of linguistic taxonomy of the sentence's surface structure. The latter aims

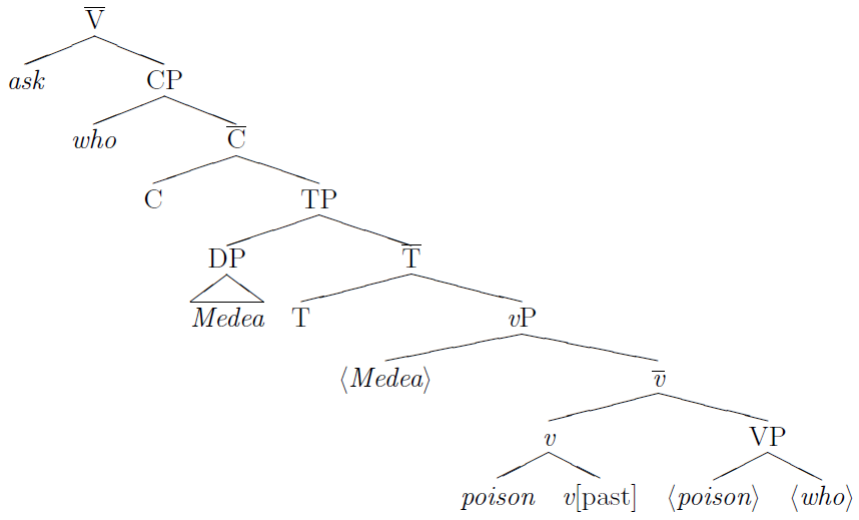
[1] Adapted from [56, p. 107]

Stellenbosch University https://scholar.sun.ac.za

to describe the mechanics of how a sentence in the passive voice is generated, on the grounds of the principles postulated within one of the branches of the Minimalist Program – that of Nanosyntax. The latter is problematic from the perspective of a sentence parser, since parsers do not change the order of the words they analyze. Yet, when the nodes with syntactic labels are removed from the tree, a different sequence of strings from the original written sentence remains: *die appel is die appel gepluk die appel gepluk*. The problem worsens when inversions come into play. Consider the following example of WH-movement in Adger [3, p. 284]:

(2) I asked who did Medea poison

Adger analyses the sentence's subordinate clause with the following graph [3, p. 285]:



The words in this graph form the string *ask who Medea Medea poison poison who* which, besides containing extra words, also does not reflect the linear order of the written sentence.

While some disciplines might prefer a Minimalist analysis over the simpler phrase structure analysis for its ability to align to the hypothesis that there exists *a posteriori* phenomena like Universal Grammar and sub-morphemic syntactic units, it does not follow that these hypotheses are useful when formally describing the structure of a single language in its written form. The various movement and merging operations postulated

Stellenbosch University https://scholar.sun.ac.za

within Minimalist Syntax, although completely valid within this particular paradigm, become cumbersome (and, in the cases illustrated above, impractical) in a computational context, because they preserve meta data serving to account for the differences between all known languages in the model of a single language. This leads to computational impracticalities as described by Schneider [93, p. 16]:

> If we really treated translations as transformations, in the sense that one word form is transformed into another, they would be as very time-intense and unwieldy to parse as Chomskyan transformations: We only know the "surface" element after the transformation (or translation: I will only use the term transformation here) from the *chaîne parlée* (which Chomsky calls numeration) and have to find a corresponding "deep" structure before the translation or transformation. We have to perform a "backwards" transformation. Because of the complexity of transformations, because Transformational Grammar and Government and Binding theory are generative we are forced to more or less blindly generate zillions of surface structures and see if any of them fits the input string. This is a major reason why Chomskyan grammars are unparsable in practical terms.

The latest Minimalist grammars do not make use of government and binding theory any more, but are (as illustrated by reference to [56] above) of an equal degree of notational complexity. Including this overhead in a computational model is bound to incur a heavy performance cost, while adding little useful information about the actual data being parsed. A syntactic model of this complexity is therefore unsuitable for the purposes of this study. Using it will be akin to using the Map of the Kingdom.

The mistake of including unnecessary complexity in a syntactic model has been made more than once during similar endeavours pertaining to Germanic languages. (None, it has to be noted, with the extreme consequences that Schneider describes.) Two examples, taken from models built for German and Afrikaans part of speech tagsets, are cases in point:

- The Stuttgart/Tübinger Tagset [49] (STTS)

Stellenbosch University https://scholar.sun.ac.za

- the Afrikaans Part of Speech tagset (APOS) defined by Pilon [81]

STTS, which forms the basis of NEGRA (a German annotated corpus of 20 602 sentences [95]), defines three distinct tags with which to classify punctuation:

Table 3.1 Punctuation labels in the Stuttgart / Tübinger Tagset

Tag	Description	Meaning	Example
\$,	Komma	Comma	,
\$.	Satzbeendende Interpunktion	Sentence terminating punctuation	. ? ! ; :
\$	Sonstige Satzzeichen; satzintern	Other punctuation marks; internal punctuation	- [] ()

Although a case can certainly be made for a distinct category for sentence terminating punctuation (\$,) and a second category for non-sentence terminating punctuation (\$.), it is unclear why an entire tag is dedicated solely to the comma. Although the comma plays an important orthographic role in written German (it must, for example, precede a relative clause), the data necessary to classify the character ',' as a comma is already contained *within the character itself*. The category 'Komma' does not tell us anything new or useful about said comma. It adds complexity to the NEGRA treebank's taxonomy without any apparent benefit.

A more extreme example of this complexity is found in APOS, which contains 32 distinct tags describing Afrikaans pronouns [81, 45-46] (see Appendix B.1). This is more than double the total number of part of speech tags used to describe German – a morphologically rich language – in STTS. Out of the 32 distinct part of speech tags reserved for Afrikaans pronouns, 29 are assigned to single, unique strings. As in the case of the STTS "Komma", all the information required to identify these pronouns is already contained within the strings themselves. It is hard to see the benefit of a tagset of this magnitude for Afrikaans, a language known for its "extreme morphological impoverishment" [8, p. 20]. (Pilon herself managed to increase the accuracy rate of her part of speech tagger with over 7% by simplifying APOS from 139 to 13 tags [81, p. 116].)

Stellenbosch University https://scholar.sun.ac.za

It is not within the scope of this study to make a judgement about the feasibility of Universal Grammar or to debate complexity in part of speech tag sets. The point to be made is simply that frugality should be prioritised in the creation of formalisms that model natural language for computational purposes. They need to describe the language in question with sufficient accuracy without sacrificing structural elegance or impeding computational performance. With this goal in mind, up-to-dateness with the latest theories of syntax and complex linguistic models does not represent a priority for this study.

The third requirement for the model is that its scope must be limited to a description of **the syntax of written Afrikaans**, that is, the rules that govern the structure of its sentences and the patterns according to which parts of sentences may be ordered. The models should therefore not concern themselves with morphology, semantics, semiotics or other linguistic fields, as these are not within the scope of this study. The grammars should also not attempt to contain information about the syntactic structures of any other natural languages, but should focus exclusively on Afrikaans.

The three requirements for a grammatical model for our purposes can therefore be summarised as follows:

Requirement 1. *The grammatical model must be sufficiently accurate.*

Requirement 2. *The grammatical model must contain only necessary information.*

Requirement 3. *The grammatical model must describe the syntax of Afrikaans.*

These requirements will inform the scope of the rest of this study.

3.2 Grammatical formalisms

The goal of this study is to compare the results of two types of grammatical frameworks on the accuracy and efficiency of computational parsing of Afrikaans sentences: constituency grammar and dependency grammar. The raw material that will form the empirical basis for this investigation is written language, and so for the purposes of this study the term "grammar" will refer to formal systems that are able to describe written sentences in a single language while being structurally parsimonious.

Stellenbosch University https://scholar.sun.ac.za

The two frameworks in question have had significant development and contributions made to them since the middle of the previous century. Both were spurred by two seminal linguistic works, published fairly close in time to each other. For Constituency Grammar it was "Syntactic Structures" (SS), a series of lecture notes published by Noam Chomsky in 1957. For Dependency Grammar it was "Éléments de syntaxe structurale" (ESS), Lucien Tesnière's treatise on syntax published posthumously in 1959. Each work introduces a unique formal approach to deal with syntax as a subject separate from other linguistic disciplines (such as phonology or semantics) – a premise that, while wholly tenable today, was not always acceptable in academic circles [98, p. xxxv]. Although the formalisms built on top of the ideas in these works have evolved considerably, the core tenets contained in both SS and ESS remain important theoretical foundations from which to construct the grammars for this study.

Constituency grammars are sometimes also referred to as "phrase structure grammars", and sometimes contrasted with the very same term [13, p. 8]. As far as terminology for this study goes, the terms "constituency grammar" and "phrase structure grammar" will be used interchangeably. This will be done deliberately so as to contrast both with "dependency grammar". While certain linguistic paradigms classify phrase structure grammars as non-transformational constituency grammars (thereby taking the former to be a subset of the latter[2]), this distinction will be of little value for this study. Parsers, by their very nature, cannot generate natural language. Rather, they require already generated language on which to operate. The transformations found in many generative grammars are therefore not relevant to this endeavour, as written sentences have already been successfully generated. As such the terms can and will be used interchangeably.

On a formal level, constituency and dependency models of language can be contrasted with each other because they are not strongly equivalent. This is illustrated by Matthews [64] who shows that dependency and constituency grammars are not mere notational variants of each other, but rather weak equivalents. This means that:

> ...for any dependency grammar there is a phrase structure gram-

[2] I am indebted to Tim Osborne (Zhejiang University, Hangzhou) for this insight.

Stellenbosch University https://scholar.sun.ac.za

> mar which will generate an identical set of sentences; likewise, for any phrase structure grammar. . . the same set of sentences can be generated by a dependency grammar. [64, p. 84]

Matthews notes, however, that:

> There are things we can say in a constituency grammar which we cannot say in a dependency grammar, just as there are other things which we can say in a dependency grammar but not in a constituency grammar. [64, p. 88]

There is therefore a theoretical advantage in making a distinction between the two formal representations of syntactic structures. Moreover, as Abney notes, there are also important practical benefits to the computational linguist as a result of treating each framework as a valid representation of human language [2, p. 7]:

> From a mathematical perspective, equivalences between types of representation provide useful tools. Some results are easier to discover or prove using one type of representation; some are easier to discover or prove using the other. The equivalence allows one to use either type of representation interchangeably.
>
> Judging from the literature, the linguist's attitude tends to be rather different. If one shows that two representations are equivalent in some respect, the immediate response is to seek other arguments for why one or the other representation is right. Either the correct structure is a phrase-structure tree, or it is a dependency tree, but it certainly cannot be both.

Being a multi-disciplinary work requiring a utilitarian approach, this study will not adopt the approach of Abney's imagined linguist. Our purpose is not to prove any one linguistic theory or framework "better" than any other. Rather, it simply aims to discover which framework is better suited to the computational parsing of Afrikaans.

The subsequent sections provide an overview of the formal properties of constituency and dependency grammar, and stipulate the notational variants used to represent them in practice.

Stellenbosch University https://scholar.sun.ac.za

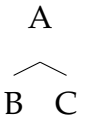
3.3 Formal properties of Constituency Grammar

Constituency grammars have historical roots in the subject-predicate structures found in term logic. This view of language as logic dates back to Aristotle, although it should be noted that the Aristotelian subject-predicate distinction does not coincide with the subject-predicate analysis found in modern linguistics [94, 30]. Rather, the structure of modern constituency grammars resembles the "Concept and Object" relation expounded on by Frege [40]. In this relation, a logically "saturated" component (like a proper noun) is connected with a logically "unsaturated" component[3] (such as a copular verb phrase), as in the following bracketed example:

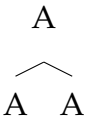
(3) {Johan}$_A$ {is a student}$_B$

Here, *A* is logically saturated (complete), while *B* cannot stand on its own and requires the presence of *A* to form a logical whole: the complete sentence.

Constituency grammars represent sentences as rooted trees: acyclical undirected graphs in which any two vertices are connected by exactly one path. In these graphs, the words of a sentence act as terminal vertices (vertices farthest from the root vertex, or "leafs"). These vertices, in turn, are contained by parent vertices representing syntactic categories and are non-terminal (root or branch nodes). Parent vertices in this model are able to contain child vertices of the same category as itself, providing constituency grammars with the property of recursion. The recursive character of constituency grammar means that this structure:



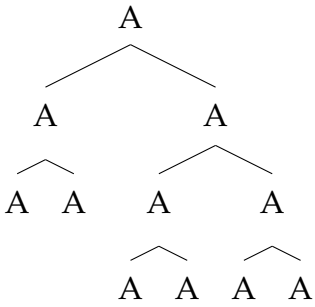
is equally valid as this structure:



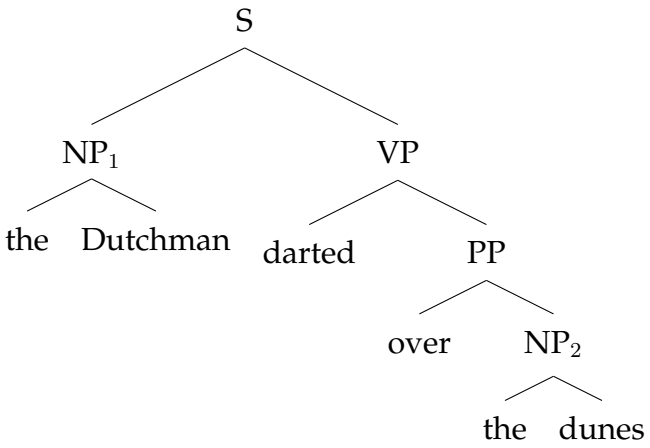
which in turn is as valid as this one:

[3] For Frege, an "unsaturated" component simply means an "incomplete" component with "the need of supplementation" by another component. Conversely, a "saturated" component is "complete", and has no such need. Frege admits that these references are "of course only figures of speech"[40, pp. 179–180]

Stellenbosch University https://scholar.sun.ac.za



"In the crudest form of constituency", therefore, "a unit *a* is related to a neighbouring unit *b* solely by their placement within a larger unit *c*" [64, p. 73]. The combination of these neighbouring units are dubbed "syntagms" and correspond – within the context of natural languages – to conventional syntactic categories: noun phrases, verb phrases, prepositional phrases and so forth, as in the following example:



where each of S, NP_1, VP, PP and NP_2's children constitute a syntagm on their own, while those of NP_1 and NP_2 (*"The Dutchman the dunes") and NP_1 and PP ("*The Dutchmen over the dunes") strung together would not qualify as a unit, or syntagm.

Sentences in constituency grammars are derived by string rewriting operations called "rewrite rules". These rules are applied exhaustively to original "start strings" until the final derivation constitutes a complete and grammatically valid sentence. Originally formalised by Chomsky as [Σ, F] grammars, these operations can be defined as "a finite sequence of strings, beginning with an initial string of Σ, and with each string in the sequence being derived from the preceding string by application of one of

Stellenbosch University https://scholar.sun.ac.za

the instruction formulas of F"; when all possible rewrite rules have been exhausted, the resulting string is called a "terminated derivation" [21, p. 29].

Formally, a constituency grammar consists of a tuple G = (V, T, R, S)[4], where:

- V is a finite set of non-terminal symbols ("variables")
- T is a finite set of terminal symbols ("terminals")
- R is a set of finite relations of V to (V ∪ T)*
- S is the start symbol, with S ∈ V

The recursive power of constituency grammars is owed to the Kleene operator found in *R* (*). In this specific context, the operator indicates that zero or more occurrences of the union of V and T are considered syntactically valid. To illustrate this union, consider that if the allowable non-terminal and terminal symbols of a constituency grammar G_{toy} were:

- V = {A, B}
- T = {x, y}

the following combinations of symbols found in V and T will all be equally valid:

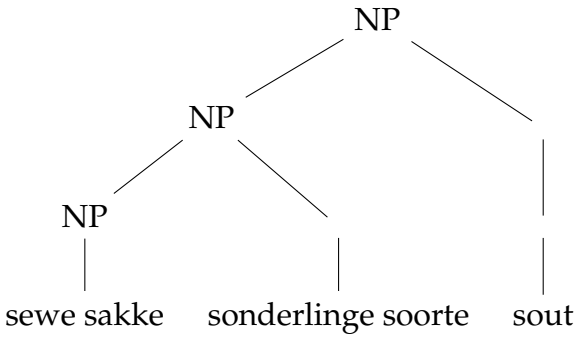
Relation	**Reason for validity**
A → x	Relation of V to T
A → B	Relation of V to V
A → Bx	Relation of V to union of V and T
A → AyAx	Relation of two occurrences of a union of V and T

The property illustrated in the fourth example in the table above implies that variables can contain any number of child variables of their own type, and aligns to constituency grammars' ability to recurse. A well known linguistic example of this phenomenon in natural languages – and indeed in Afrikaans – is compound noun phrases: noun phrases which contain other noun phrases, as illustrated in the example below:

[4]The symbols used in this example are arbitrary, and differs from source to source in the literature.

Stellenbosch University https://scholar.sun.ac.za

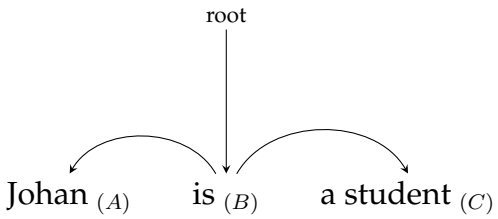
(4) sewe sakke sonderlinge soorte sout
seven bags rare types salt
'seven bags of rare types of salt'



3.4 Formal properties of Dependency Grammar

Dependency grammar takes a different approach to modelling human language than constituency grammar. First developed by Tesnière in the 1950s in ESS (around the same time as Chomsky entered the linguistic arena), the various grammars developed within this approach eschew the division of the clause into a subject and a predicate.[5] Where constituency grammars strive to "find the logical opposition between subject and predicate in the sentence" [98, p. 98], dependency grammar relies on verb centrality, meaning that the verb is the root of all clause structures.[6]

In a dependency relation an element can be connected to any other element in one of two roles: as a governor or a dependent. To illustrate, consider the example sentence (2) from the previous section, this time structured as a dependency graph:



[5]Tesnière himself argued that this division belongs to the field of Logic, and has nothing to do with Linguistics [98, p. xxxix].

[6]This contrasts it with constituency structures, where a start symbol subsuming unsaturated and saturated elements (such as S in early generative studies) is considered the root.

Stellenbosch University https://scholar.sun.ac.za

Here,

- *B* (a copular verb) is the root of the sentence.
- *A* and *C* are *B*'s dependents, and are both said to be "governed" by *B*.
- The direction of each of the edges ("arrows") indicates the type of relationship that exists between two nodes. In this representation, edges are directed towards dependents.

Formally, a dependency grammar is a tuple G = ($\sum$, P, S) where:

- $\sum$ is a set of word categories
- P is a set of productions
- S is the set of start symbols, and S $\in \sum$

Each production is of the form X (α; β), where:

- X $\in \sum$
- α is a sequence of categories, linearly to the left of X
- β is a sequence of categories, linearly to the right of X

For each production, X acts as the governor of α and β, and α and β act as dependents of X. Abney describes each production as "licensing" a dependency tree, and states that a tree [2, pp. 1–2]:

> is licensed by G if every node is licensed by some production of G. A production X ($Y_1, \ldots, Y_m$; $Z_1, \ldots, Z_n$) licenses a node if the node is of category X, its left dependents are of categories $Y_1, \ldots, Y_m$, in that order, and its right dependents of categories $Z_1, \ldots, Z_n$, in that order.

In simpler terms, this means that a valid dependency tree can be generated from the nodes of a sentence if every node adheres to the rules in P. These nodes (words) must be present in $\sum$, "a lexicon which assigns words to categories" [2, p. 2], and enable the grammar to map each category in $\sum$ to a word in the lexicon.

Stellenbosch University https://scholar.sun.ac.za

In a dependency grammar, the words in a sentence are modelled as vertices on an acyclical directed graph.[7] The verb is assumed to be the root of the graph, and is said to "govern" its immediate syntactic dependents. (Which verb assumes this role is the topic of section 4.5.2.) With the exception of the root vertex, every other vertex in the graph can be governed by exactly one other vertex. These vertices are dependents of their governers and can in turn govern other vertices. These relationships are strictly binary: each dependent can have only one governor. A governor, however, is permitted to have more than one dependent.

According to Nivre [74, p. 74], given a set R of dependency types, a dependency graph for a sentence x = (w_1, ..., w $_n$) can be considered a labelled directed graph G = (V, E, L) where:

- V = Z $_{n+1}$
- E $\subseteq$ V $\times$ V
- L: E $\rightarrow$ R

In the above, V is "the set of non-negative integers up to and including n" [74, p. 74]. Given any sentence, therefore, each token (word) can be numbered with a natural number, as can be seen in the following example:

(5) [ROOT] John eats green apples
0 1 2 3 4

Thus "every token index i of the sentence is a node ($1 \leq i \leq n$)" and, importantly, there is "a special node 0, which does not correspond to any token of the sentence and which will always be a root of the dependency graph (normally the only root)" [74, p. 74]. Furthermore, E represents the set of arcs (edges) connecting ordered pairs of governers and dependents (i, j), with each pair containing two of the nodes found in V. These edges represent grammatical relations between nodes. Lastly the function L "assigns a dependency type (arc label) r $\in$ R to every arc e $\in$ E [74].

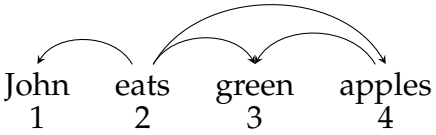
Beside the first node being the root of a dependency graph, three additional constraints are also commonly found in the literature [74, p. 75]. (For each of

[7] This contrasts it with constituency grammars, where sentences are modeled as undirected graphs.

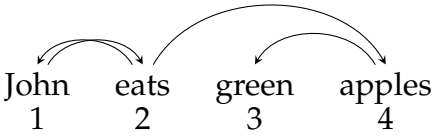
Stellenbosch University https://scholar.sun.ac.za

the constraints listed below, the symbol $\rightarrow$ indicates a dependency relation, and the root node is being omitted for the sake of clarity.)

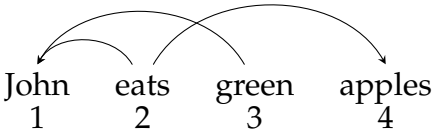
1. Every node in graph G has **at most one head** (if $i \rightarrow j$, then there is no node k such that $k \neq i$ and $k \rightarrow j$). The following graph is therefore invalid, since node 3 has two heads, thereby violating the single head constraint:



2. The graph G is **acyclic** (if $i \rightarrow j$ then not $j \rightarrow i$). The following is therefore invalid, since node 1 governs node 2, while at the same time being governed by node 2:



3. The graph G is **projective**, that is to say edges between nodes do not cross one another (if $i \rightarrow j$ then not $i \rightarrow k$, for every node k such that $i < k < j$ or $j < k < i$). The following graph is therefore non-projective, since $2 \rightarrow 4$ crosses $3 \rightarrow 1$.



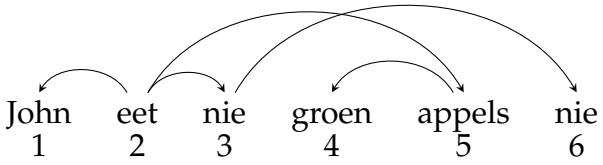
Whereas the single head and acyclical constraints are commonly assumed in the literature, this is not the case for the projectivity constraint – especially in languages with a relatively free word order. This is also the case for Afrikaans, especially where its double negation – one of the salient characteristics of the language – is concerned. The challenge presented by this phenomenon can be illustrated by means of the following sentence:

(6) John eet nie groen appels nie
John eat not green apples not

Stellenbosch University https://scholar.sun.ac.za

'John does not eat green apples'

The following dependency graph can be drawn for this sentence:



In the example above, node 3 (the adverb indicating negation) governs node 6 (the negation particle), thereby resulting in crossed dependency edges with the edge between node 2 and node 5. This pattern is common in Afrikaans, and therefore any dependency grammar for Afrikaans will readily violate the projectivity constraint. For this study, dependency grammar will be assumed to adhere to the following definition presented by Nivre [74, p. 75]:

> Let G = (V,E,L) be a well-formed dependency graph, satisfying SINGLE-HEAD and ACYCLICITY.

Dependency grammars are implicitly stratified, with a clear distinction between linear and structural order. Tesnière posited this in ESS, stating that "speaking a language involves transforming structural order to linear order, and conversely, understanding a language involves trans-forming linear order to structural order" [98, p. 12]. This property will be illustrated throughout the next chapter.

3.5 Summary

This chapter has broadly outlined the formal properties of constituency and dependency grammars. These grammars have been described as formal systems that map words and syntactic categories onto graph nodes. Constituency grammars have been shown to be parsable as acyclical undirected graphs, with words being terminal-nodes and syntactic categories acting as non-terminal nodes. Edges in these graphs are unlabelled and indicate parent-child relationships. Dependency grammars have been shown to be acyclical directed graphs where words are mapped onto the graph as terminal nodes, and syntactic categories or roles are mapped onto edges as edge labels. In short, then, the key difference between the two formalisms

Stellenbosch University https://scholar.sun.ac.za

is the mechanisms they provide to indicate syntactic categories: in the case of constituency grammar, graph nodes are utilised whereas in dependency grammar, graph edges are utilised.

Stellenbosch University https://scholar.sun.ac.za

Chapter 4

Afrikaans Syntax

In this chapter we provide an overview of the syntactic patterns occurring in well formed Afrikaans sentences, and describe each from the perspective of constituency and dependency grammar. We focus on so-called *Standaardafrikaans* ("Standard Afrikaans"), abstracting away from potentially interesting but less prominent constructions found in dialectic variations.

The purpose of this chapter is not to provide a comprehensive overview of the syntax of Afrikaans. Other publications on the matter already exist, and provide much more detail than this study allows.[1] While comprehensive, these publications are heavily influenced by the generative tradition and ultimately serve a different goal than aimed for in this study. We are not interested in language generation, but in analysing the patterns of already generated sentences in order to map linguistic categories onto them by means of supervised machine learning. To do so, we require training data built from clear guidelines on how to annotate the syntactic structures of Afrikaans using both constituency and dependency formalisms. No such guide exists, hence this chapter aims to provide one.

The sections in this chapter provide an overview of constituency and dependency parses for each type of sentence identified, and a symbolic representation of the linear order of its key building blocks (heads, arguments and adjuncts) as a string of non-terminal variables.[2] Parsing ambiguities encountered in each section will be disambiguated at the end of the chapter.

[1] Notably that of Murray [100], Donaldson [34], Ponelis [82] and Biberauer [9].

[2] We use the term "argument" to mean "valency dependant" and "adjunct" to mean "non-valency dependant" similar to Przepiórkowski [86]

Stellenbosch University https://scholar.sun.ac.za

This allows maximum consistency during the treebank annotation phase.

Manually presenting the bulk of valid syntactic patterns in written Afrikaans becomes a futile task if undertaken in extreme detail. If one takes the following factors (all of which play a role in determining word order) into account, the unique combinations possible become impractical to discuss:

Factor	**Type**
Form	Declarative
	Interrogative
	Imperative
Tense	Present
	Past
	Future
Valency	Intransitive
	Monotransitive
	Ditransitive[3]
Voice	Active
	Passive

To compensate for this, the structures presented in this chapter will condense multiple factors into single examples. This will be done by merging example sentences with identical voices and tenses, but differing verbal valencies, into a single section. (This is possible because changes in valency do not have a significant effect on word order in Afrikaans.) Each example sentence will therefore be a "macro sentence", containing in itself intransitive, transitive and ditransitive sentences. For ease of reading, colours will be used to highlight the position of verbal arguments:

[3]Tritransitive structures are allowed in Afrikaans, but the third object is always headed by a preposition and found in the same position as other verbal adjuncts, as in the example: *Johan betaal Jomari R7 vir die appels* ('Johan pays Jomari R7 for the apples'). This, combined with the fact that not all grammars acknowledge tritransitive verbs [69], makes it unnecessary to include it as a factor determining Afrikaans word order.

Stellenbosch University https://scholar.sun.ac.za

Table 4.1 Colours used in graphs to indicate grammatical roles

Grammatical role	**Color**
Subject	Green
Direct object	Blue
Indirect object	Olive
Passive agent	Red
Adjectival modifier	Violet

For constituency grammar, we employ the following set of basic labels in each example sentence:

Table 4.2 Constituency labels

Constituent	**Label**
Complementiser phrase	CP
Tense phrase	TP
Verb phrase	VP
Noun phrase	NP
Prepositional phrase	PP
Adjectival phrase	ADJP
Adverbial phrase	ADVP

For dependency grammar, we employ an augmented set of labels from the Stanford Universal Dependency types [28] (SUD). The entire set is listed in Appendix B.2

All examples in the following sections will be illustrated visually by means of a constituency graph and a dependency graph with the corresponding node and edge labels. Every section will also contain a set of strings symbolizing the linear order of the key syntactic elements of each valid sentential and clausal pattern. These strings will be formed from the following symbols:

Stellenbosch University https://scholar.sun.ac.za

Table 4.3 Symbols indicating the linear order of syntactic elements

Symbol	**Description**
S	Subject
O	Direct object
I	Indirect object
V	Lexical verb
v	Auxiliary verb
A	Adjectival syntagm
e	Expletive
c	coordinate conjunction
s	subordinating conjunction
r	relative clause marker
ι	Infinitival clause marker
τ	Infinitival marker
η	Syntagm standing in first position in an inverted sentence

The constituency graphs presented in this chapter can only very loosely be called "Chomskyan generative". Binary branching is not considered an important constraint, nor are movement operations of any sort indicated. This is done to avoid the grammars of impractical complexity discussed in section 3.1 and allows us to adhere to Requirement 2.

Stellenbosch University https://scholar.sun.ac.za

4.1 Sentential patterns

Written Afrikaans can be considered a "well-behaved" V2 (verb second) language [8, p. 29]. This means that the first finite verb in a declarative sentence will always occupy second position regardless of tense or voice, and that the first position will usually be occupied by one of its nominal constituents. ("Sentence" in this case refers to an independent or superordinate clause; subordinate clauses in Afrikaans are verb final when introduced by an overt complementiser such as *dat* or *of*.)

From a syntactic perspective, three basic tenses are expressed in Afrikaans: present, past and future tense. Although some vestiges of seventeenth century Dutch remain in the language (such as a handful of verbs that still have imperfect forms [34, p. 222]) these do not contribute anything to the word order of Afrikaans sentences that merit separate investigation.

Each example sentence in the subsequent sections will be a variant of a single base sentence. Because each example will essentially contain multiple sentences with differing valencies, the main verb for this sentence will be one that can act both intransitively and transitively. The Afrikaans verb "gee" ("give") fits this bill, and therefore the base sentence will be the declarative[4]:

(7) Johan gee Jomari appels
Johan give Jomari apples
'Johan gives Jomari apples'

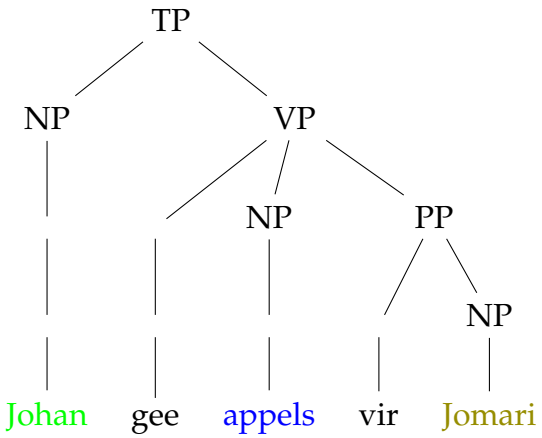
Permutations of this sentence will now be analysed, with part of speech tags excluded from graphs for simplicity's sake.

4.1.1 Declarative form – active voice

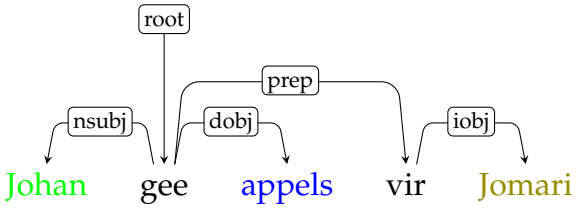
A declarative sentence in the present tense and the active voice exhibits SVO word order. A constituency tree for this sentence can be structured as follows:

[4] An example of the intransitive use of *gee* is *Johan gee altyd sonder om te kla* ("Johan always gives without complaining)

Stellenbosch University https://scholar.sun.ac.za



while its dependency graph is structured as illustrated below, with the main verb ("gee") acting as the root node of the graph:



In the examples above, the indirect object is headed or governed by a mandatory preposition, and the prepositional phrase in which it is contained occurs in fourth position. Should the indirect object be moved to third position – thereby displacing the direct object to the fourth position – this preposition becomes optional. This variation:

(8) Johan gee Jomari appels
1 2 3 4

is therefore grammatically valid and semantically equal to

(9) Johan gee vir Jomari appels
1 2 3 4

which is in turn equal to:

(10) Johan gee appels vir Jomari
1 2 3 4

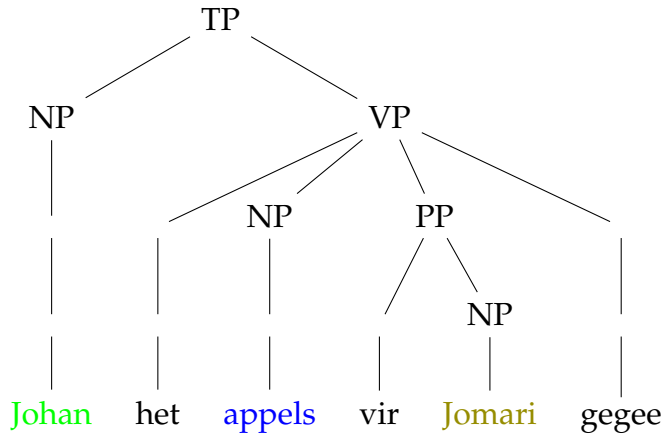
One could argue that, once headed by a proposition, the indirect object cedes its status as verbal argument to the prepositional phrase dominating it.

Stellenbosch University https://scholar.sun.ac.za

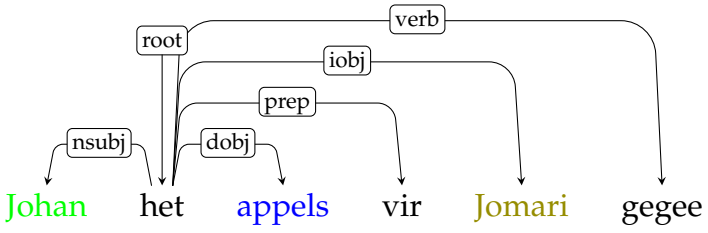
This will be set aside for the moment, as the examples in this section aim to compact multiple sentences into a single instance, and indicate the placement and movement of different syntactic constituents. For our purposes, whether the indirect object happens to find itself inside a prepositional phrase or not, it will be visually indicated as an indirect object.

When the declarative example in (7) is converted into past tense, its structure changes slightly. The aspectual auxiliary verb *het* ("have") becomes the finite verb in second position, with the main verb, in the form of the past participle *gegee* ("PAST-give"), occurring at the end of the sentence. The first constituent remains the grammatical subject, the direct object remains in the third and the indirect object in the fourth position.

In the case of the constituency tree, the auxiliary and main verbs are contained in the verb phrase, which in turn contains the direct and indirect objects in the form of noun and prepositional phrases:



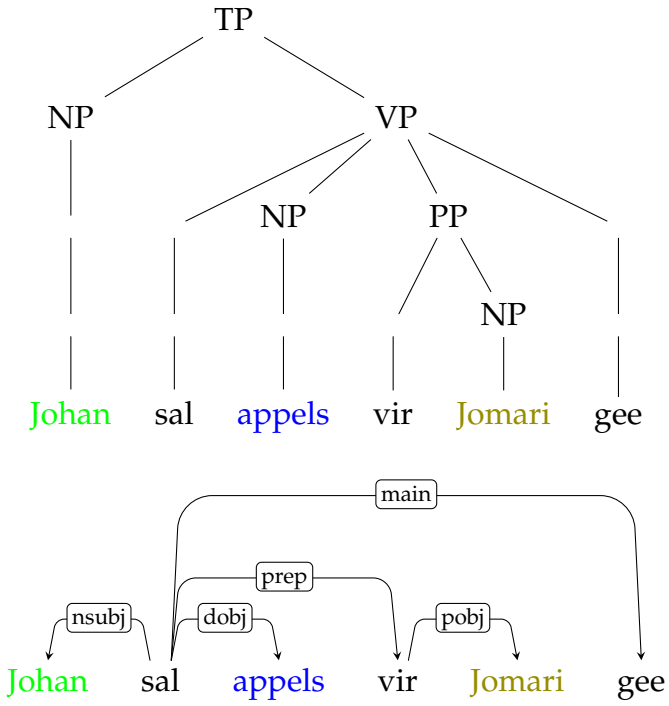
In the dependency representation of this sentence, the auxiliary verb is considered the root node of the sentence, and governs the infinitival main verb. This is in line with most of the literature on the subject [4] [48] [98], and fitting for a V2 language. The finite auxiliary therefore also governs the subject, direct object and indirect object:



Stellenbosch University https://scholar.sun.ac.za

For our purposes, the finite verb will always be considered the root of the sentence. The reason for this decision is explained in section 4.5.2. For now it is simply important to note that – because of this decision – we add a new edge label to the standard set of SUD labels. This label is called "verb", and will be used for all edges between finite auxiliary and infinitival main verbs as per the example above.

When converting the sentence to future tense, the same syntactic pattern found in the past tense example above is observable in both frameworks. The only difference between the two instances is the lexical difference between the finite and non-finite verbs (i.e. *het* versus *sal*, *gegee* versus *gee*). Insofar as the hierarchical structure of the sentence goes, everything remains the same:



So far we have illustrated the following valid patterns for Afrikaans sentences:

- SVOI
- SvOIV

Taking into account the possibility of object-switching, we also add the following variations:

Stellenbosch University https://scholar.sun.ac.za

- SVIO
- SvIOV

4.1.2 Interrogative form – active voice

Interrogative sentences in written Afrikaans are formed in two main ways. The first is by WH-movement, "a syntactic feature by which interrogative words appear at the beginning of an interrogative sentence or clause" [55, p. 5]. In Afrikaans these interrogatives include *wie* ("who"), *wat*, ("what"), *waar* ("where"), *hoe* ("how") and *hoekom* ("why"). Our base sentence (7):

"Johan gee appels vir Jomari"

can, for example, be converted into:

(11) wie gee appels vir Jomari
who give apples to Jomari
'who gives apples to Jomari'

This type of interrogative sentence will not be analysed further below, since from a computational perspective it is structurally equivalent to its non-interrogative counterpart: the various constituents remain in their original positions, with the finite verb still occurring in second position. The lexical content of the constituent in the first position differs (an interrogative instead of a nominal element), but this is a lexical matter which will not be examined in this study.[5]

Rather, we turn our attention to the second mode of question formation in Afrikaans: that of forming "Yes/No questions" by switching around the finite verb in second position with the first constituent of the sentence, as in:

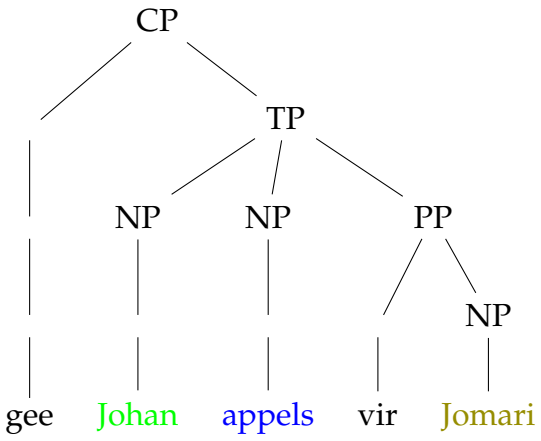
(12) gee Johan appels vir Jomari?
give Johan apples to Jomari?
'does Johan give apples to Jomari?'

The structure of these sentences is different from that of their declarative counterparts. In the constituency framework we use a node labelled *CP*

[5]One might be tempted to cite the presence of a question mark, but this is an orthographical matter.

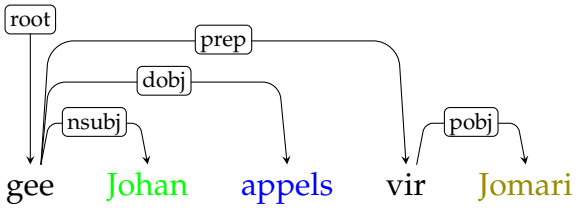
Stellenbosch University https://scholar.sun.ac.za

("complementiser phrase") to contained the preposed verb, while the TP contains the subject, direct object and PP containing the indirect object:



Strictly speaking, from the perspective of generative grammar, the TP should contain a VP. In turn, this VP should contain the verb and the objects, while the subject remains in the TP. Given the constraints of context free grammars we refrain from adding this VP, since the position of the verb would require crossing graph edges. The alternative - adding the VP to the TP and only having it contain the objects - is equally unattractive from a practical point of view, since it would result in a parser predicting VPs without any verbs. For our purposes, we resort to using only a TP to contain the non-preposed elements in a sentence such as (12), while acknowledging this as a known limitation of our annotation framework.

In the dependency representation of this sentence – where constituents are not nested into one another to begin with – the only change is in word order:

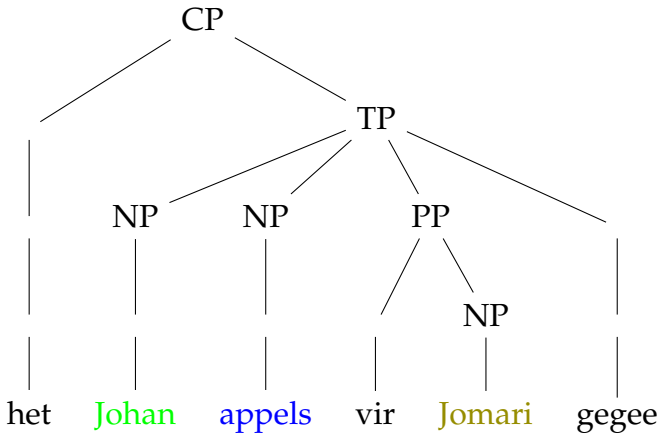


An important characteristic of dependency grammars now becomes apparent. While question-forming changes the hierarchy of a constituency structure, the hierarchy of a dependency structure remains intact. That is because even though word order changes, the root of the sentence and the binary relations

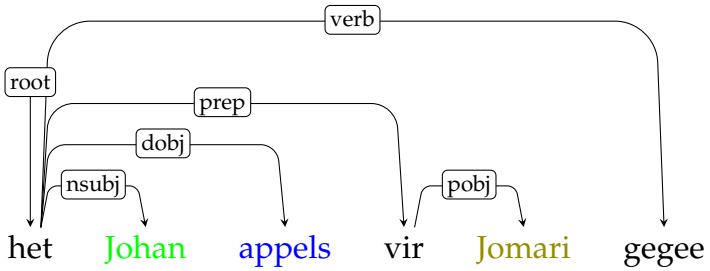
Stellenbosch University https://scholar.sun.ac.za

between vertices (words) do not change. All dependents remain governed by the same governers as would have been the case in a declarative sentence.

Converting the tense of an interrogative sentence from present to past tense slightly increases the complexity of the structure, but does not radically change the order of the arguments. The finite auxiliary verb (which occupies the second position in the declarative sentence) is preposed and move ahead of the subject, while the main verb remains at the end of the sentence. The rest of the structure remains unchanged:

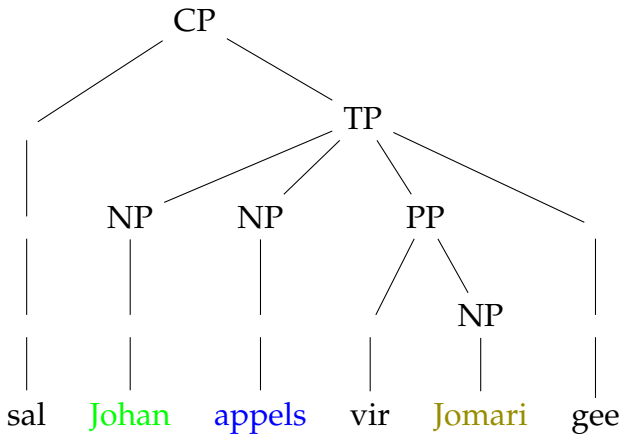


The dependency representation, again, keeps the original hierarchy found in its declarative equivalent intact:

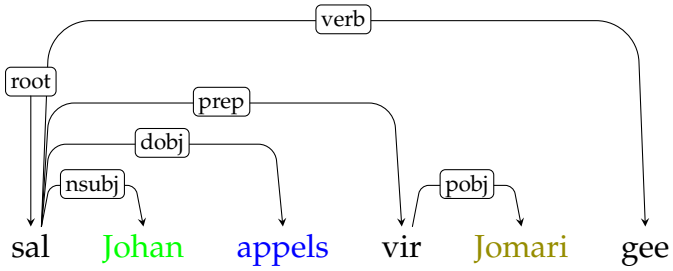


When converting the sentence from past to future tense, a recurring phenomenon can be observed: although lexical changes are visible, the order of constituents remains the same:

Stellenbosch University https://scholar.sun.ac.za



This also holds for the dependency representation:



Having examined interrogative base sentences in the active voice, we can add the following patterns to our set:

- VSOI
- vSOIV

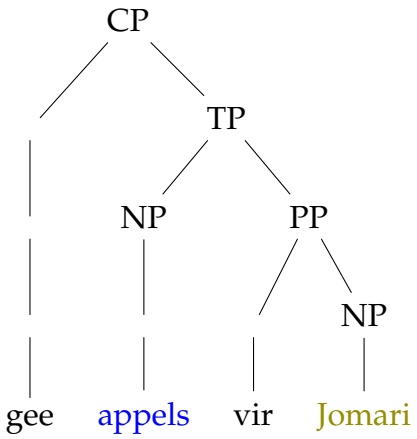
as well as the following object variations:

- VSIO
- vSIOV

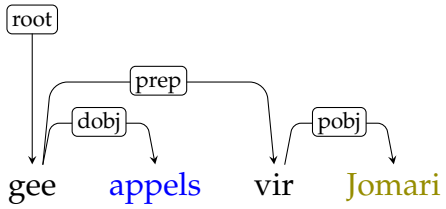
4.1.3 Imperative form – active voice

The imperative form in Afrikaans is formed by omitting the subject from the structure normally associated with a simple declarative sentence [92]. For our purposes, its constituency structure looks as follows:

Stellenbosch University https://scholar.sun.ac.za



It's dependency structure looks like that of a declarative sentence without a subject:



It is optionally possible to include the addressee in the sentence:

(13) Johan, gee appels vir Jomari
Johan, give apples for Jomari
'Johan, give apples to Jomari

In this case, the constituency and dependency structures would be the same as in the declarative sentence.

We can therefore add the following pattern and variation to our set:

- VOI
- VIO

4.1.4 Declarative form – passive voice

The passive voice in Afrikaans can be described in terms of the following main features or steps:

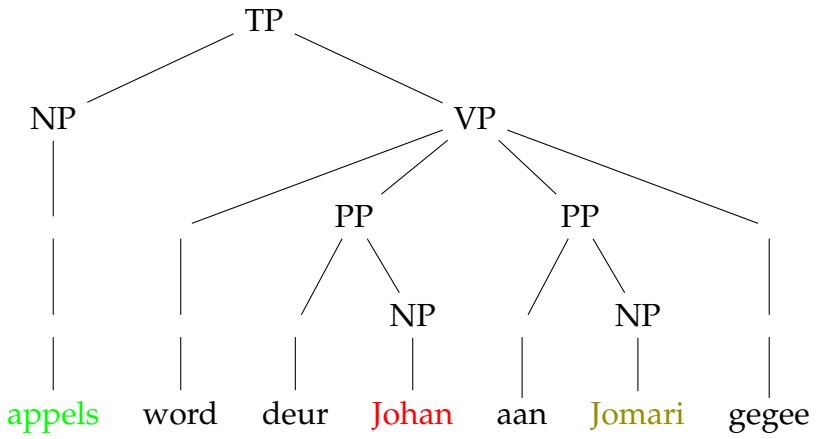
Stellenbosch University https://scholar.sun.ac.za

1. The main verb takes the form of a passive participle, and has to co-occur with a passive aspectual auxiliary verb (i.e. different forms of *wees* (“be”), such as *word* (“is”/“is being”), *is* (“was”), *was* (“had been”); in the absence of any other auxiliary verbs, e.g. modal auxiliaries, the aspectual auxiliary serves to express the tense.

2. The direct object is moved to sentence initial position, thereby turning it into the structural subject of the new construction.

3. The agent (i.e. the entity represented by the subject in the corresponding active construction) forms part of a prepositional phrase headed by the preposition *deur* (“by”); this PP occurs in the position where the direct object would be found in the active construction. (The agent *by*-phrase can also be omitted, resulting in an agentless passive.)

To illustrate, consider the following example which doubles as the constituency representation in this section:



In this sentence the finite verb of the active base sentence (*gee*) occurs in the passive participle form *gegee* (“given”) and co-occurs with the passive auxiliary *word* (“is being”), as per step 1. The direct object (*appels*) is shifted to sentence initial position, taking on the role of structural subject, as per step 2. The semantic subject (i.e. the NP *Johan*) forms part of a prepositional phrase indicating the thematic agent of the sentence (*deur Johan*).

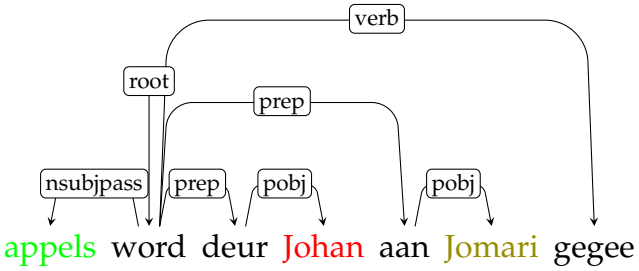
While a constituency parse can employ the same non-terminal labels for the passive voice, this approach does not translate well to the edge labels of a dependency based representation. Whereas it is quite possible to label

Stellenbosch University https://scholar.sun.ac.za

the new grammatical subject as "nsubj", it makes little sense to use the label "dobj" as, syntactically speaking, there is no direct object to refer to.

The question of what happens to the subject of the corresponding active construction does not have a clear answer. Chomsky originally claimed that it becomes part of a manner adverb [22]. Perlmutter and Postal argue that it is "demoted" to a chômeur [85, pp. 409] – an element that is not part of the nucleus of the clause[6]. McCawley states that it is common practice to refer to it as an "agent", but avoids this approach himself [66], stating that it may not be represented overtly at all, in which case the original subject was an abstract thing called "UNSPEC" ("unspecified"). As different as these and other approaches are, they have one thing in common: they consider the former subject to be on the periphery of the clause.

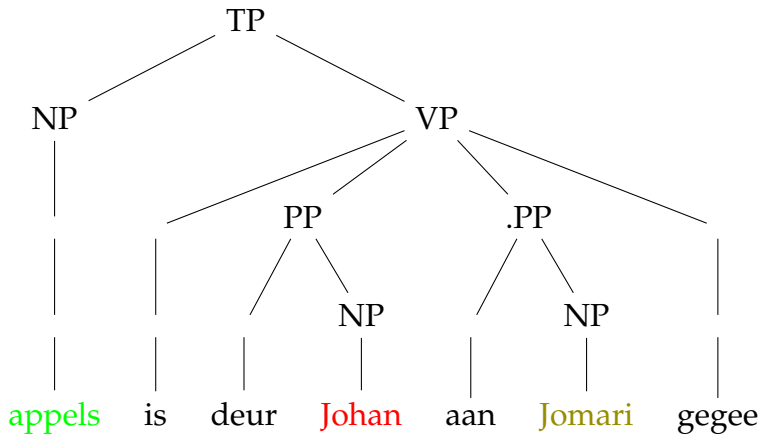
For our purposes it is important that a decision on nomenclature is reached, specifically from the perspective of dependency grammar. Taking a queue from SUD, we will employ the label *nsubjpass* ("passive nominal subject"). The remainder of what were nominal arguments in the declarative voice become prepositional objects.



Much like previous examples, the structure of declarative sentences in the passive voice remains the same when converted to past tense:

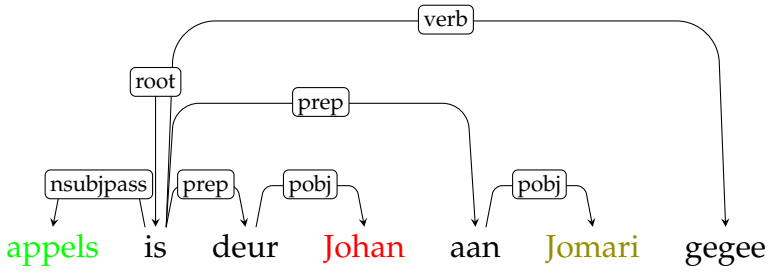
[6] 'Chômeur' literally means 'unemployed' in French.

Stellenbosch University https://scholar.sun.ac.za



The only difference between these two forms is the string of characters used for the auxiliary verb, which changes from *word* ("be") to *is* ("is"). (This is, again, a lexical difference, and not relevant to word order.)

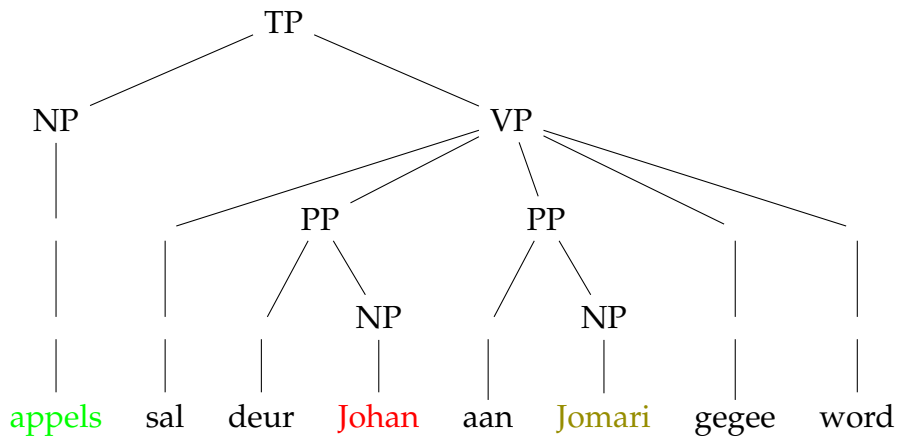
The dependency graph for the same sentence follows the same pattern as its present tense version:



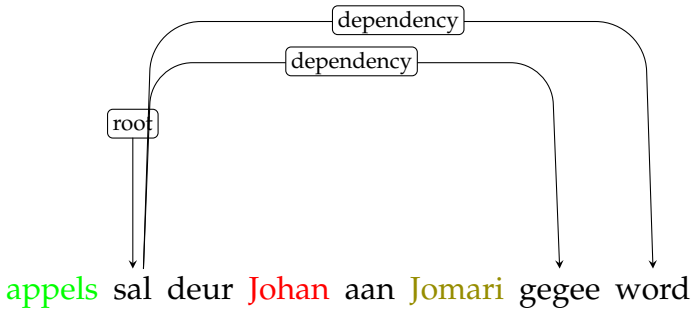
In both frameworks, no structural difference between present and past tense versions of a sentence is apparent. The non-terminal categories of the constituency grammar does not change, nor the edge labels of the dependency grammar.

This situation changes when the sentence is converted to future tense, with the modal auxiliary verb "sal" added as tense marker. The constituency representation in this case looks as follows:

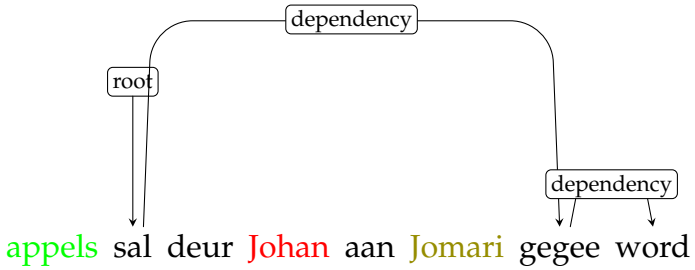
Stellenbosch University https://scholar.sun.ac.za



The dependency framework presents additional complexity to be resolved in the form of the new auxiliary verb that now requires a governor. Assuming the finite verb as root, it is not immediately clear whether the graph should be:



or whether it should look as follows:



This issue will be addressed in section 4.5.2 of this chapter.

We have added two patterns to our set:

- OvSIV
- OvSIVv

Stellenbosch University https://scholar.sun.ac.za

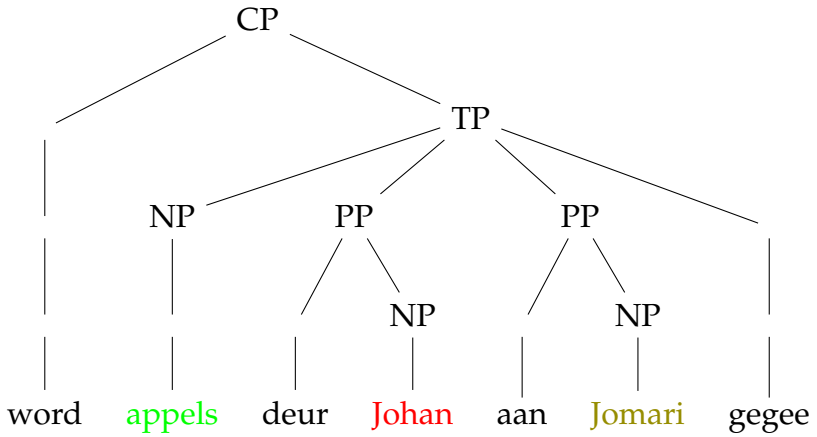
And two object variations:

- OvISV
- OvISVv

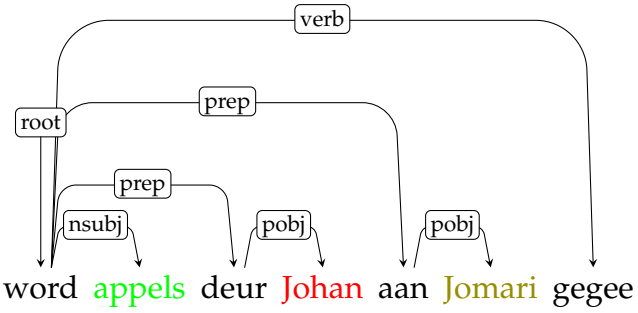
In these patterns, the symbol "S" indicates where the semantic subject has been moved during the change of voice. Similarly, "O" indicates where the original object has moved. Neither of these symbols in these and subsequent passive examples indicates actual grammatical roles.

4.1.5 Interrogative form – passive voice

Yes/no interrogative sentences in the passive voice contain the same phrasal components as their declarative counterparts. Here, again, word order signals that a question is being asked. This change in order alters the structure of the constituency graph:

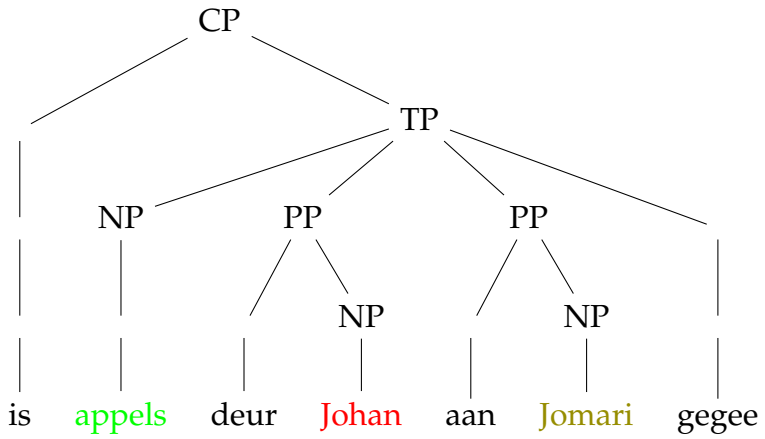


The dependency relations, however, remain identical to those found in the corresponding declarative form:

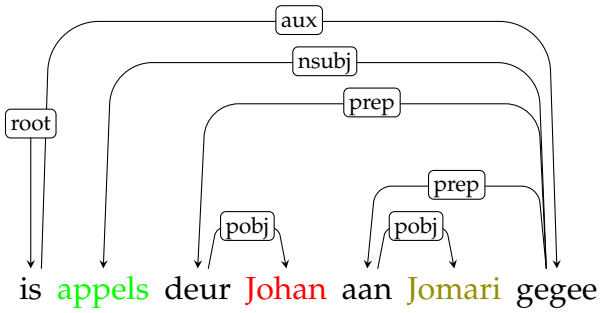


Stellenbosch University https://scholar.sun.ac.za

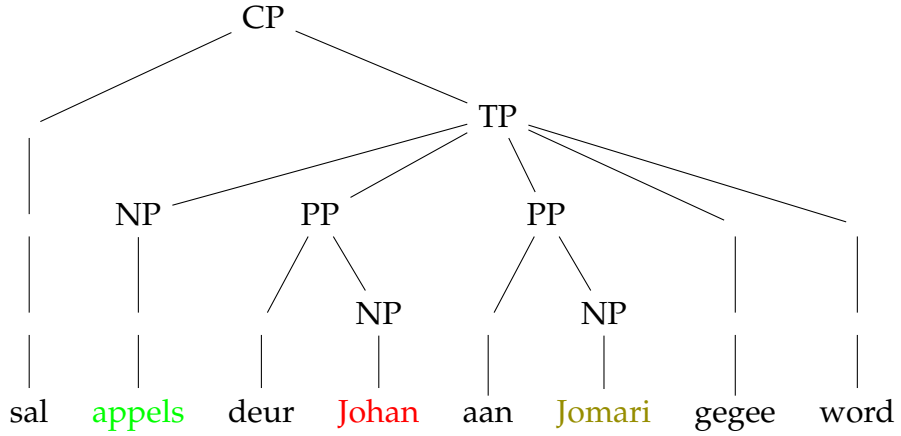
Converting the sentence into past tense results in the same basic constituency structure as in the present tense version:



The same applies to its dependency graph:

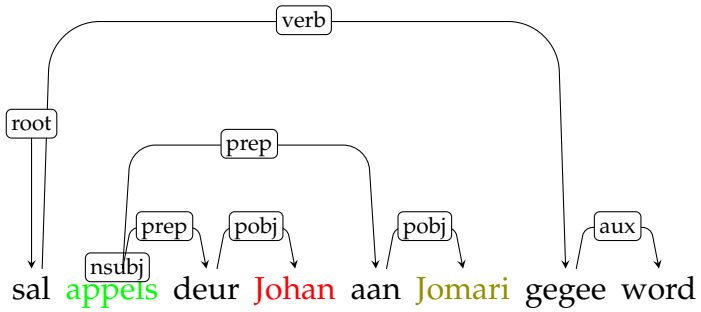


Converting the sentence to future tense results in the following constituency structure:



Stellenbosch University https://scholar.sun.ac.za

The dependency representation, again, retains the same form as its declarative counterpart: all vertices in the graph are connected to each other in the same fashion, the only difference between them being linear word order:



We can therefore add the following additional base patterns:

- vOSIV
- vOSIVv

as well as the following object variations, where the chômeur is positioned after non-finite verbs:

- vOIVS
- vOIVvS

4.1.6 Imperative form – passive voice

The imperative form in the passive voice in Afrikaans is formed with the use of the verbal elements:

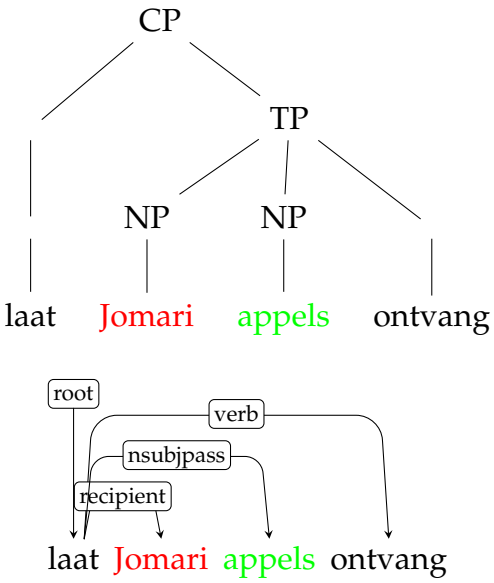
laat (“let”) + passive participle + *word*

as in:

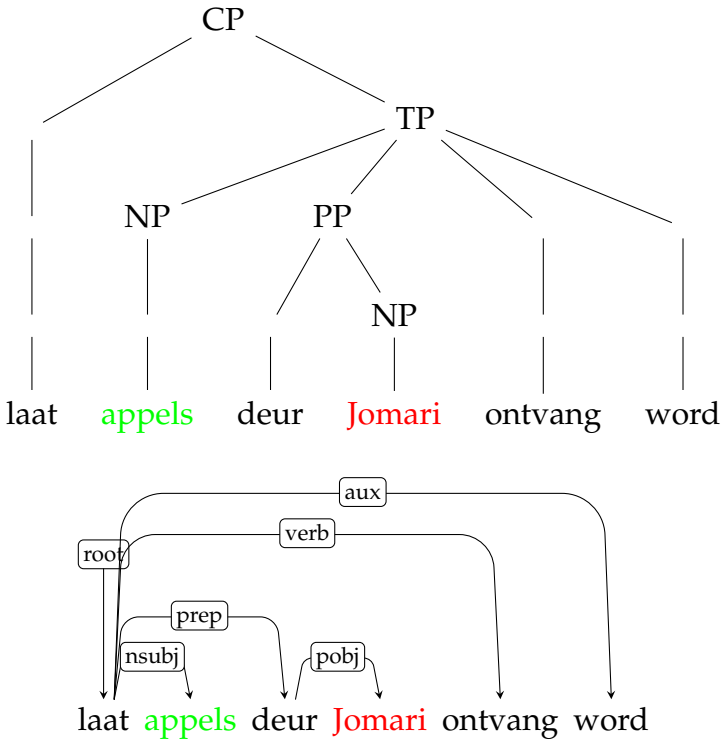
(14) laat Jomari appels gegee word (deur Johan)
let Jomari apples PAST-give be (by Johan)
let Jomari be given apples (by Johan)

Here the auxiliary verb *laat* (“let”) is combined with a passive participle and the passive aspectual auxiliary verb *word* (“be”). We illustrate two variations. The first variation is as follows:

Stellenbosch University https://scholar.sun.ac.za



The second variation is:



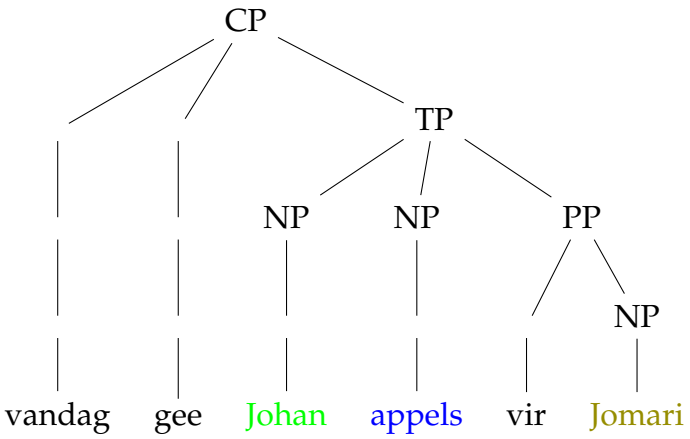
We therefore add the following patterns to our set:

- vOIV
- vIOVv

Stellenbosch University https://scholar.sun.ac.za

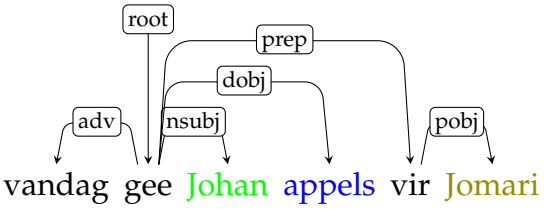
4.1.7 Inverted sentences: active voice

Other inversions besides those used to construct interrogative sentences are possible in Afrikaans. Phrases indicating, amongst others, time, manner and place – along with the finite verb – can be moved in front of the subject. Consider the following constituency tree:



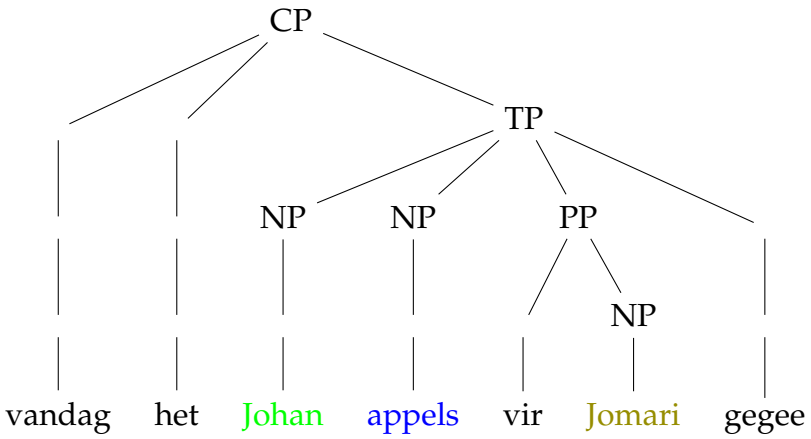
In this example, an adverb of time *vandag* ("today") and the finite verb *gee* ("give") are moved in front of the subject *Johan*.

The dependency parse exhibits the same basic structure as it would have had in a non-inverted declarative sentence:

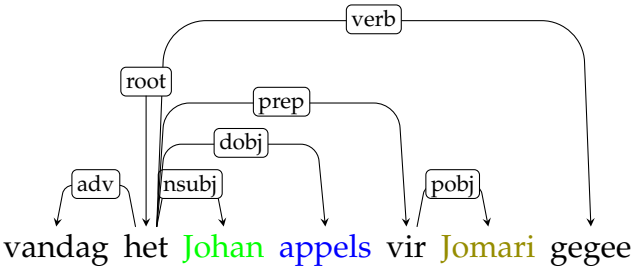


In the case of a conversion to past tense, the main verb occurs in sentence-final position, with the finite auxiliary verb *het* in the second position. The rest of the constituents remain in the same positions:

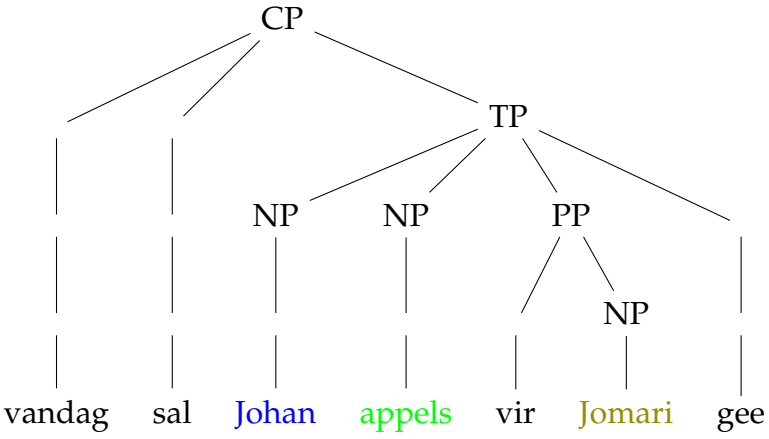
Stellenbosch University https://scholar.sun.ac.za



This is also the case for the dependency graph:

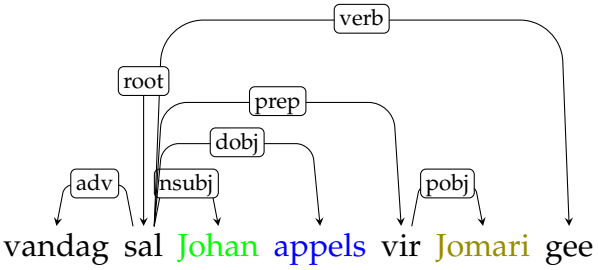


Conversion to future tense follows the same syntactic behaviour with the usual lexical changes:



and:

Stellenbosch University https://scholar.sun.ac.za



We therefore add the following patterns to our set, using the symbol η to signify any of the allowable syntagms that can take the place of the subject in an inverted sentence:

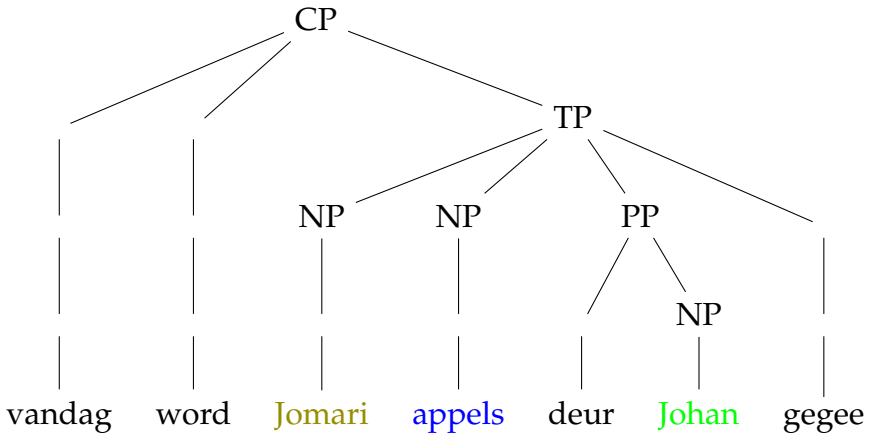
- ηVSOI
- ηvSOIV

We also include the following object-variations:

- ηVSIO
- ηvSIOV

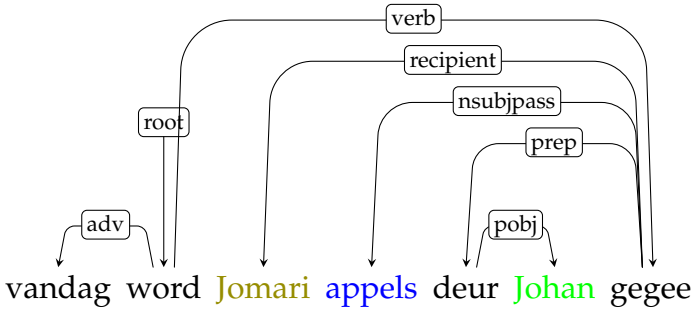
4.1.8 Inverted sentences: passive voice

Afrikaans sentences in the passive voice are able to undergo the same inversion described in the previous section. In the present tense, the constituency structure of our base sentence looks as follows:

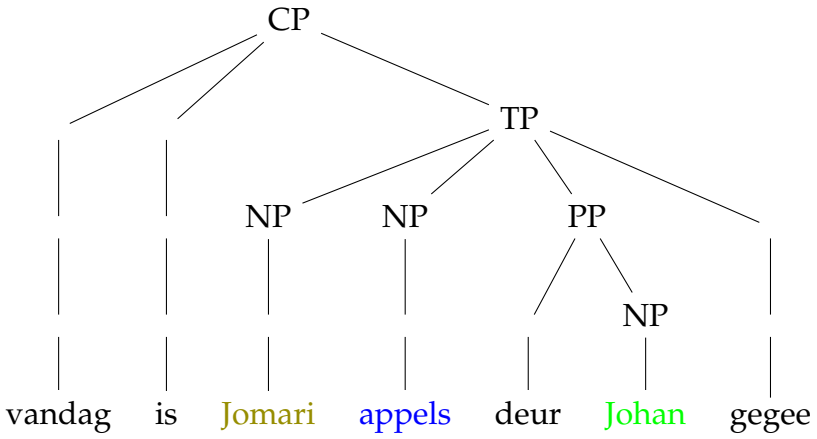


Its dependency structure looks as follows:

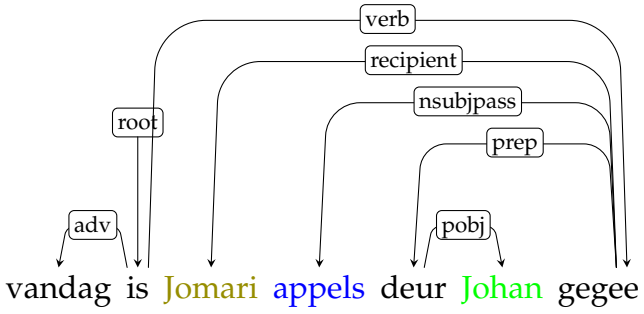
Stellenbosch University https://scholar.sun.ac.za



Conversion to past tense again yields the same syntactic hierarchy with minor lexical changes:

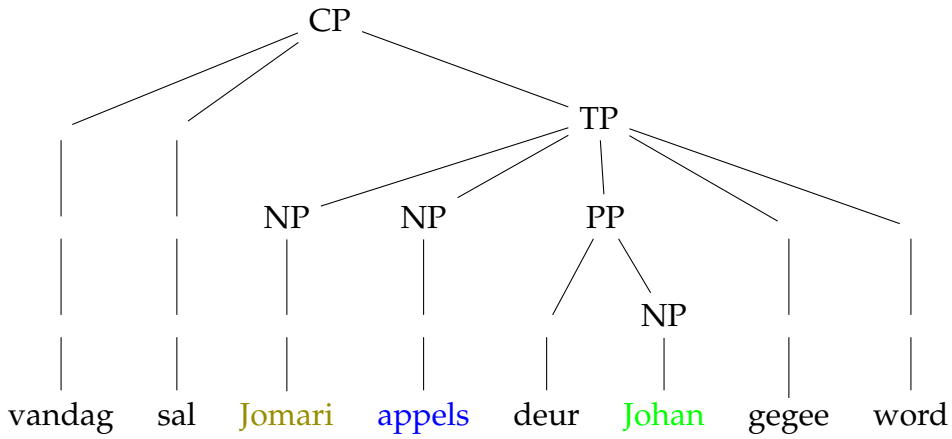


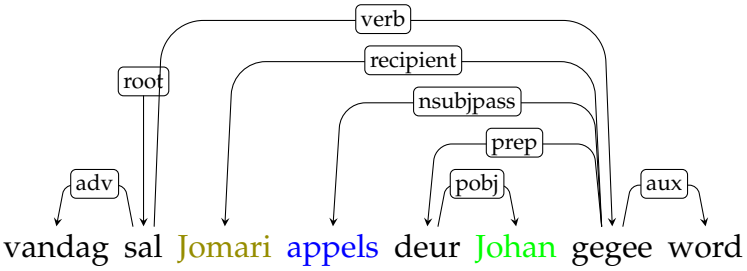
This also holds for the dependency structure:



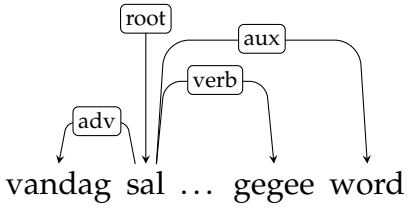
In the case of a conversion to future tense, additional complexity (due to the additional auxiliary verb) is visible in both constituency and dependency structures:

Stellenbosch University https://scholar.sun.ac.za





In the previous dependency graph a syntactic ambiguity crops up again: the multitude of ways in which verbs can be connected to one another.[7] By convention found in the literature, the modal auxiliary *sal* ("will") which serves to indicate future tense is designated the root node of the graph. It is not immediately clear, however, how the second auxiliary verb should be treated: as a dependent of the finite root or the infinitival main verb, which would result in this alternate parse:



This complexity will be disambiguated in section 4.5.2. For now, we can add the following valid base patterns and variations:

- ηvIOSV

[7] This ambiguity exists specifically because of Afrikaans' morphological impoverishment, which forces the language to indicate aspects such as tense and voice analytically.

Stellenbosch University https://scholar.sun.ac.za

- ηvIOSVv
- ηvOISV
- ηvOISVv

4.1.9 Declarative copular sentences

In the context of this study, the expression "copular sentences" refers to sentences where the main verb is a copular verb, and where qualities or attributes are projected onto the subject, for example by an NP:

(15) Johan was 'n student
Johan PAST-be a student
'Johan was a student'

an AdjP:

(16) Johan is volleerd
Johan be full-learned
'Johan knows everything'

a PP:

(17) Johan is in die skuld
Johan be in the debt
'Johan is in debt"

or a clause:

(18) Johan se skuld is wat mens kry as jy te lank studeer
Johan POSS debt be what one get if you too long study
'Johan's debt is what you get if you are a student for too long'

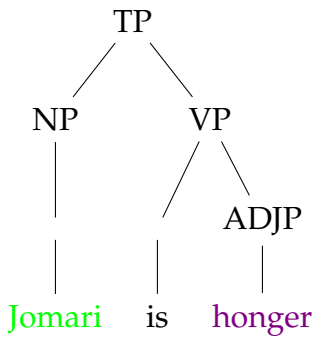
Although their structures are like those of previous examples, copular sentences are singled out for discussion here because of a parsing difficulty encountered in their dependency structures.

While the examples below make use of single adjectives, it should be noted that they are merely a subset of a broader set of permissible syntactic constituents in Afrikaans that can play the role of assigning qualities to nominal constructions. Furthermore, a different base sentence than (7) is required. We will use:

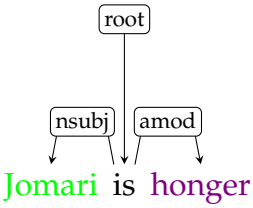
Stellenbosch University https://scholar.sun.ac.za

(19) Jomari is honger
Jomari be hungry
'Jomari is hungry'

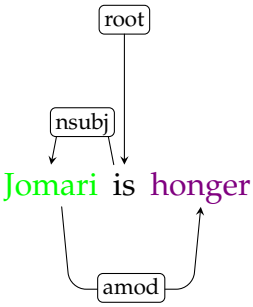
Starting with this simple declarative copular sentence in the present tense, the structural similarities to non-copular declarative sentences are immediately apparent, with a subject-predicate relationship clearly visible in the constituency structure:



The dependency parse runs into some uncertainty. While the behaviour of the copular verb and the subject in this framework remains the same, it is not entirely clear which governor should be connected to the adjectival head: the copular verb, or the noun which it technically modifies? In other words, it is not immediately clear whether this dependency parse:

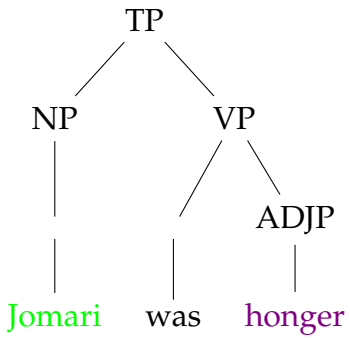


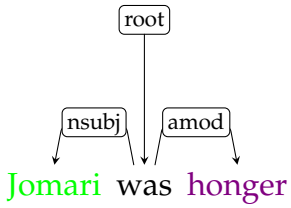
should be preferred over this one:



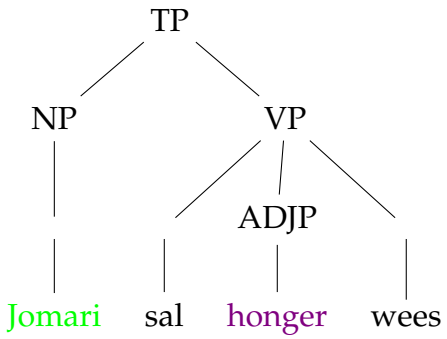
Stellenbosch University https://scholar.sun.ac.za

This question will be addressed in section 4.5.3. For the remainder of this chapter, it will be assumed that the adjectival syntagm is governed by the root node, as indicated by the first dependency graph above. Therefore, converting the sentence to past tense results in identical patterns for both constituency and dependency formalisms:



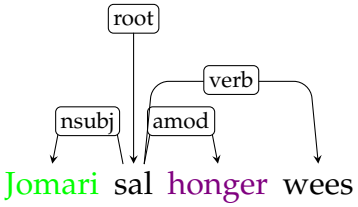


For the slightly more complex future tense, where an additional modal auxiliary verb is added, the following pattern applies in constituency grammar:



and the following in dependency grammar:

Stellenbosch University https://scholar.sun.ac.za

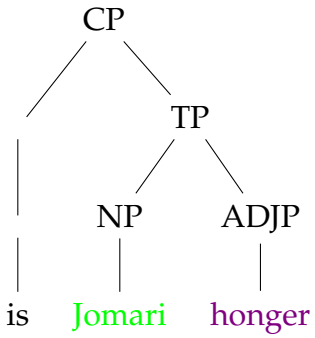


The following patterns can now be added to our list, using the symbol A to indicate adjectival syntagms:

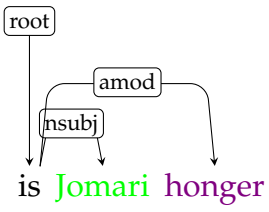
- SVA
- SvAV

4.1.10 Interrogative copular sentences

Question formation in copular sentences occurs in the same manner as in previously discussed sentences, with the finite verb preposed in front of the subject. In constituency grammar we analyse this as follows:

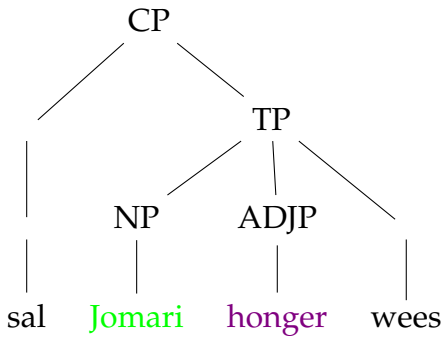


The dependency parse, again, allows for the same ambiguity as encountered in the declarative form. Continuing with our assumption that the adjective connects to the root node, the dependency parse for this sentence looks as follows:

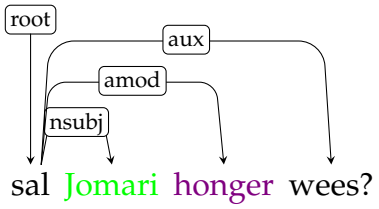


Stellenbosch University https://scholar.sun.ac.za

The slightly more complicated future tense can be represented as follows in constituency grammar:



And its corresponding dependency graph will take the following form:



The following patterns can accordingly be added to our list:

- VSA
- vSAV

4.1.11 Copular sentences: passive voice

A copular sentence in the passive voice is a nonsensical concept. Copular sentences do not have direct objects, and also no constituent that could play the role of the "agent" in a passive conversion. No patterns from this category are therefore added to our list.

4.1.12 Declarative expletive sentences

Expletive sentences are characterised by the presence of syntactic expletives – semantically empty words that fulfil syntactic or other functions in a sentence. One of the common functions of expletives in Afrikaans is to indicate the existence of an idea or object, or the occurrence of an event. These functions

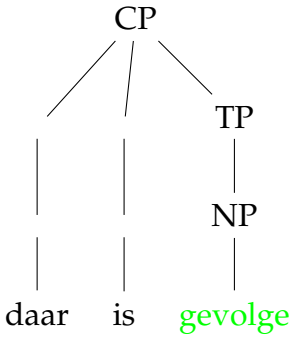
Stellenbosch University https://scholar.sun.ac.za

are fulfilled by the expletives *daar* ('there') and *dit* ('it'), as illustrated in the examples below:

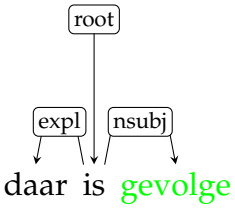
(20) daar is gevolge
there be consequences
"there are consequences"

(21) dit reën
it rain
"it rains"

In the case of constituency grammar, an expletive sentence in the present tense can be analysed as follows:

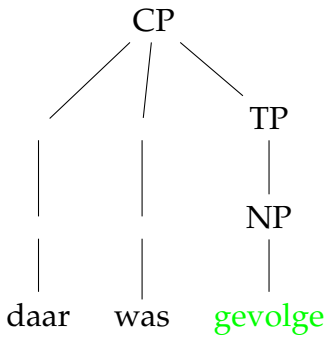


The dependency graph for this sentence is:

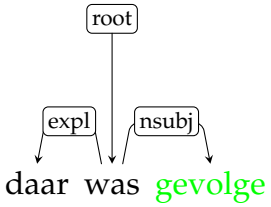


Conversion of the sentence to past tense requires a lexical change, while the underlying syntactic pattern remains the same. This holds true for the constituency parse:

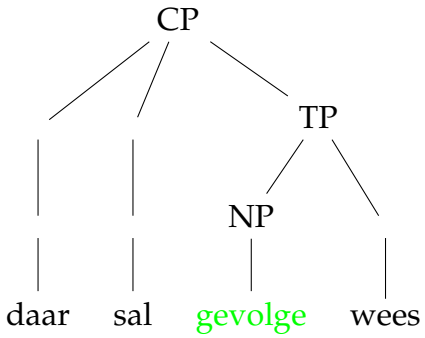
Stellenbosch University https://scholar.sun.ac.za



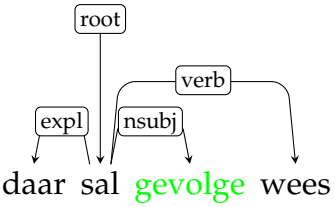
as well as the dependency parse:



Conversion to future tense, again, follows a similar pattern as previous examples, owing to the obligatory use of an auxiliary verb.



The dependency representation also expands slightly to account for the extra node in the graph:



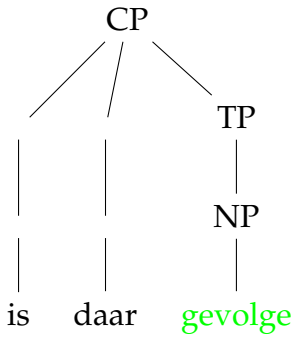
We therefore add the following patterns to our set:

Stellenbosch University https://scholar.sun.ac.za

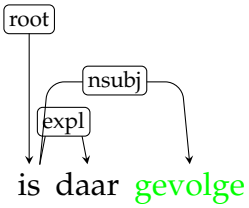
- eVS
- evSV

4.1.13 Interrogative expletive sentences

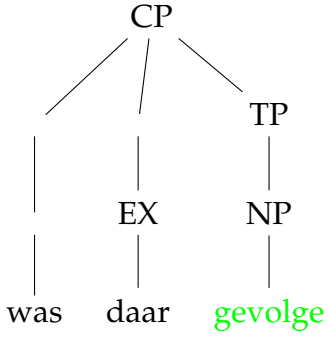
Inversion can be used to convert each of the declarative examples in the prior section to interrogative sentences. As in prior examples, the TP in our constituency representation of these structures contains the element that used to stand in first position:



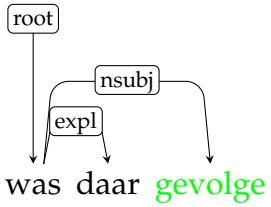
Apart from word order, the logical structure of the dependency parse is the same as its declarative counterpart:



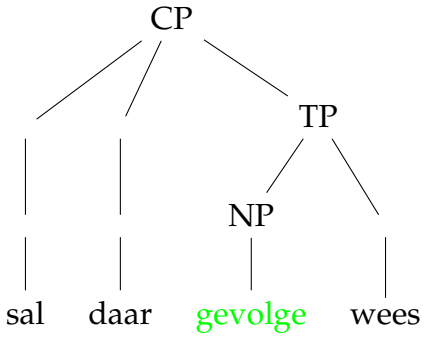
Conversion to past tense results in identical constituency and dependency parses with a lexical change:

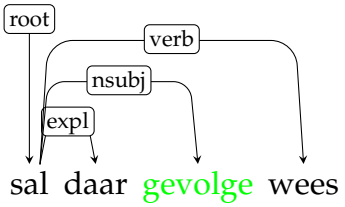


Stellenbosch University https://scholar.sun.ac.za



Conversion to future tense changes these parses as follows:





This allows us to add the following patterns to our set:

- VeS
- veSV

4.1.14 Expletive sentences: passive voice

For our purposes, expletive sentences in the passive voice exhibits the same pattern as those in the active voice, as in the following examples:[8]

(22) daar is gedans (deur die kinders)
there be PAST-dance (by the children)
'There was dancing (by the children)'

[8] A word of thanks to Dr Johan Oosthuizen from Stellenbosch University for this information.

Stellenbosch University https://scholar.sun.ac.za

(23) dit word beweer (deur die opposisie) dat...
it be claim (by the opposition) that...
'it is claimed (by the opposition) that...'

Sentences like these are therefore not discussed in more detail, nor do they contribute to the set of sentential patterns.

4.1.15 Base patterns

The patterns identified in the sections above constitute the majority of combinations in which syntactic constituents can be used to form main clauses in written Afrikaans. We therefore define them as members of the non-terminal set C_m, where:

C_m = { SVO, SvOV, VSO, vSOV, VOI, OvSIV, OvSIVv, vOSIV, vOSIVv, vOIV, ηVSOI, ηvSOIV, ηvIOSV, ηvIOSVv, SVA, SvAV, VSA, vSAV, eVS, evSV, VeS, veSV, SVIO, SvIOV, VSIO, vSIOV, VIO, OvISV, OvISVv,vOIVS, vOIVvS, vIOVv, ηVSIO, ηvSIOV, ηvOISV, ηvOISVv }

and add them to the non-terminal superset C:

$$C = \{C_m\}$$

4.2 Clause structures

Written sentences rarely consist of single clauses. Afrikaans permits a variety of clauses to be added to main clauses or other constituents by means of coordination and subordination. We discuss both these phenomena in this section, and furthermore make a distinction between three types of subordinate clauses:

Stellenbosch University https://scholar.sun.ac.za

Table 4.4 Types of subordinate clauses

Type	Description
Finite subordinate clauses	Clauses introduced by subordinate clausal markers
Infinitival clauses	Clauses introduced by the infinitival marker *om*
Relative clauses	Clauses introduced by relative pronouns (*wat*, *wie*, *waarheen*, *waarna* et cetera)

These clauses are linked to or embedded in main clauses or other phrases to form compound sentences. Depending on the formality and register of a written text, these clauses can also stand on their own.

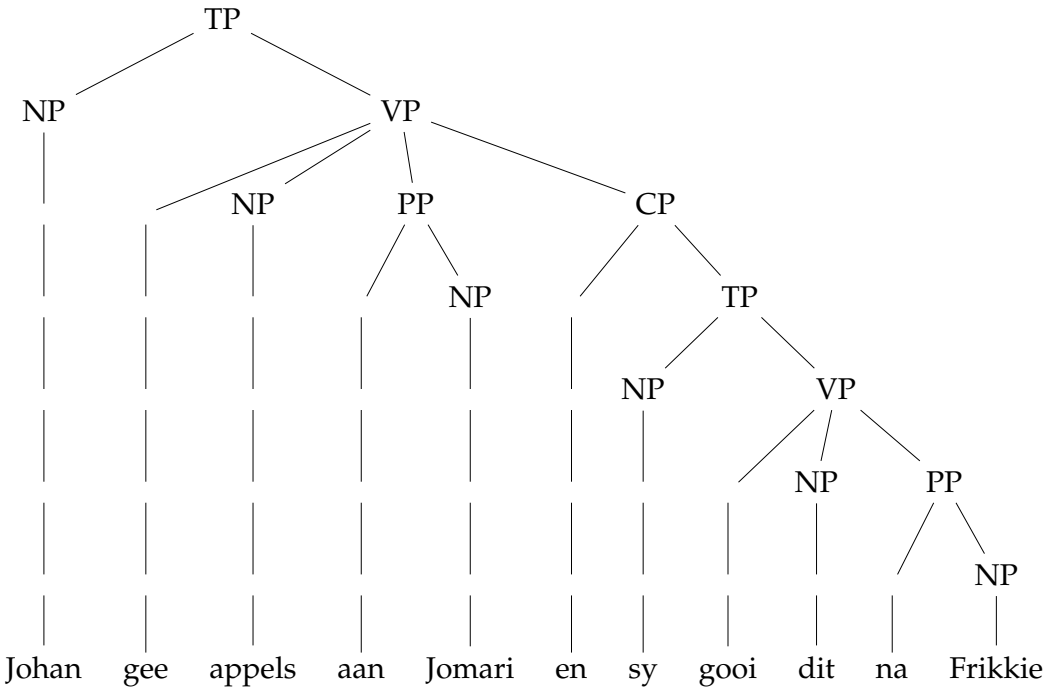
4.2.1 Coordinate Clauses

Coordinated clauses are clauses that are formed by means of coordinate conjunctions such as *en* (“and”), *of* (“or”), *maar* (“but”) and *want* (“because”), as in the following example:

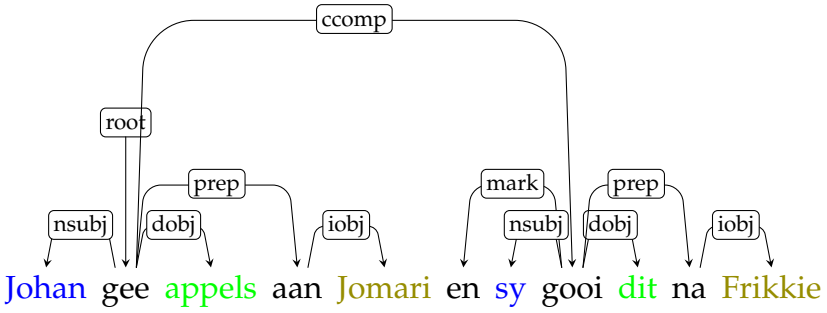
(24) Johan gee appels aan Jomari en sy gooi dit na Frikkie
Johan give apples to Jomari and she throw it to Frikkie
‘Johan gives apples to Jomari and she throws them at Frikkie’

Each clause has the same basic structure. We include the coordinator *en* within the second clause, resulting in the following structure:

Stellenbosch University https://scholar.sun.ac.za



The decision to place the coordinator under the second CP is taken to maximize equivalence with the dependency representation, in which the finite verb in the first clause governs the finite verb in the second clause, and *en* is indicated as clausal marker:



As can be observed from the example above, the structure of the second conjunct does not change in terms of word order or hierarchy.

Because no transformations of existing sentential patterns are required to form coordinate clauses in Afrikaans, and because each of the base patterns in C_m can be prefixed with a coordinate conjunction to turn it into a coordinate clause, we can define the set of valid coordinate clauses in Afrikaans as C_c:

$$C_c = \{ cS_m \}$$

Stellenbosch University https://scholar.sun.ac.za

where $_c$ is the set of all allowable coordinate conjunctions in Afrikaans, consisting of the following terminals:

$$\text{c} = \{\ \textit{of}, \textit{en}, \textit{maar}, \textit{want}\ \}$$

Since zero or more coordinate clauses can be appended to an Afrikaans sentence, we extend our definition of *C* as follows:

$$\text{C} = \{\text{C}_m S_c *\}$$

4.2.2 Finite subordinate clauses

Afrikaans finite subordinate clauses, specifically those that are introduced by an overt complementiser (i.e. a subordinate conjunction), have a different word order than that of the base pattern (7). Yes/no question formation is ruled out in subordinate clauses, and clauses must adhere to the verb final pattern. Whereas a basic declarative sentence (containing a single verb) in Afrikaans follows the following pattern:

SVOI

A subordinate clause with an overt complementiser changes this order by requiring the verb to be in sentence final position, thereby changing the order to:

SOIV

As is the case with coordinate conjunctions, subordinate conjunctions form a finite set of terminal strings. In Afrikaans, they are:

s = {*aangesien*, *as*, *asof*, *behalwe*, *dan*, *dat*, *derhalwe*, *deurdat*, *hoewel*, *mits*, *nadat*, *nog*, *of*, *ofskoon*, *omdat*, *sedert*, *sodat*, *sodra*, *sonder*, *soos*, *tensy*, *terwyl*, *toe*, *totdat*, *voordat*}

In cases where coordinate deletion[9] inside a subordinate clause causes a subordinate conjunction to be deleted, we treat the coordinate conjunction as

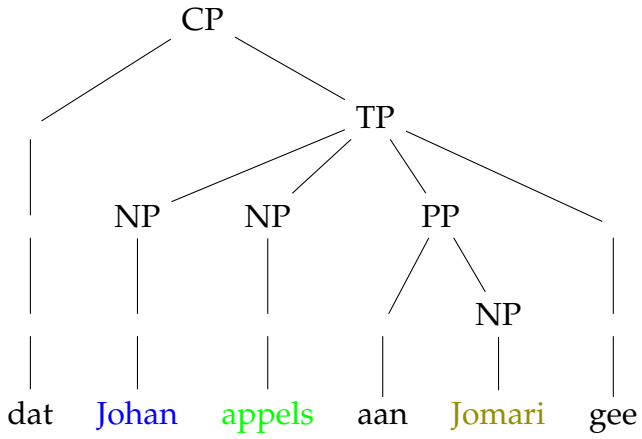
[9]The deletion of constituents in a coordination "in which each conjunct includes a constituent which is identical to the corresponding constituent of each other conjunct" [54, p. 347]

Stellenbosch University https://scholar.sun.ac.za

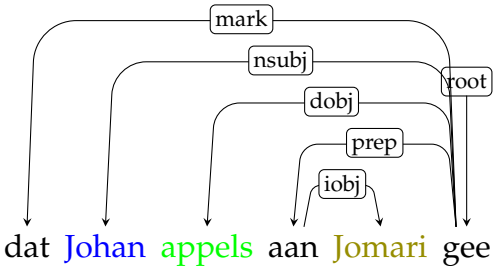
the clause marker. This can be illustrated by means of the following example, in which the coordinate conjunction *en* has been included in the subordinate clause (marked by curly brackets):

(25) Hy weet dat die hond blaf {en die kat miaau}
He know that the dog bark {and the cat meow}
'He knows that the dog barks {and the cat meows}'

The constituency representation of a subordinate clause in the present tense and the active voice looks as follows:



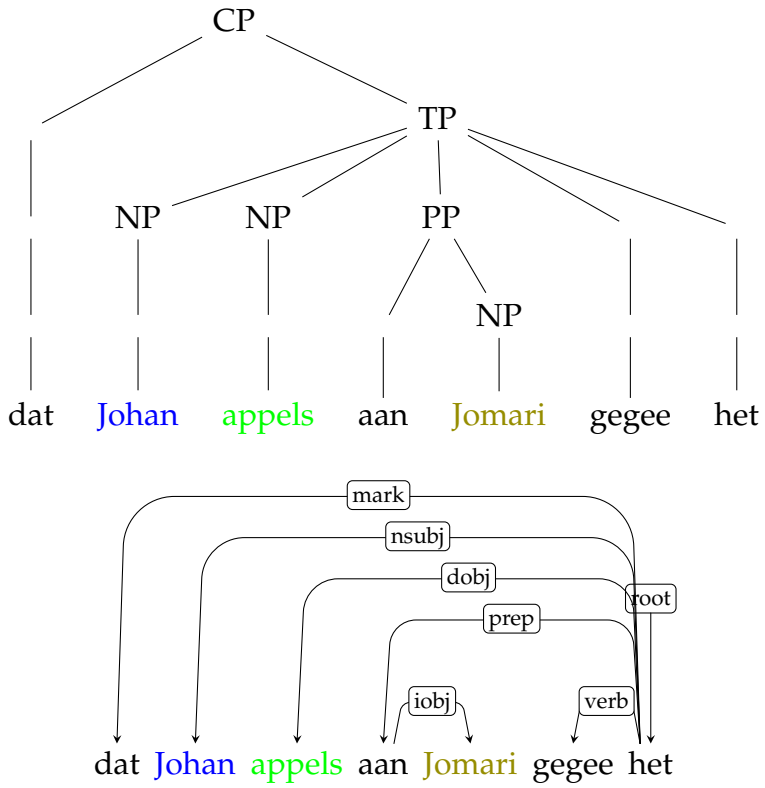
Its dependency graph is:



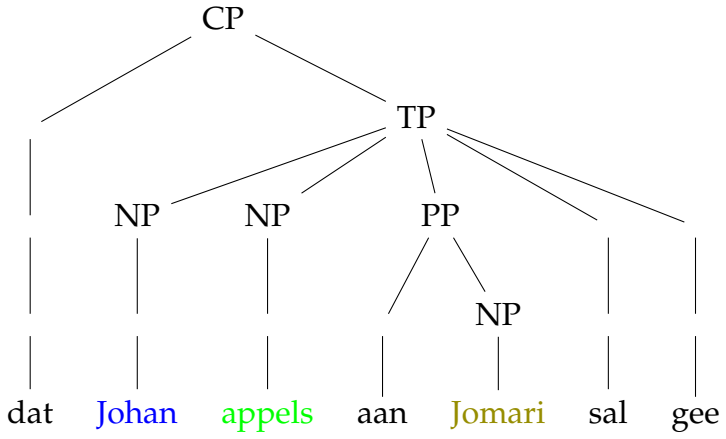
Conversion to past tense yields the following graphs[10]:

[10]Note how, in keeping with decisions made in prior sections, the finite verb remains the root of the dependency graph.

Stellenbosch University https://scholar.sun.ac.za

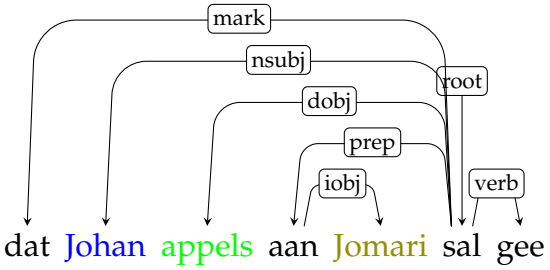


Converting the clause to future tense results in a slightly different ordering of verbs (auxiliary before instead of after the main verb), but does not introduce additional structural changes to the constituency parse:

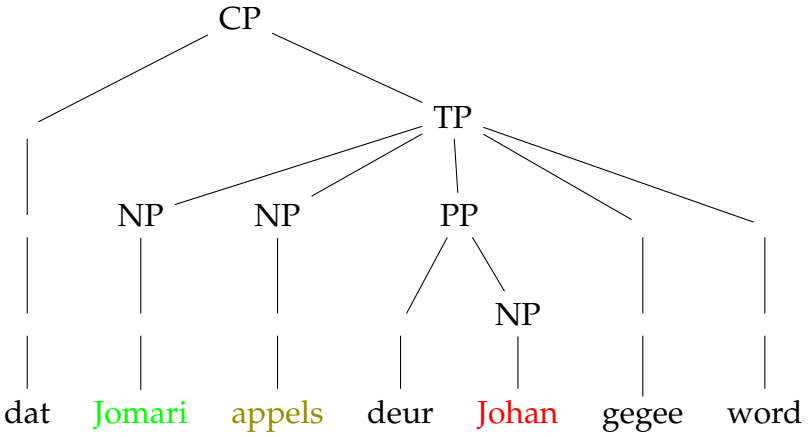


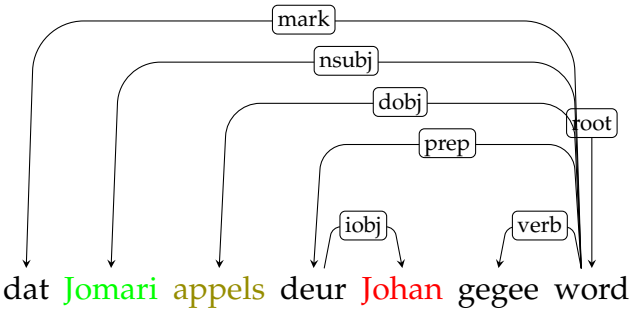
The dependency parse for this sentence is:

Stellenbosch University https://scholar.sun.ac.za



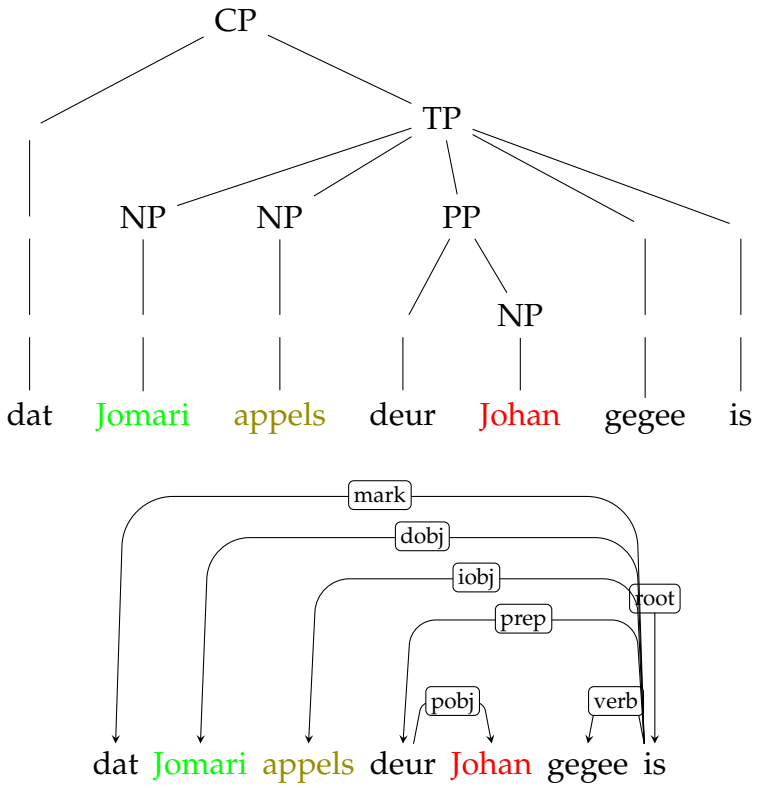
Subordinate clauses can also occur in the passive voice. In the present tense, this results in the following constituency and dependency graphs:



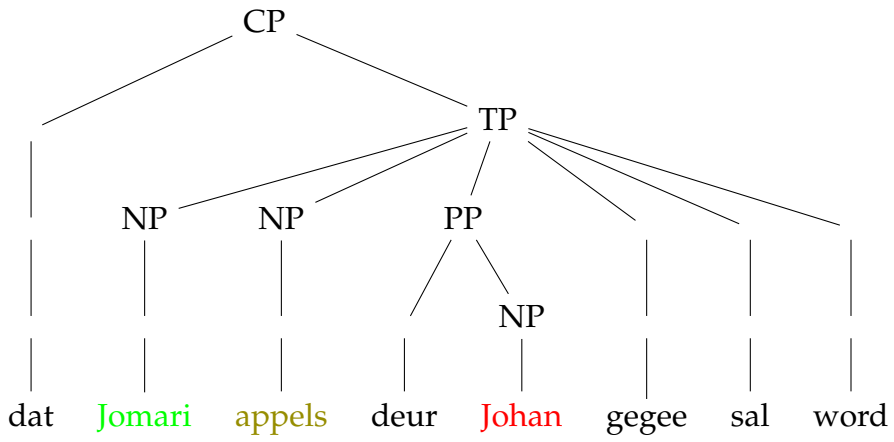


Past tense is structurally identical to present tense with lexical changes:

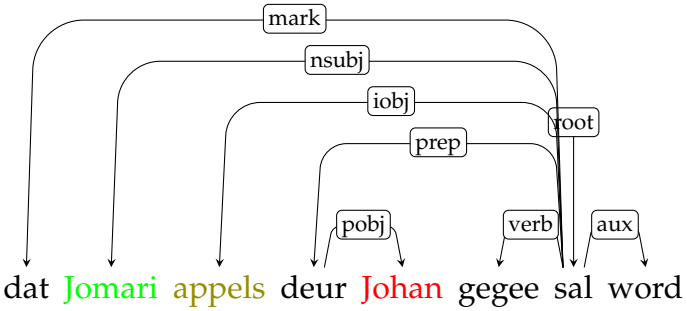
Stellenbosch University https://scholar.sun.ac.za



Future tense, again, gains additional complexity in the form of the additional auxiliary verb *word* ("be"):



Stellenbosch University https://scholar.sun.ac.za



We therefore define a new set C_s containing the following subordinate patterns:

$$C_s = \{cSOIV, cSOIvV, cSOIVv, cOISVv, cOISVvv\}$$

and extend the definition of C accordingly:

$$C = \{C_m C_c * C_s *\}$$

4.2.3 Infinitival clauses

Infinitival clauses are characterised by the absence of an overt grammatical subject and a verb in the infinitive form. In Dutch three types of infinitival clauses are permissible: bare infinitives, constructions with an infinitive marker (*te*) and constructions including a complementizer (*om*) and *te*, as illustrated by Deumert [33] (Afrikaans glosses added by the author)

(26) *NDL* slapen is gezond
AF slaap is gesond
'sleeping is healthy'

(27) *NDL* Laura was blij met haar moeder te reizen
AF *Laura was bly met haar moeder te reis
'Laura was pleased to travel with her mother'

(28) *NDL* het is een meisje om te zoenen
AF dit is 'n meisie om te soen
'the girl is worth kissing'

Afrikaans has retained all three forms, but their frequency and productivity vary considerably. Bare infinitives are used frequently, as in the case of:

Stellenbosch University https://scholar.sun.ac.za

(29) trou is nie perde koop nie
marrying is not horses buy PART-not
'getting married is no simple affair'

Full infinitive constructions with *te* have become less common and are productive only in combination with the copula [34, 278], the modals *behoort*, *hoef* and *durf*, and in more formal contexts also with the verbs *skyn*, *meen* and *wens* [33, p. 204]:

(30) dit is nie te versmaai nie
it is not to distain PART-not
'it is not to be sneezed at'

(31) Jan behoort te wen
Jan should to win
'Jan should win'

(32) die storie skyn waar te wees
the story appear true to be
'The story appears to be true'

Full infinitive constructions with *om te* have become completely generalised and is the standard form of the infinitival clause in Afrikaans [33, p. 204]:

(33) Ek is bly om jou te sien
I be happy for-INF you to see
'I am happy to see you'

As is the case with Dutch, the *om te* construction in Afrikaans has a word order that requires three mandatory elements: complementizer, infinitive particle and non-finite verb. The argument pattern for this type of clause (with mandatory elements italicised) can therefore be rendered as follows:

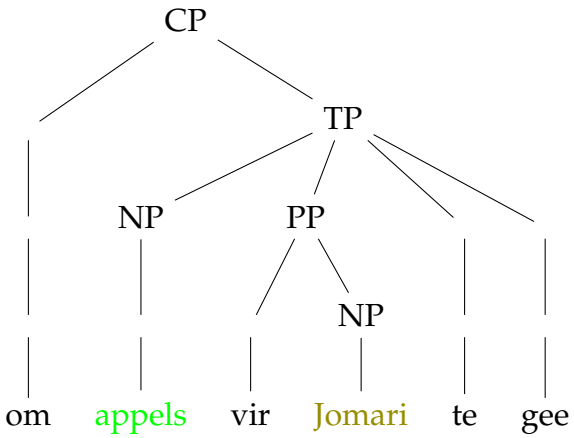
om + arguments + *te* + *non-finite verb*

For example:

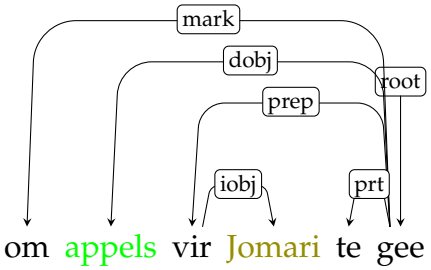
(34) ek beplan *om* die appels vir Jomari *te gee*
I plan *for-INF* the apples for Jomari *to give*
'I plan to give the apples to Jomari'

From a constituency perspective, this construction is parsed as follows:

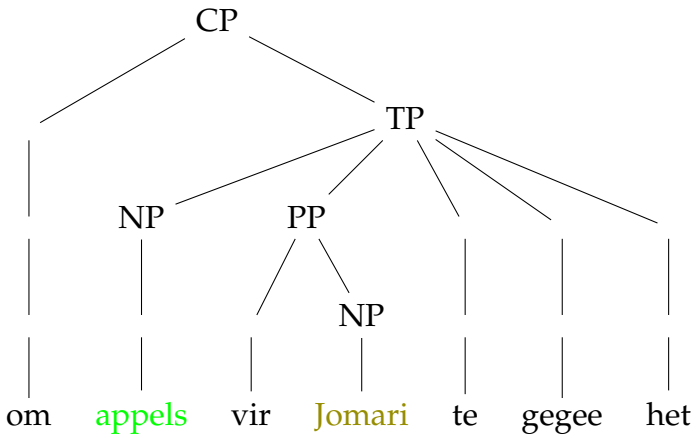
Stellenbosch University https://scholar.sun.ac.za



The dependency representation is structured as follows:

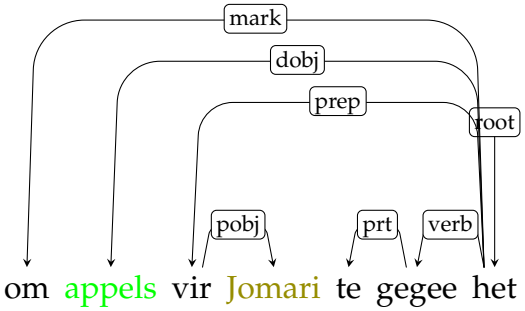


Converting this version of the full infinitive to a form in which an action in the past is indicated results in the following structures[11]:



[11] An infinitival clause cannot be in past or future tense, since it has no finite verb able to convey this information.

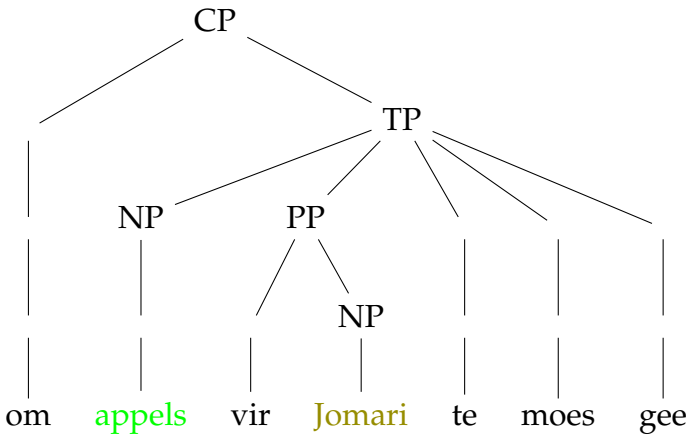
Stellenbosch University https://scholar.sun.ac.za

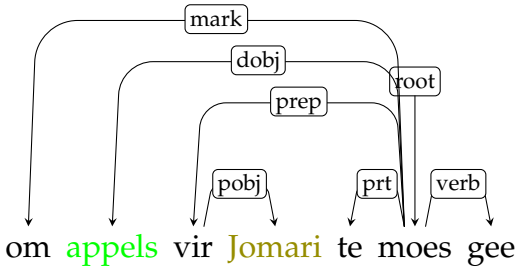


Converting the clause to a form in which future action is indicated results in a somewhat ungrammatically sounding clause:

?om appels vir Jomari te sal gee

We can however use another modal verb to indicate a third possible structure for the infinitive clause:





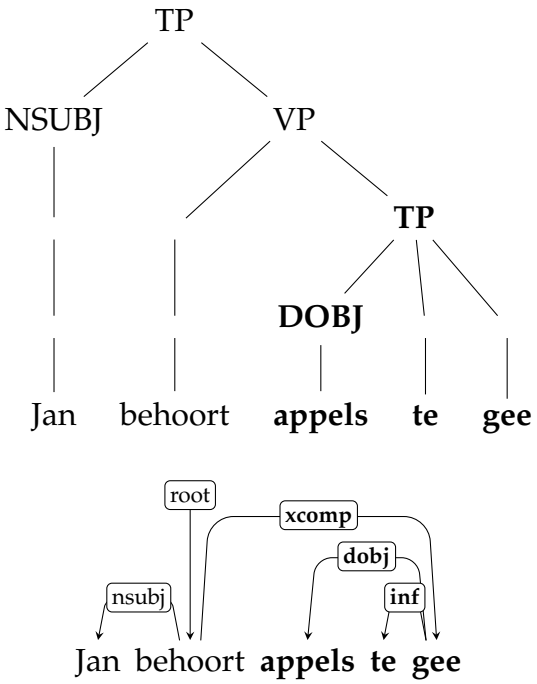
This provides us with the following argument patterns for infinitive clauses, utilising the symbols ι and τ for "infinitive complementizer" and "te" respectively (variations included):

- ιOIτV

Stellenbosch University https://scholar.sun.ac.za

- ιOIVτVv
- ιOIVτvV
- ιIOτV
- ιIOVτVv
- ιIOVτvV

The *te* infinitival clause has a structure similar to the *om te* infinitival clause. Taking a modified version of (28) as an example, we render its constituency and dependency representations as follows:



The bare infinitive can consist of a single word, namely the verb in its infinitival form (which is morphologically identical to a verb in the present tense) as in (35/36), or such a verb accompanied by one or more further constituents, as in (37):

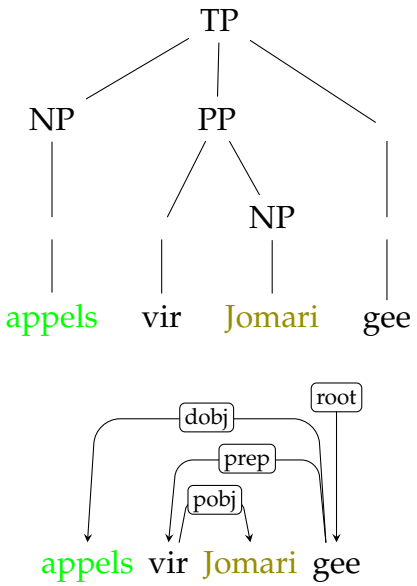
(35) Ek hou van gee
I like of give
'I like to give / giving'

Stellenbosch University https://scholar.sun.ac.za

(36) Gee is lekker
Give be nice
'To give / giving is satisfying'

(37) Ek hou van op die bed lê
I like of on the bed lie
'I like lying on the bed'

We parse bare infinitives as follows:



We therefore identify an additional pattern:

- τOIV

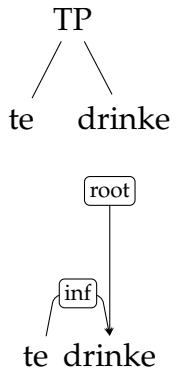
Finally, Afrikaans contains a small number of fixed, idiomatic expressions that take the form of infinitival constructions consisting of "te" plus a verb that only occurs in that construction or that shows obsolete inflection. Consider:

(38) hy gee haar iets **te ete**
he give her something **to eat**
'he gives her something to eat'

(39) iets te drinke is nie **te versmaai** nie
something to drink be not **to shun** not
'a drink is not to be sneezed at'

Stellenbosch University https://scholar.sun.ac.za

In these constructions the infinitive clause plays the role of a modifier to the indefinite pronominal direct object of a finite clause. For these examples, we are left with simpler structures:



- τV

Because of its fixed, almost idiomatic nature, this last structure will not be treated as part of our set of infinitive clauses that can be embedded in C, although we will return to it in later sections. This will also be our approach for the bare infinitive, which is used in slots usually reserved for verbal arguments.

We therefore define C_i as:

$$C_i = \{\iota OI\tau V,\ \iota OIV\tau Vv,\ \iota OIV\tau vV,\ \iota IO\tau V,\ \iota IOV\tau Vv,\ \iota IOV\tau vV\ \}$$

Consequently, we expand the definition of C as follows:

$$C = \{C_m C_c{*} C_s{*} C_i{*}\}$$

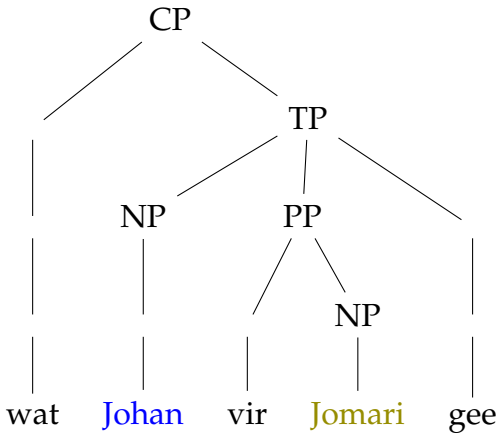
4.2.4 Relative clauses

Relative clauses behave differently from coordinate and subordinate clauses in that they are embedded within noun phrases rather than directly into other clauses. The start of a relative clause is signalled by a relative pronoun, which is part of a closed set of pronouns:

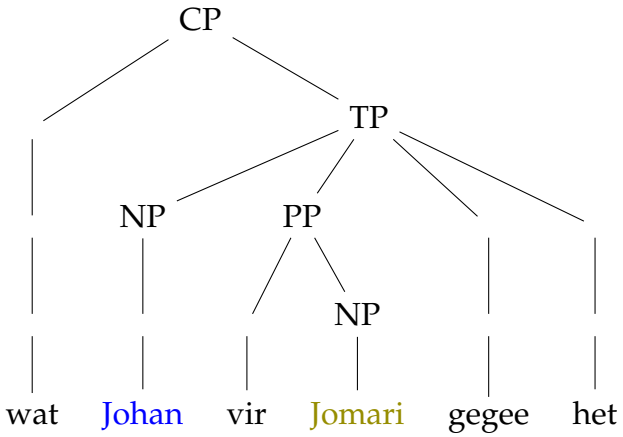
r = {*wat*, *wie*, *waar*, *wanneer*, *waarheen*, *waarvandaan*, *waarnatoe*, *waarop*, *waaronder*, *waarin*, *waaruit*, *waarlangs*, *waarmee*, *waartussen*}

Stellenbosch University https://scholar.sun.ac.za

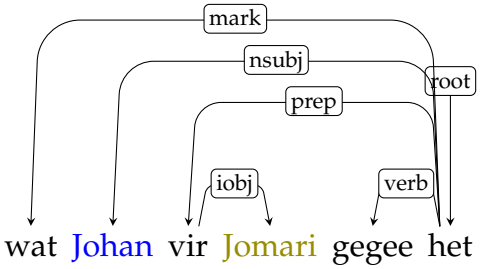
For the purposes of this study, we identify two types of relative clauses in Afrikaans, namely actively and passively voiced versions. The present tense for the actively voiced version is structured as follows:



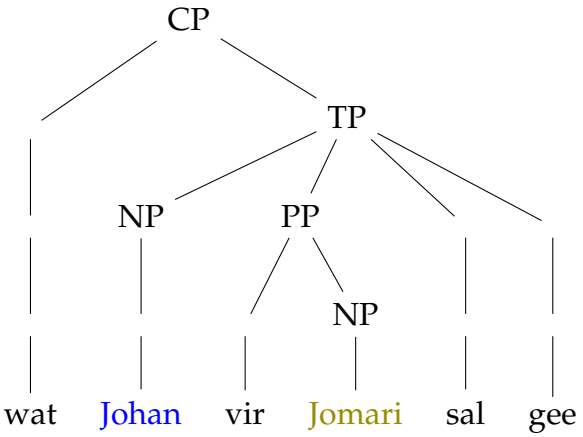
Conversion to past tense results in the following slightly more complex structures:

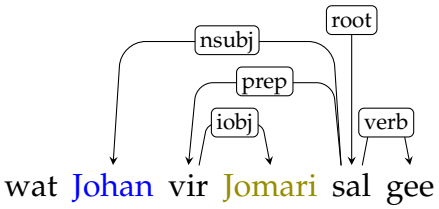


Stellenbosch University https://scholar.sun.ac.za



Conversion to future tense yields:



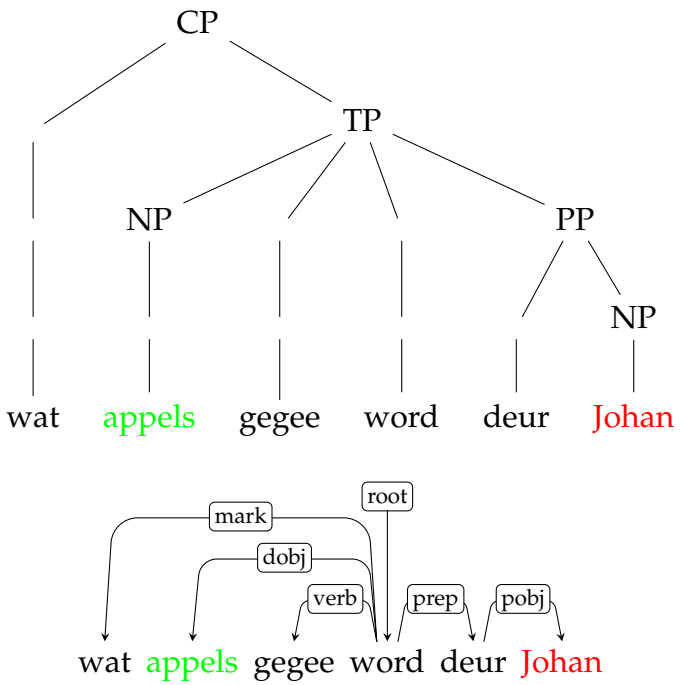


Using the symbol "r" to indicate the relative pronoun, we therefore identify the following patterns for relative clauses:

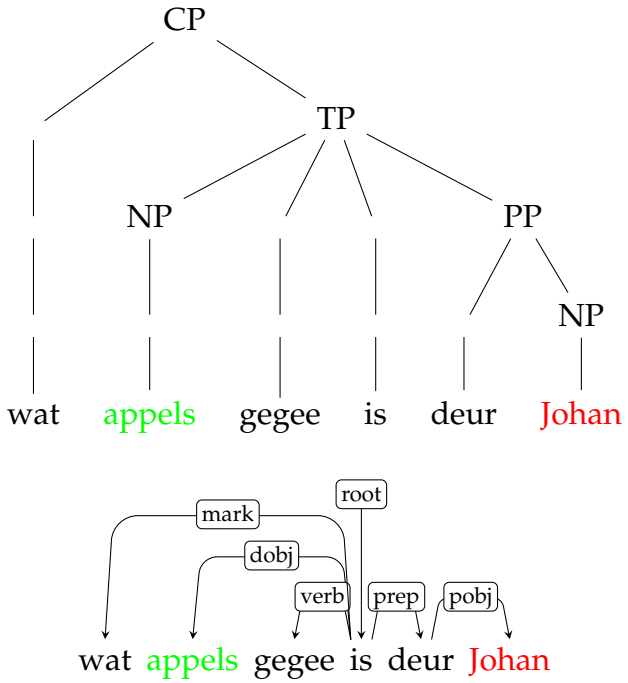
- rSIV
- rSIVv
- rSIvV

When each of the examples above are converted to the passive voice, they change as follows:

Stellenbosch University https://scholar.sun.ac.za

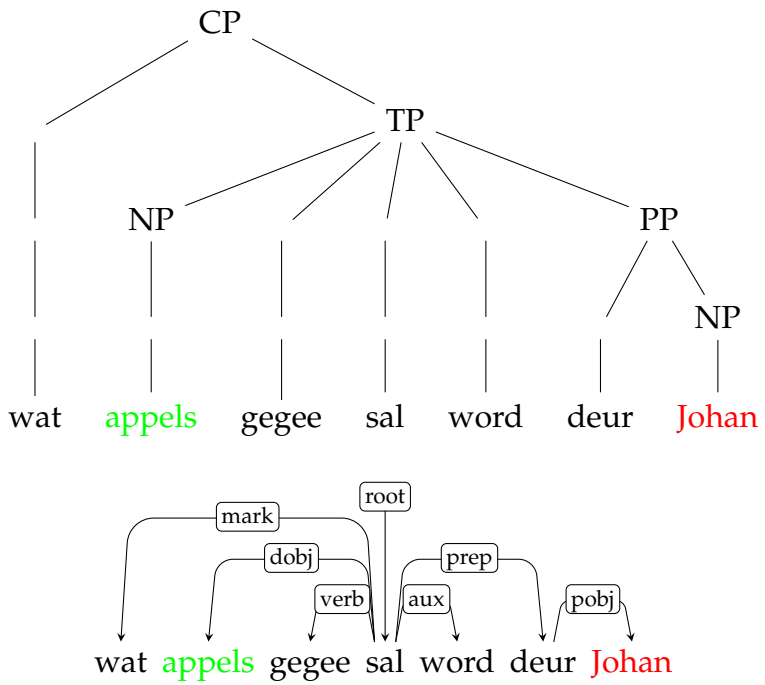


When converted to past tense the basic structure remains the same:



Conversion to future tense introduces the auxiliary verb *sal* between the past participle and the aspectual auxiliary, while the remainder of the structure remains constant:

Stellenbosch University https://scholar.sun.ac.za



We therefore identify the following additional patterns for relative clauses:

- rOVvS
- rOVvvS
- rSOI

We can therefore state that set C_r contains the allowable syntactic variations of Afrikaans relative clauses, and define it as:

$$C_r = \{\text{rSIV, rSIVv, rSIvV, rSIV, rSIVv, rSIvV}\}$$

Because relative clauses form part of an NP, we do not need to expand our definition of C.

4.2.5 An optimization

Prior sections have examined some of the main clause structures that occur in Afrikaans, and expanded the high level definition for an Afrikaans sentence to:

$$C = \{C_m C_c{}^* C_s{}^* C_i{}^*\}$$

Stellenbosch University https://scholar.sun.ac.za

A problematic assumption underlying this definition is that a main clause will always be followed by a linear sequence of other clauses, and that this sequence is fixed per clause type. In the current definition, for example, it is impossible for an infinite clause to occur before a coordinate clause. To allow greater flexibility, we can make use of OR operators (indicated by pipes) to include a new set of clauses C:

$$C_{\forall} = \{C_c * | C_s * | S_i *\}$$

and rewrite the core definition as follows:

$$C = \{C_m C_{\forall} *\}$$

Finally, since it is possible in written Afrikaans for subordinate clauses to stand on their own, we adjust our definition as follows:

$$C = \{C_m * C_{\forall} *\}$$

4.3 Phrase structures

4.3.1 Noun phrases

Afrikaans noun phrases are head initial, meaning that their nominal heads[12] are found to the right of their specifiers. Conversely, they sit on the left of their complements. A simple definition of a permissible noun phrase in Afrikaans is therefore:

NP = { SPECIFIER* HEAD COMPLEMENT*}

In this definition zero or more specifiers are allowed to the left of the head and zero or more complements are allowed to the right of the head, as can be seen in the following example (head rendered in bold):

[12] An ongoing debate in Linguistics concerns the question whether determiners or nouns should be considered the heads of nominal phrases. This study will assume that the noun is the head based on the fact that computational parsers seek to maximize the information extracted from their subject material, and one is hard-pressed to find a practical use for isolated determiners.

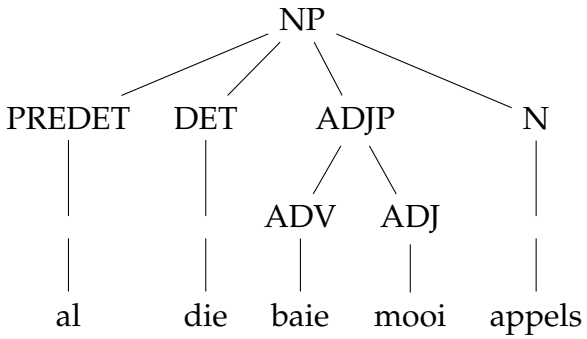
Stellenbosch University https://scholar.sun.ac.za

(40) die rooi **appels** aan die boom
the red **apples** against the tree
'the red apples on the tree'

The allowable order of specifiers are:

SPECIFIER = {PREDET? DET? ADJP*}

That is to say, one or zero predeterminers, one or zero determiners, and zero or more than zero adjectival phrases:[13]



Complements permitted to the right of the head include relative clauses, prepositional phrases and appositive adjective phrases[14], as illustrated by the examples in (41)–(43) respectively.

(41) die maer man **met die groen trui**
the thin man **with the green jersey**
'the thin man with the green jersey'

(42) 'n vrou **wat truie brei**
a woman **that jerseys knit**
'a woman who knits jerseys'

(43) die trui, **geruit en wollerig,** is gebrei deur die vrou
the jersey, **checkered** and **woolly**, PAST-be knitted by the woman

[13] We use the term *predeterminer* to mean "functors which select a nominal of type *determined* and which yield a nominal of type *quantified*" [101, 125] and *determiner* to mean words which "select an unmarked nominal and turn it into a marked NP of type determined" [101, p. 115]

[14] We use the term "apposition" to mean "specifying non-restrictive post-modifier modifying a noun phrase", borrowing from [47, p. 64].

Stellenbosch University https://scholar.sun.ac.za

'the jersey, checkered and green, was knitted by the woman.'[15]

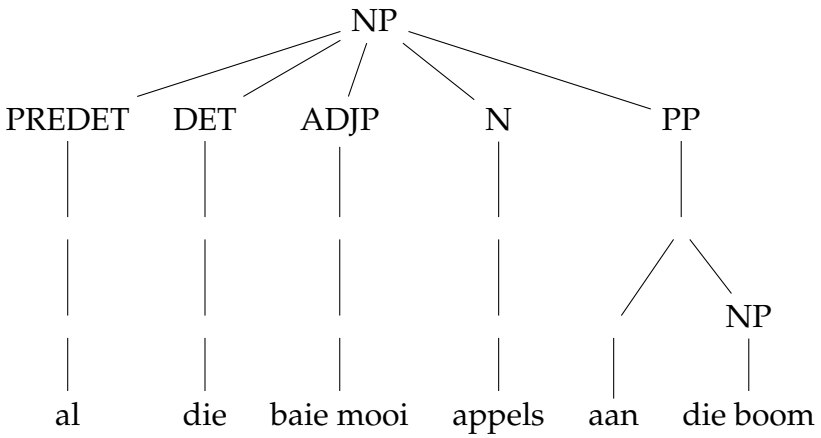
In the first example, the head of the noun phrase (*man*) is modified by a prepositional phrase. In the second the head (*vrou*) is modified by a relative clause. In the third, an adjectival phrase with adjectives in the predicative form acts as an appositive, modifying the head (*trui*). We can therefore state that the complements of an Afrikaans noun phrase can be defined as a set containing three members:

$$\text{COMPLEMENT} = \{ \text{ADJP}_{appos}, \text{S}_r, \text{PP*} \}$$

The phrasal head, for the moment, will simply be defined as a single nominal element (noun or pronoun):

$$\text{HEAD} = \{\text{N}\}$$

This allows us to build the following structure, illustrating the position of specifiers and complements in a noun phrase:[16]

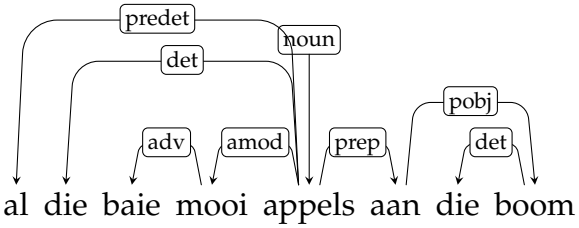


The dependency representation of the structure is:

[15]This example consists of an entire sentence, not simply a noun phrase, so as to properly indicate the adjectival apposition.

[16]In generative grammars, the structure of noun phrases are not as flat as presented in the graph above. We note this as a limitation, and do not consider further branching of the NP important. This is because all necessary information about the noun phrase that can be extracted by a parser is already contained within the flat structure.

Stellenbosch University https://scholar.sun.ac.za



Structural ambiguities do not represent themselves in either constituency or dependency representations of noun phrases. Our definition, however, is not complete, as it does not account for the recursive nature of noun phrases. Consider the following example:

(44) die bruin streepsakke rooi appels van sy plaas in die Boland
the brown grain-bags red apples from his farm in the Boland
'the brown grain-bags full of red apples from his Boland farm'

Here, a noun phrase ("die rooi appels") is embedded inside another noun phrase ("die rooi appels") as an adjunct. To account for this recursion, we restate our initial definition of N:

$$N = \{N^*\}$$

The head is now a single noun, or a noun containing zero or more nouns, containing zero or more nouns, containing zero or more nouns ad infinitum.

Noun phrases can also be coordinated with each other by coordinate conjunctions and (in the case of written language) punctuation. Consider:

(45) die streepsakke, kratte en palette of enigiets anders
the grain-sacks, crates and pallets or anything else
'the grain bags, crates and pallets or anything else'

The only constraints governing the joining of nouns in this construction is that the first noun (which, with our current definition, can also be a compound noun phrase) cannot be preceded by a conjunction, and that subsequent coordinated nouns have to be preceded by a coordinate conjunction. We can include these facts into our current definition by expanding it as follows:

$$N = \{N^* (cN)^*\}$$

Stellenbosch University https://scholar.sun.ac.za

The head is now a single noun, or a noun containing, as its specifiers, zero or more nouns or a noun coordinated with another noun – each of which can recursively contain more of the same.

This leaves us with a final type of construction to account for, as can be seen in the following correlative phrases:

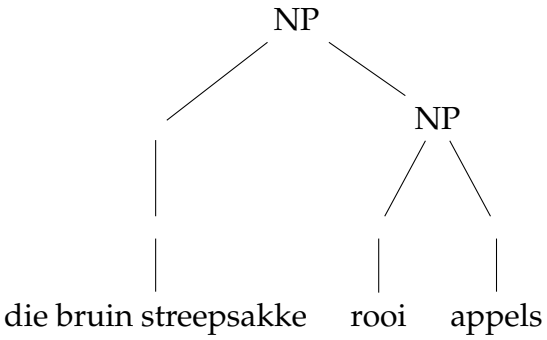
(46) sowel die streepsakke as die kratte
as-well the grain-sacks as the crates
'the grain bags as well as the crates'

(47) nóg die streepsakke nóg die kratte
neither the grain-sacks neither the crates
'neither the grain bags nor the crates'

These patterns require matching correlative conjunctions, which we will define as κ. This allows us to expand our definition of the head of a noun phrase to the following final version:

$$\text{N} = \{\text{N}^* \ (\text{cN}) \mid \kappa\text{N}+\kappa\text{N}+\}^{17}$$

In the event that embedded structures are required in the NP containing the head noun, they will be embedded as follows in the constituency treebank:



The question of how to handle coordination will be explored in section 4.5.1.

4.3.2 Adpositional phrases

Afrikaans allows prepositional as well as postpositional structures.

Prepositional phrases in Afrikaans follow the same linear pattern as in other Germanic languages:

[17] The plus symbol should be read as a Kleene plus, and indicates one or more instances of N.

Stellenbosch University https://scholar.sun.ac.za

$$PP = \{P\ NP\}$$

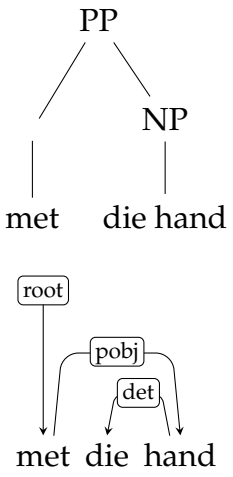
That is to say: a prepositional phrase in Afrikaans is formed by appending a single preposition in front of a noun phrase. The prepositions form part of a closed lexical set. Three examples of prepositional phrases are:

(48) met die hand
'with the hand'

(49) langs Jomari
'next to Jomari'

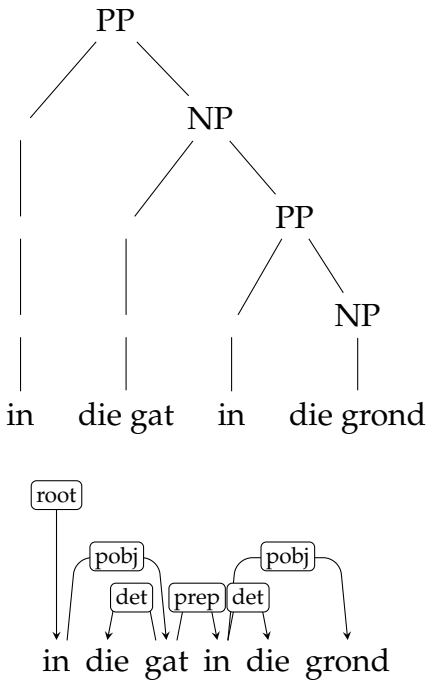
(50) binne die bruin streepsakke rooi appels
'inside the brown grain-sacks of red apples'

The structure of a prepositional phrase can be represented as follows from a constituency and a dependency perspective, respectively:



Afrikaans prepositional phrases can be embedded into other prepositional phrases on account of the fact that prepositional phrases are permissible complements of noun phrases, as in *in die gat in die grond* ("in the hole in the ground"). Therefore this extended structure is possible:

Stellenbosch University https://scholar.sun.ac.za

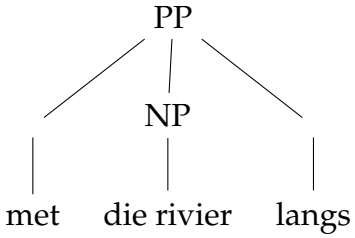


Phrases containing postpositions are often used to indicate movement or direction, as in the cases of:

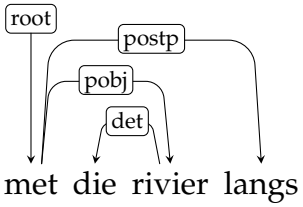
(51) hy spring van die dak af
he jump from the roof off
'he jumps from the roof'

(52) hy hardloop met die rivier langs
he run with the river along
'he runs along the river'

In these cases, we take the preposition to be the governor of the postposition, as shown in the graphs below. (In the constituency representation, this puts both adpositions on the same hierarchical level.)



Stellenbosch University https://scholar.sun.ac.za



4.3.3 Possessive constructions

Afrikaans largely lost the case system of its Germanic ancestors, and with it the ability to express the genitive case with articles.[18] As with modern Dutch, the possessive function can be expressed by the preposition *van*, as in the case of:

(53) (NDL) Het huis van de man
(AF) Die huis van die man
'The man's house'

Afrikaans also has a possessive particle *se*[19] which is highly productive and embeddable in the same way as adpositional phrases:

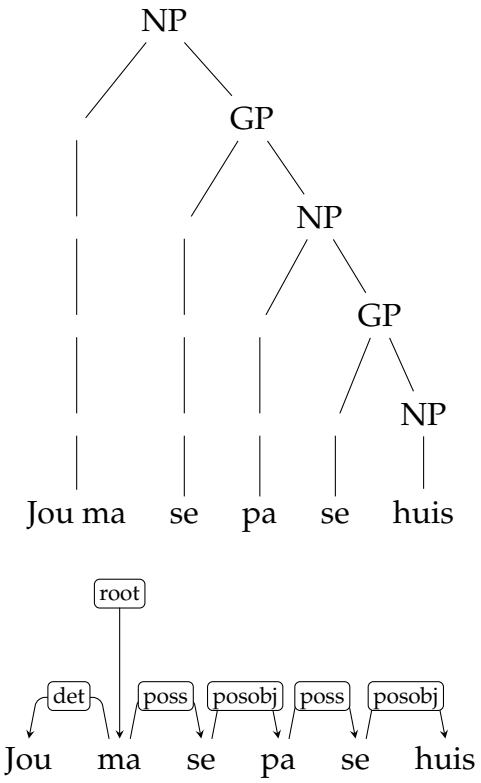
(54) Jou ma se pa se huis
Your mom POSS father POSS house
Your mom's father's house

We analyze this structure as follows:

[18] Archaic remnants of genitive articles in Afrikaans include the idiomatic expression "steen des aanstoots" and the adverb "desnoods".

[19] Historically an un-emphatic form of of *syn/zijn* ("his") [34, p. 98]

Stellenbosch University https://scholar.sun.ac.za



We use the label **GP** ("genitive phrase") for the constituency framework, and the labels *poss* ("possessive dependency") and *posobj* ("possessive object") for the dependency framework.

4.3.4 Verb phrases – Main clauses

Afrikaans second language students are often taught the $S_{v1}TOMP_{v2}I$ rule – a generalisation about the allowable structure of an Afrikaans sentence[20]. Being a generalisation it is, by its very nature, not entirely accurate (it does not, for example, account for inversions), but it does contain a measure of truth. Should one strip away the *S*, the remainder of the pattern is essentially a formula or guideline for forming the verb phrase in a declarative sentence.[21] Although it does not tell us anything in particular about each of the components expected in each position, we can gain a number of basic insights from this formula.

[20] STOMPI = Subject + Time + Object + Manner + Place + Infinitive
[21] This holds for main clauses: see section 4.3.5 for other patterns.

Stellenbosch University https://scholar.sun.ac.za

Firstly, no modifiers (e.g. adverbs) are allowed in front of the finite verb. Note that whereas the verb phrase (indicated in bold) in the sentence in (56) is grammatical:

(55) Johan **hardloop vinnig langs die rivier**
Johan **run quick along the river**
'Johan **runs quickly along the river**'

the one in (57) is not:

(56) *Johan **vinnig langs die rivier hardloop**

Secondly, modifiers cannot stand to the right of a non-finite verb, as illustrated by the difference in grammaticality between the following examples:

(57) Johan **het baie vinnig gehardloop**
Johan **have very quick PAST_run**
'Johan ran very quickly'

(58) *Johan **het gehardloop baie vinnig**
Johan **have PAST_run very quick**

The first observation holds without exception. The second, however, is where STOMPI breaks down: while it holds true for adverbial modifiers as in the examples above, it does not do so for prepositional modifiers. Consider the following semantically equivalent variations of the same sentence:

(59) hy **het gesels met die meisie**
he **has chat with the girl**
'he talked to the girl'

(60) hy **het met die meisie gesels**
he **has with the girl chat**
'he talked to the girl'

On the basis of these observations, we can now provide a definition of a syntactically well-formed Afrikaans verb phrase in a declarative main clause:

$$\text{VP} = \{\text{V}_f \text{ ADVP* PP* V}_{nf}\text{? PP*}\}$$

In this definition:

- V_f is the finite verb / verb catena starting with a finite verb

Stellenbosch University https://scholar.sun.ac.za

- ADVP* indicates zero or more adverbial phrases
- PP* indicates zero or more prepositional phrases
- V_{nf}? indicates zero or one occurrences of a non-finite verb

This does not yet account for the positioning of objects. As discussed in section 4.1.1, direct and indirect objects are allowed a relative degree of freedom, granted to it by the markers "vir" and "aan". If a non-finite verb is present, objects occur before it. Furthermore, adverbials are allowed between objects as in the case of:

(61) hy het Jomari vinnig appels gegee
he has Jomari quickly apples PAST-give
'he quickly gave Jomari apples'

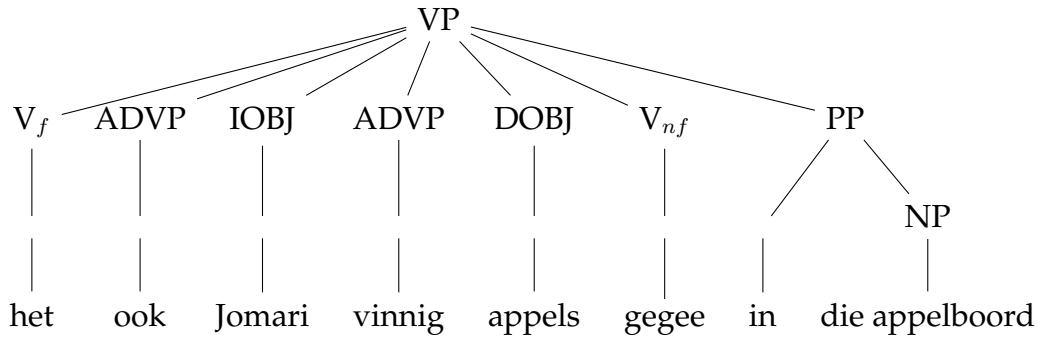
Borrowing from the German topological sentence model [35], we therefore introduce a new variable MF[22] to define the allowable arguments and adjuncts between the finite and the non-finite verb:

$$\text{MF} = \{ \text{(DOBJ? ADV* IOBJ?)} \mid \text{(IOBJ? ADV* DOBJ?)} \}$$

The definition of VP can accordingly be rewritten as follows:

$$\text{VP} = \{V_f \text{ ADV* MF } V_{nf}\text{? PP*}\}$$

The constituency representation of this structure, taken to its logical fullest with all allowable slots populated, is:[23]

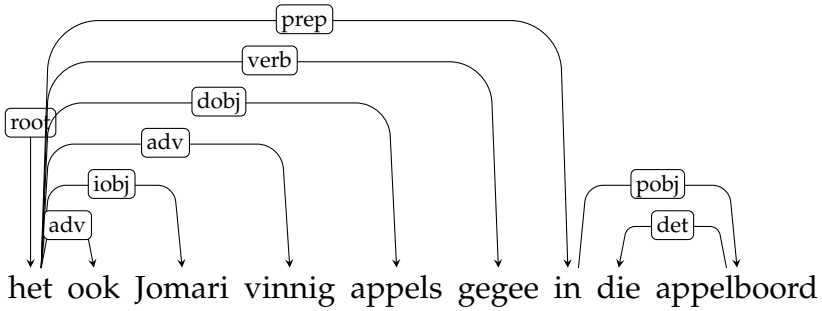


[22] Where MF stands for Mittelfeld – literally "middle field".

[23] In generative grammars, the structure of verb phrases are not as flat as presented in the graph above. We note this as a limitation, and do not consider further branching of the VP important. This is because all necessary information about the verb phrase that can be extracted by a parser is already contained within the flat structure.

Stellenbosch University https://scholar.sun.ac.za

The dependency version is:



From this we conclude that, for our purposes, the structure of the Afrikaans verb phrase in a main clause poses no structural ambiguities that need resolving in either the constituency or dependency frameworks.

4.3.5 Verbal adjuncts – Subordinate clauses

As illustrated in section 4.2.2, Afrikaans subordinate clauses that are introduced by an overt complementiser such as *dat* ('that') or *of* ('whether'/'if') are verb final, and exhibit SOV word order in the active voice and OSV[24] word order in the passive voice. To test how this affects the order of adjuncts, we can slot adjuncts into different sentence positions to gain an understanding of where in a subordinate clause they are permitted.

Performing this exercise with a subordinate clause in the active voice, shows that adverbial adjuncts are permitted after the grammatical subject, but not after any verbal complements introduced by prepositions:

(62) *dat **vinnig** hy appels aan Jomari gegee het

(63) dat hy **vinnig** appels aan Jomari gegee het

(64) dat hy appels **vinnig** aan Jomari gegee het

(65) *dat hy appels aan Jomari **vinnig** gegee het.

[24]It is important to note that the passive voice does not actually have a subject. We use the symbols "O" and "S" to ease reading, with "O" to refer to the grammatical subject of the passive construction, and "S" to the phrase functioning as the Agent. This applies for all future uses of this phrasing in this study.

Stellenbosch University https://scholar.sun.ac.za

(66) *dat hy appels aan Jomari gegee het **vinnig**.

Performing the same exercise with a prepositional adjunct shows that it can stand anywhere but in front of the subject:

(67) *dat **in die aand** hy appels aan Jomari gegee het.

(68) dat hy **in die aand** appels aan Jomari gegee het.

(69) dat hy appels **in die aand** aan Jomari gegee het.

(70) dat hy appels aan Jomari **in die aand** gegee het.

(71) dat hy appels aan Jomari gegee het **in die aand**.

Performing this exercise with a sentence in the passive voice, renders the same results for adverbial adjuncts:

(72) *dat **vinnig** appels aan Jomari deur Johan gegee is.

(73) dat appels **vinnig** aan Jomari deur Johan gegee is.

(74) *dat appels aan Jomari **vinnig** deur Johan gegee is.

(75) *dat appels aan Jomari deur Johan **vinnig** gegee is.

(76) *dat appels aan Jomari deur Johan gegee is **vinnig**.

And zero restrictions at all for prepositional adjuncts:

(77) dat **in die aand** appels aan Jomari deur Johan gegee is.

Stellenbosch University https://scholar.sun.ac.za

(78) dat appels **in die aand** aan Jomari deur Johan gegee is.

(79) dat appels aan Jomari **in die aand** deur Johan gegee is.

(80) dat appels aan Jomari deur Johan **in die aand** gegee is.

(81) dat appels aan Jomari deur Johan gegee is **in die aand**.

4.4 Long distance dependencies and discontinuities

The phenomenon of long distance dependencies is not a unique feature of Afrikaans. However, three salient features of the language – particle verbs, double negation and discontinuous relative clauses – allow for long distance dependencies that could present challenges to computational parsers. We therefore provide a brief overview of each.

4.4.1 Particle verbs

Particle verbs (also known as "phrasal verbs") are found in all Germanic languages, and consist of "combinations of verbs and preposition-like elements" which together "[form] a close semantic unit [68, p. 611]". They do not, however, behave the same in each language, as noted by Muller et al.:

> In German, Yiddish, Dutch and Afrikaans, particles precede the verbal stem in the infinitive. The particle is postverbal in V2 contexts ...and in imperatives. Note that the languages differ with respect to the position of the nominal object in relation to the particle: the object precedes the particle in German, Dutch and Afrikaans but follows it in Yiddish [68, p. 613].

To illustrate, consider the following sentence, in which the verb and its associated particle are indicated in bold, with the direct object italicised:

Stellenbosch University https://scholar.sun.ac.za

(82) hy **slaan** *die tent* **op**
he **put** *the tent* **up-PRT**
'he pitches the tent'

This example is in standard V2 format and therefore the particle is located after the verb and the direct object. Consider now the following variation, illustrating the behaviour in the infinitive, as adapted from Donaldson [34, p. 374]:

(83) ek het vergeet om die lig **af** te **skakel**
I have forget for the light **off-PRT** to-INF **switch**
"I forgot to switch off the light"

Muller et al. illustrate that particles in subordinate clauses are also pre-verbal: [68, p. 615]

(84) dat hy *die tent* vinnig aan die voet van die berg **opslaan**
that he *the tent* quickly against the foot of the mountain **up-put**
'that he quickly pitches the tent at the foot of the mountain'

They also note that Dutch (and by extension Afrikaans) allows an auxiliary between the particle and the verb in subordinate clauses:

(85) dat hy *die tent* vinnig **op** sal **slaan**
that he *the tent* quickly **up** will **put**
'that he will quickly pitch the tent'

and that the particle generally remains pre-verbal "in contexts where its verb is not in second position . . . because this position is filled by an auxiliary or another finite verb" [68, 614]. Muller et al. provide the following example from Donaldson [34, p. 374] to illustrate this:

(86) ek sal die lig AFskakel
I shall the light PART.turn
'I'll turn off the light'

Four observations regarding the behaviour of particles merit mention for our purposes. Firstly, pre-verbal particles and their verbs are orthographically conjoined in written Afrikaans, that is, part of the same string of characters and therefore not in need of any additional syntactic considerations. It is therefore only necessary to expound the behaviour of post-verbal particles.

Stellenbosch University https://scholar.sun.ac.za

Secondly, post-verbal particles are allowed a relatively great degree of separation from their verbal heads, as illustrated in:

(87) hy **slaan** *die tent* gou in die middel van die bos aan die voet
he put **the tent** quickly in the middle of the forest at the foot
van die berg langs die dorp **op**
of the mountain next the town **up**
'he pitches the tent quickly in the middle of the forest at the foot of the mountain next to the town.'

Thirdly, while post-verbal particles are strictly limited in their scope in terms of positioning relative to objects, they are allowed a high degree of freedom to move around between verbal complements in the same clause:

(88) hy **slaan** *die tent* **op** in die middel van die bos aan die voet van die berg langs die dorp

(89) hy **slaan** *die tent* vinnig **op** in die middel van die bos aan die voet van die berg langs die dorp

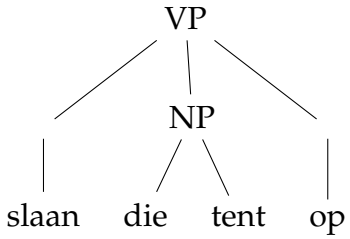
(90) hy **slaan** *die tent* vinnig in die middel van die bos **op** aan die voet van die berg langs die dorp

(91) hy **slaan** *die tent* vinnig in die middel van die bos aan die voet van die berg **op** langs die dorp

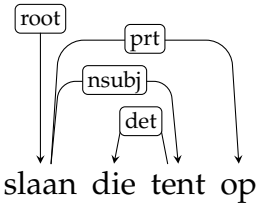
Fourthly, post-verbal particles are not permitted to stand before adverbial adjuncts:

(92) *hy **slaan** *die tent* **op** vinnig
*he **put** *the tent* **up** quickly

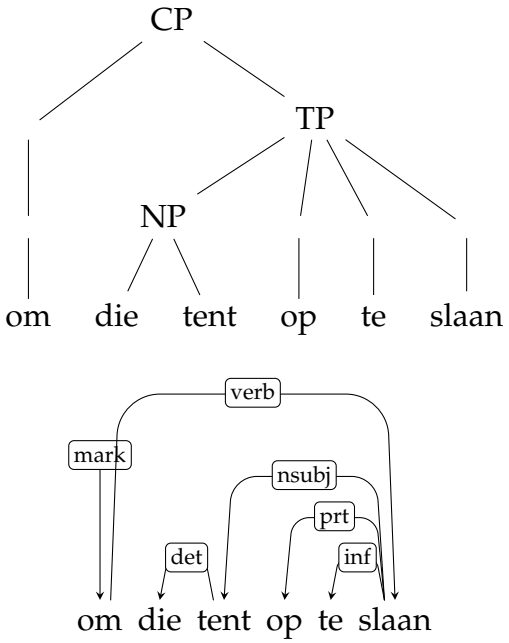
We are therefore left with two basic structures to account for post-verbal particles – a base form:



Stellenbosch University https://scholar.sun.ac.za



and an infinitive form:



4.4.2 Negation

Afrikaans' system of negation is unique within West-Germanic languages. Whereas negation in languages such as Dutch, German, Frisian and others is usually indicated with single nouns or adverbs[25], Afrikaans generally requires a negation particle (*nie*) in addition to the initial negator, where the initial negator could be *nie* ('not'), *geen* (*none*), *nooit* ('never') and various others. With a few exceptions, this particle is mandatory in written Afrikaans and serves as an indicator of the scope of that which is being negated. (Donaldson refers to it as the "scope marker" [34, p. 401]) – nomenclature we will borrow.) Consider the following example:

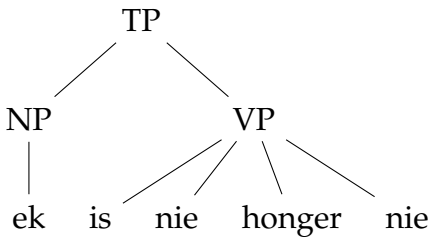
(93) ek is **nie** honger **nie**
I be **not** hungry **not-PRT**

[25] there are exceptions [32, p. 2]

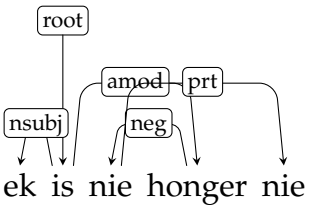
Stellenbosch University https://scholar.sun.ac.za

"I am not hungry"

The constituency representation of this structure simply puts the negation particle on the same hierarchical level as the negator:



The dependency representation looks as follows and (as mentioned in the previous chapter) readily illustrates the non-projective nature of the construction:



In this example the syntactic scope of the negation is the adjective "hungry", and the subject is assigned the property of "not hungry". Now consider a much longer version of this sentence (an English translation is included in a footnote to improve readability):

(94) ek is **nie** so honger as wat ek gisteraand tydens die partytjie ter ere van ons rugbyspan se oorwinning oor die Blou Bulle was **nie**[26]

Here the scope of negation is the entirety of the adjectival phase, spanning from *so* to *was*. This example illustrates the theoretically limitless range of Afrikaans negation particles.[27]

[26] 'I am not as hungry as I was yesterday evening during the party in honour of our rugby team's victory over the Blue Bulls'

[27] Human cognition, of course, places some very real limits on this range.

Stellenbosch University https://scholar.sun.ac.za

4.4.3 Discontinuous relative clauses

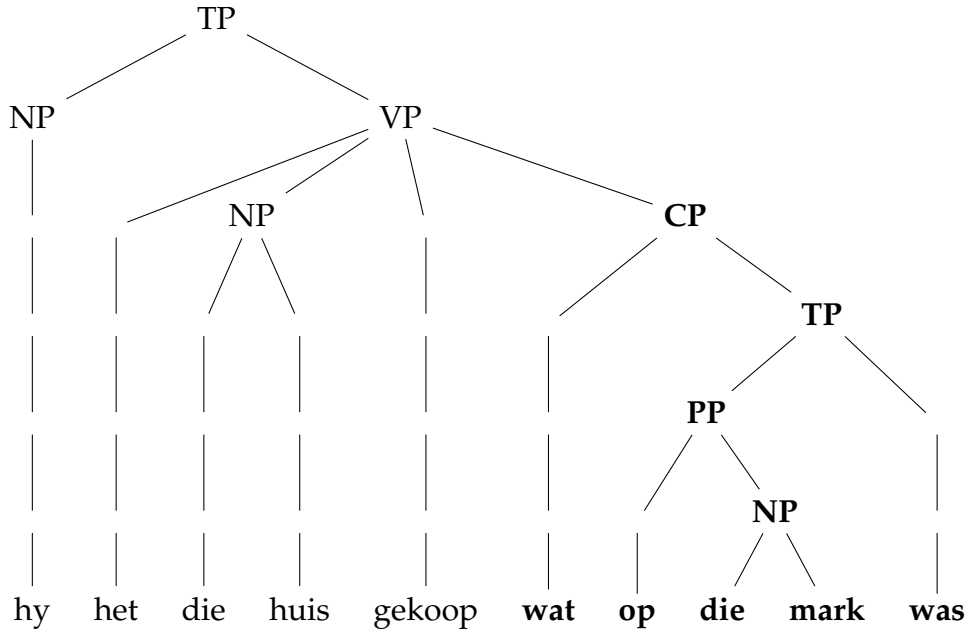
In cases where the lexical verb in a verb phrase is non-finite, a relative clause embedded in one of the verbal arguments can stand outside of said verb phrase, thereby causing a discontinuity, such as in the example of:

(95) hy het die huis gekoop wat op die mark was
hy has the house PAST-buy that on the market was
'he bought the house that was on the market'

The non-discontinuous version of this sentence is:

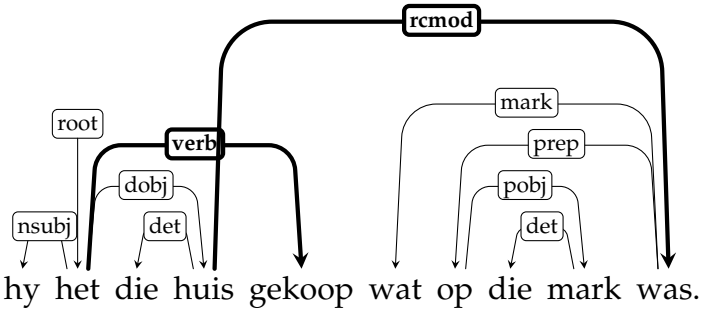
(96) hy het die huis wat op die mark was gekoop
Hy has the house which on the market was PAST-bought
'He bought the house that was on the market.'

In the constituency framework, we have no mechanism with which to annotate the discontinuity, and thus we locate it directly outside of the NP:



In the dependency framework, we annotate this discontinuity as follows:

Stellenbosch University https://scholar.sun.ac.za



In each of the three cases above, namely particle verbs, double negation and discontinuous relative clauses, dependency grammar seems to allow a more nuanced description of the sentence. On the face of it, the information contained in the relationships between governors and dependants of non-projective relations are lost when carried over to the constituency framework, which does not allow us to see exactly where a particle or a discontinuous phrase connect to its head. It could therefore be argued that, at least with regard to these three discontinuities, dependency grammar provides a more suitable framework for the computational parsing of written sentences in Afrikaans than constituency grammar.

4.5 Resolution of ambiguities

In the preceding sections several parsing ambiguities were highlighted. A resolution for each is critical for the integrity of the databank and eventual experimental results obtained from the parsers. A brief overview of each ambiguity will be presented below, as well as a discussion of how each will be handled in the treebank.

It is important to note that the decisions made in the sections below are not made because they are intrinsically "correct". Rather, they are made for the sake of consistency. If syntactically ambiguous sentences are annotated inconsistently, measurements of parser results will become inaccurate. This is not to say that all syntactic ambiguities are resolvable – natural languages are simply too fuzzy for this. However, where ambiguities can be resolved by means of justifiable a priori decisions, this will be done in order to ensure that the treebank is as consistent as possible.

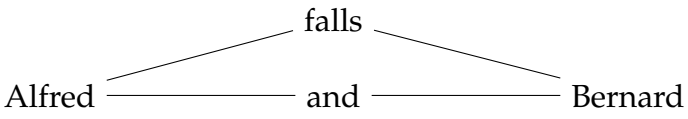
The decisions made in the subsequent sections are based on a combination of linguistic feasibility and practicality.

Stellenbosch University https://scholar.sun.ac.za

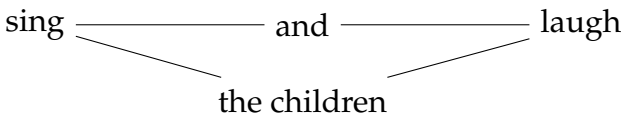
4.5.1 Coordination

While coordination presents hard problems to deal with in any formalism, it is "notoriously difficult to represent in dependency formalisms" [84][517]. The binary relationships that form the vertices in a dependency graph means that paratactic constructions are impossible to represent fully, since they require hierarchical equality from elements subordinated to each other.

This problem was recognised from the onset by Tesniére [98], who included both coordination and apposition into this problematic category, and introduced the concept of "junction" (original French: "jonction") as a solution. Using the examples "Alfred and Bernard falls" and "The children sing and laugh", he proposed that the issue be solved by means of additional horizontal lines, as illustrated in [98][328]:



and [98][343]:



The vertices linked by the horizontal edges (junctions) are considered hierarchically equivalent, and therefore a solution to the problem as Tesniére has defined it. (Essentially the same solution is also proposed for appositions.) Tesniére provided a variety of dependency parses for many more complicated structures to expound on his solution. There is, however, little sense in describing them in further detail, as all of them are essentially cyclical graphs and therefore in direct contradiction with the property of acyclicity required by both constituency and dependency grammars (see sections 3.3 and 3.4).

Mel'čuk, in the Meaning Text Theory framework [67], proposes a unidimensional approach that is summarized as follows by Mazziotta [65][pp. 29-30]:

1. In a phrase of the form *X and Y*, no element can remain "independent", i.e., unrelated to any other element.

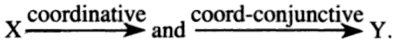
Stellenbosch University https://scholar.sun.ac.za

2. In the phrase *X and Y*, the conjunction cannot be the head, since the distribution of the phrase is determined by its conjuncts and by no means by the conjunction.

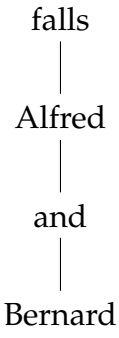
3. *X* is the head of the phrase, since the distribution of *X and Y* is that of *X*, and by no means that of *and Y*.

4. In the chunk *and Y*, the conjunction is the head: it determines the distribution of the expression to a greater degree than *Y*.

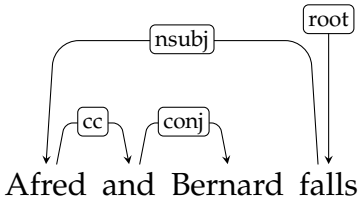
As a result, as Mel'čuk illustrates, we get [67, p. 41]:



The implication here is that a coordinated structure in a dependency grammar should be linear. Taking this view (and ignoring edge directions for the moment) Mazziotta adapts Tesnieré's original example as follows [65, p. 30]:



Mel'čuk's solution considers the initial coordinated component to be the root of the structure, which means that Tesniére's original example sentence can be parsed as follows within Mel'čuk's linear paradigm:



This approach suffers from a major deficiency (to which Mel'čuk admits) in that it is unable to account for constituent coordination properly as soon

Stellenbosch University https://scholar.sun.ac.za

as the semantics gets too complicated and additional syntactic elements are included. This is especially noticeable in the case of shared modifiers. Mazziotta explains [65, p. 30]:

> For instance, there is no difference in the description of *old men and women* meaning "old men + old women" and "old men + women (either old or not)" (Mel'čuk, 2009, 93). Another limit of the formalism appears in gapping coordinations or valency slot coordinations (non-constituent coordination). There is no way to correctly describe clustering as observed in: *John loves Mary*; and *Peter, Ann and John get a letter from Mary and roses from Ann*.

Solutions for these problems that have been suggested include the creatively named "bubbling" [51] and "paradigmatic piles" [52]. These solutions are somewhat deceptive, however, as they treat coordinated syntagms as phrases or groups, thereby blending dependency and constituency formalisms. This makes them impractical for this study.

While hybrid approaches and Tesniére's cyclical solution are clearly infeasible, Mel'čuk's solution seems plausible, and could be used as a basis for a pure dependency approach. In this regard, the work of Popel et al. [84] is especially relevant. Their study investigates the handling of coordination in treebanks for 26 different languages, and the various approaches to coordination are categorized into three distinct "families": [84][pp. 520-521][28]

- The Prague Family
- The Moscow Family
- The Stanford Family

In each of these families, the following key factors are identified:

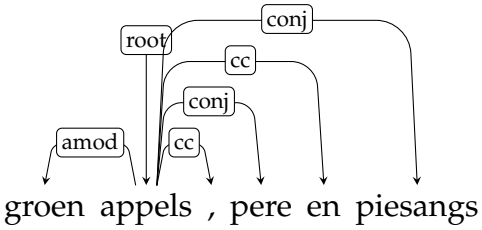
- Choice of head
- Attachment of shared modifiers
- Attachment of coordinate conjunctions

[28] See Appendix B

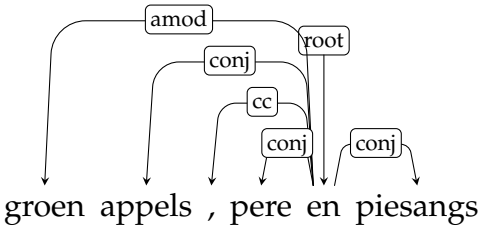
Stellenbosch University https://scholar.sun.ac.za

- Attachment of punctuation

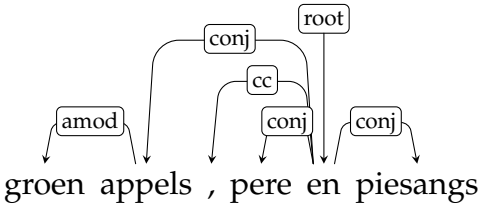
In terms of the **choice of head**, we choose the approach that allows the most expressiveness in terms of modifiers. Although one might be tempted to use a conjunct as the head, this automatically introduces a structural limitation in terms of the attachment of modifiers. Consider the following dependency parse for the compound noun phrase: *groen appels, pere en piesangs* ("green apples, pears and bananas"):



Because the first conjunct (*appels*) is both the root of the phrase and the governor for the adjectival modifier *groen*, it is impossible to know whether the modifier is meant to modify all the conjuncts, or only the one to which it is attached. This syntactic ambiguity can be resolved by instead using the leftmost conjunction as the root of the construction, which allows an additional layer of meaning. Thus, in the following parse all conjuncts are modified by the adjectival modifier



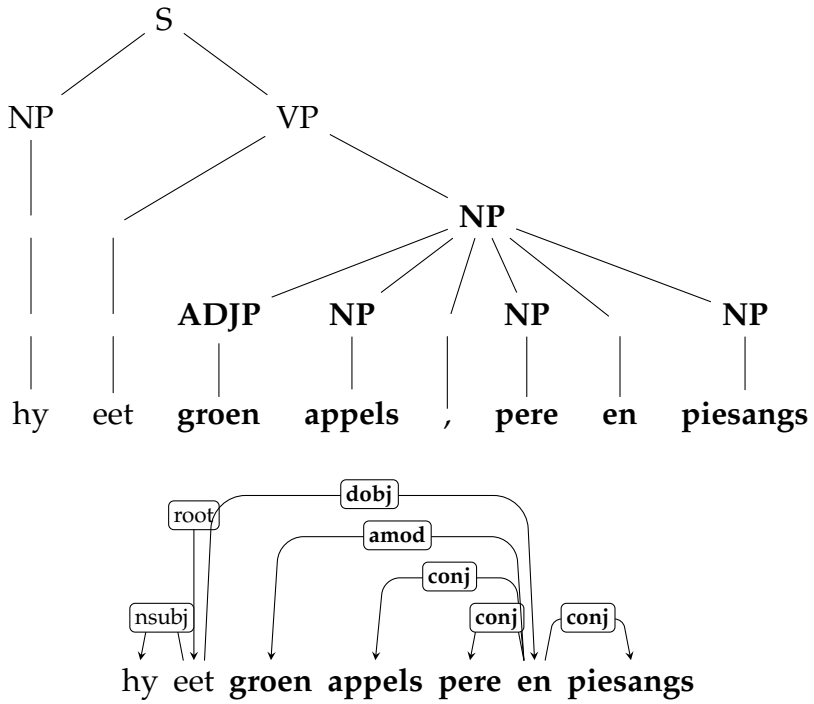
On the other hand, in the following alternate parse the modifier is governed only by the conjunct which it is supposed to modify:



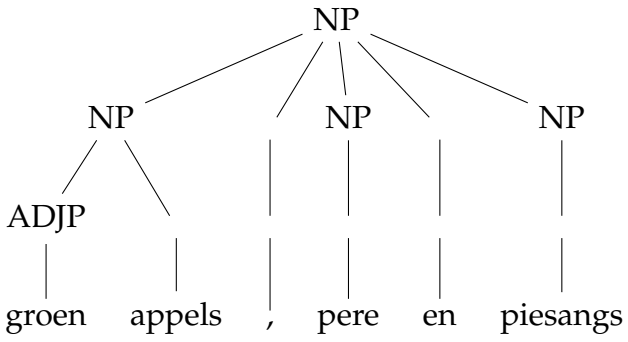
Stellenbosch University https://scholar.sun.ac.za

This approach allows greater expressive flexibility without including significant additional complexity. We therefore deviate from SUD in this regard, and align to the Prague family of treebanks.

To ensure that both frameworks remain as consistent as possible, we will label the coordinated structures in each according to their constituents or dependants, as in the cases of:



The attachment of shared modifiers will follow the same guideline as in the dependency framework. Therefore the example above indicates that all three noun phrases are modified by “groen”, while in the example below it only modifies the first noun phrase:



Stellenbosch University https://scholar.sun.ac.za

In the dependency framework, the **attachment of the coordinate conjunction** and **placement of punctuation** will be dependent on how many of each are available in a coordinated structure. If more than one are available, they will be governed by the first conjunction (or equivalent punctuation mark) as "cc" dependencies. If only one is available, it will be considered the root of the coordinated structure. In the constituency framework, because the coordinated structure of the phrase is already classified by its name, all conjunctions and punctuation marks will be on the first hierarchical level below the phrasal root node. (Here, again, we align to the Prague family of treebanks.)

By taking this approach, both frameworks stick to a strategy of prioritising function words as heads over content words. More importantly, it allows multiple types of conjuncts in a coordinated structure, as opposed to only conjuncts of the same type, as in the case of:

(97) om die gras te sny en ander lekker takies...
for the grass to cut and other nice task-DIM-PL
cutting the grass and other enjoyable tasks

In the example above an infinitive clause is coordinated with a noun phrase. In the SUD paradigm, the governor would be the infinitive clause and the second conjunct would be tagged as "conj", thereby removing any possibility of indicating the second constituent as nominal. From the perspective of our chosen paradigm, this is not a problem, as the conjunction is the head, allowing each dependent its own unique edge label.

4.5.2 Verb catenas

Afrikaans, being a language with very few verbal inflections, relies heavily on auxiliary verbs to indicate tense and mood. Verb catenae are therefore a common occurrence in sentences. In the constituency framework employed above this phenomenon is not problematic, as verbs in a VP occur on the same hierarchical level. In the dependency framework, where non-terminal syntactic nodes do not exist, this introduces the possibility of syntactic ambiguity.

As indicated in section 4.1.4, it is not immediately clear how the different verbs in a catena should connect to each other. The answer largely depends

Stellenbosch University https://scholar.sun.ac.za

on the particular analytical/theoretical framework that is adopted.

In the computational sphere, the Stanford Universal Dependencies [75] (SUD) is widely regarded as the standard. SUD is a set of annotation types and guidelines that has been built with the goal of aiding computational information extraction, and has co-evolved with other well-known frameworks built with similar goals in mind, including Google's universal part-of-speech tagset [80] and Interset [106], an interlanguage morphosyntactic tagset. SUD is based on the assumption that content words are hierarchically superior to function words, leading it to treat main verbs as governors, and auxiliary verbs as dependents. By implication, this school of thought treats auxiliary finite verbs as dependents of non-finite verbs.

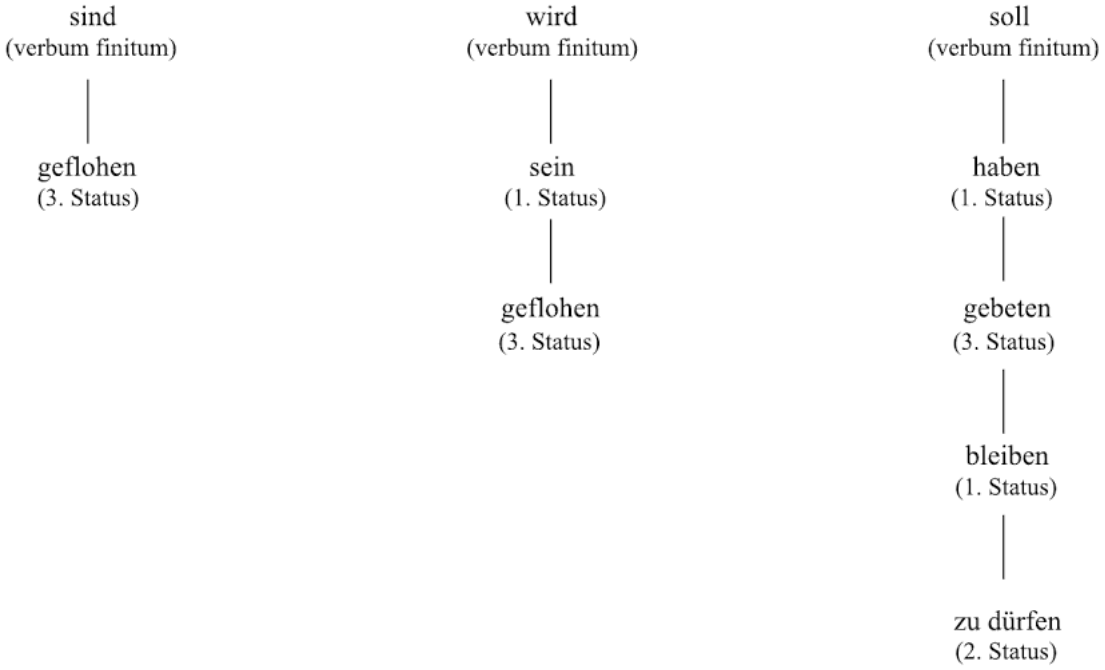
In linguistic circles, this approach is contested. Groß and Osborne, for example, note that most constituency grammars (including HPSG[29], LFG[30], CG[31], GB[32] and Minimalist Syntax) and dependency grammars (including WG[33], MTT[34], the German Schools – notably Helbig [46], Engel [39] and Rojek [89]) assume exactly the opposite: that function words are hierarchically superior to content words [42]. Examples of this can be seen in the work of Rojek [89, p. 113], who states:

> Im Zentrum des Interesses einer reinen Dependenzgrammatik stehen jedoch nicht die koordinativen Verbverbindungen und deren Statuskongruenz, sondern die subordinativen (hypotaktischen) Verbketten, die durch Statusrektion miteinander verbunden sind.
>
> ('At the center of the interest of a pure dependency grammar, however, are not the coordinative verb connections and their status congruence, but the subordinative (hypotactic) verb chains, which are connected by status regression.')

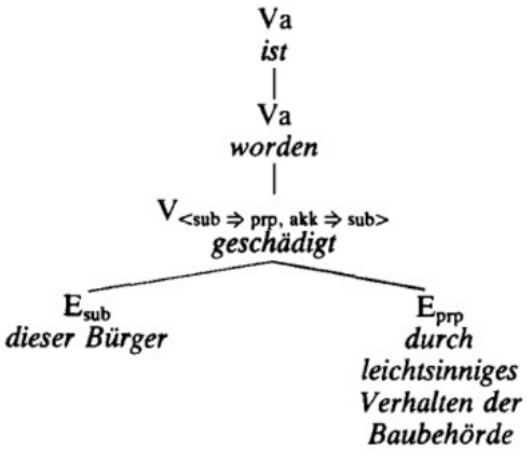
and includes examples rendering finite verbs hierarchically superior to non-finite verbs:

[29] Head Phrase Structure Grammar
[30] Lexical Functional Grammar
[31] Categorial Grammar
[32] Government and Binding
[33] Word Grammar
[34] Meaning Text Theory

Stellenbosch University https://scholar.sun.ac.za

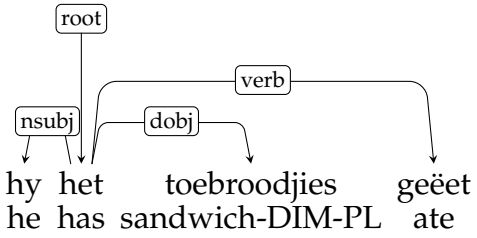


Another example of finite verbs being considered hierarchically superior in dependency grammars can be found in the work of Engel [39, p. 433]:



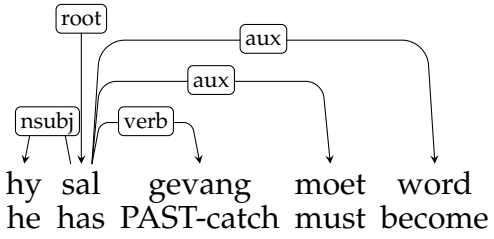
This study will side with the linguists, as there is no clear benefit by siding with the NLP community. We therefore consider finite verbs more significant than non-finite verbs in the context of dependency grammars, and therefore will always assign a finite verb as the root of a dependency graph, as illustrated in the following example:

Stellenbosch University https://scholar.sun.ac.za

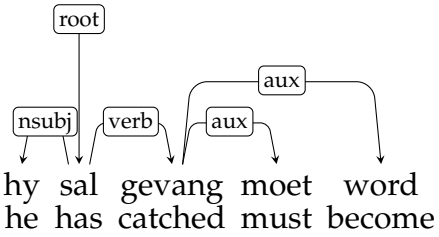


Where multiple non-finite verbs are present in a single catena, additional complexity arises, allowing for the following possible parses[35]:

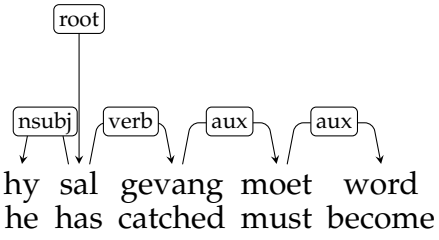
1. Finite verb governs all other verbs:



2. Finite verb governs main verb, main verb governs subsequent auxiliaries:



3. Finite verb governs first verb to its right, all subsequent verbs are governed by preceding verbs:



[35]There are likely additional possibilities, but enumerating them would not offer any significant benefits.

Stellenbosch University https://scholar.sun.ac.za

The third option will be chosen for treebank construction in this study. The reason for this is twofold. Firstly, this option is the simplest and most efficient. Annotators performing treebank annotation will be able to do their work quicker, as they simply have to link all verbs in a catena linearly. Secondly, this structure most closely resembles that of the constituency grammar above, where the catena, at least for the purposes of this study, is simply a flat structure.

Another unique property of Afrikaans verbs is what Ponelis [83] and De Vos [29] refer to as "complex initials". A complex initial is "a construction in which more than one verb appears in the verb-second position" [29][123], as illustrated by the following examples adapted from Ponelis [83, 326] and De Vos [29, p. 124]:

(98) sy kom lees vandag die boek
she come read today the book
'she will read the book today'

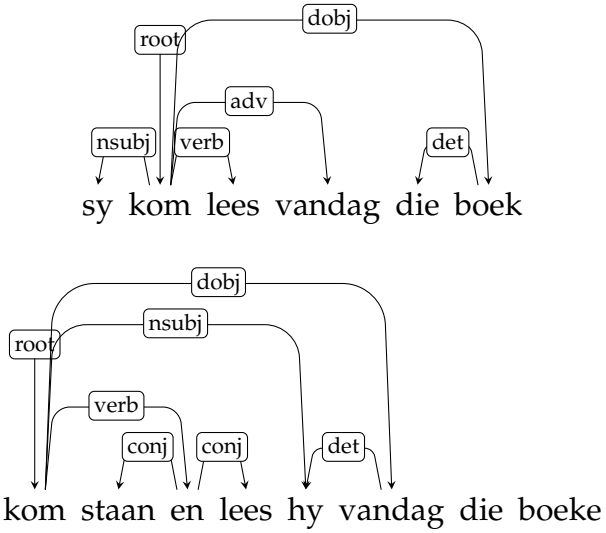
(99) kom staan en lees hy die boeke?
come stand and read he the books?
'does he come and read the books?'

The co-occurrence of verbs like *kom* ("come") with lexical verbs in complex initials contrasts with the fact that aspectual and modal auxiliaries cannot occur in such constructions, as illustrated by Ponelis [83, p. 326]:

(100) *sy het gelees vandag die boek
she AUX PAST-read today the book
'she read the book today'

The complex initials noted by Ponelis, De Vos and others are, essentially, verbal catenae consisting of a variety of verbs, often arranged paratactically. In these cases, the same principle as above will be followed: the first finite verb will be considered the root of the sentence, and all subsequent verbs will simply be linked to it linearly. Taking into account the structural decisions made regarding coordination in section 4.5.1, the examples above will be annotated as follows in the dependency framework:

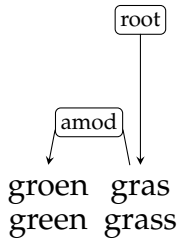
Stellenbosch University https://scholar.sun.ac.za



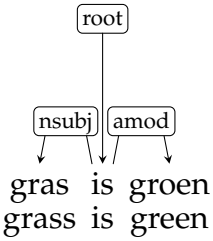
4.5.3 Adjectival governors

In section 4.1.9 uncertainty regarding the governors of predicate adjectives was raised.

Attributive adjectives, by the very nature of their function and positioning in a sentence, modify nouns. Within a dependency framework, this means that they should be governed by the nouns that they are modifying, as in the example below:

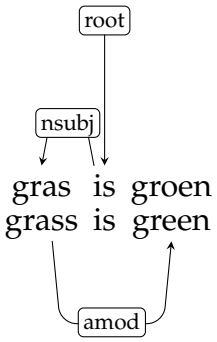


Transforming the same construction into a predicative one, however, requires incorporating a copular verb into the structure, and forces a decision on whether the adjective should be governed by the verb:



Stellenbosch University https://scholar.sun.ac.za

or whether the noun should continue to act as the governor, as would be the case when the adjective is in the attributive position:



Two main proposals have been put forward in the literature. SUD [75], being focused on information extraction, allows both approaches but prefers the latter. The Stanford Dependency Parser is therefore configured to apply the latter approach to sentence by default [28]. Since the type of influence exerted by SUD on NLP research communities is comparable to Chomsky's continuing influence on linguistic research communities, this approach is followed widely in the NLP world.

From a theoretical standpoint, dependency grammarians disagree with this stance in the same way that they disagree with SUD's decision to assign root status to non-finite verbs. As indicated by Osborne, this view is held by a large number of prominent DG scholars, including Mel'čuk and grammarians from the German schools [78]. In this view, the copular verb remains the root of the sentence, and governs the adjectival modifier.

As was the case with the decision on finite versus non-finite verbs, we will side with the linguistically more tenable standpoint. Predicative adjectives in copular sentences will therefore be taken to be governed by their associated copular verbs.

4.5.3.1 Complex negation

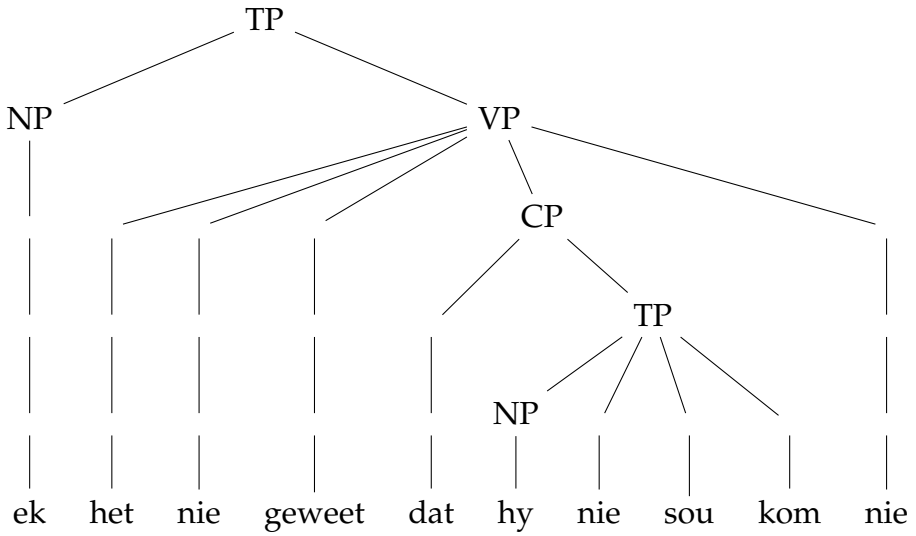
As indicated in previous sections, Afrikaans' double negation results in non-projectivity and potentially long distances between governors and dependants. These remarks were made based on simple examples, where a single negator governed a single negation particle. Language of course is seldom simple. As is the case with many other languages, Afrikaans allows

Stellenbosch University https://scholar.sun.ac.za

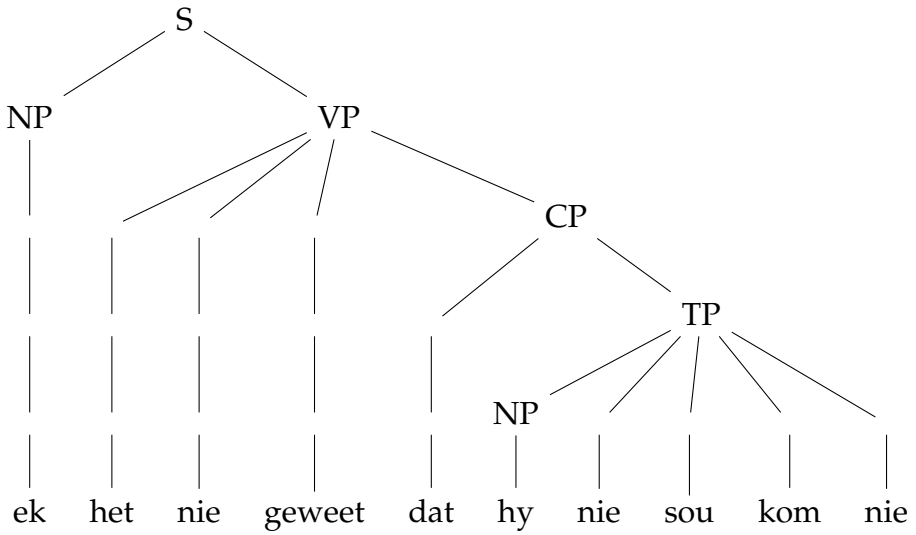
nesting of negated structures. In this case, however, multiple negators do not result in multiple scope accompanying markers, as shown by the following example adapted from Donaldson [34, p. 404]:

(101) ek het **nie** geweet dat hy **nie** sou kom **nie**
I have **not** known that he **not** would come **not**-PRT
'I did not know that he would not come.'

This presents different problems in different frameworks. In the constituency framework it is not entirely clear in which phrase the marker should be located, as it could equally be structured as follows:

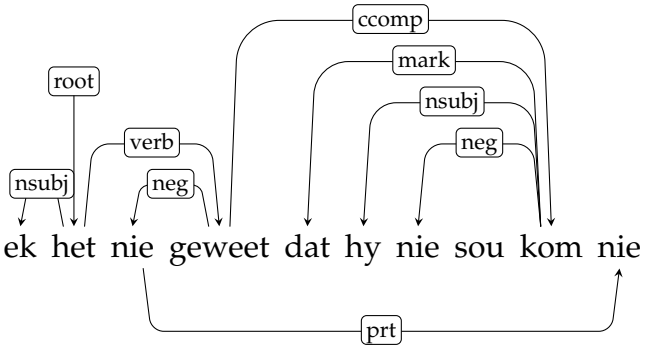


or as follows:

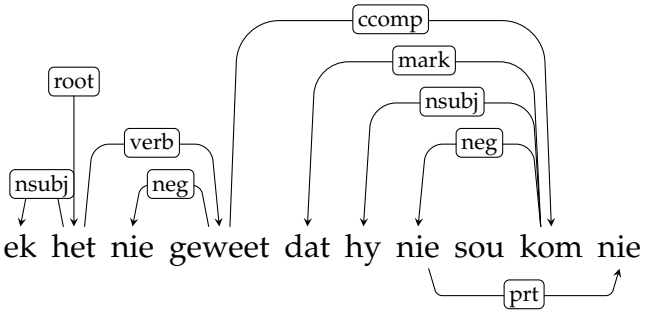


Stellenbosch University https://scholar.sun.ac.za

In the dependency framework the marker has to connect with either the first or the second negator, as illustrated in



and:



Since there does not appear to be a plausible solution to this problem in either of the two frameworks under discussion, our task is simply to choose an analysis that can consistently be mirrored in both frameworks.

All things being equal, the first options are chosen for both frameworks. The negation scope marker in complex negated structures will therefore sit in the same phrase as the leftmost negator in the constituency grammar, and be governed by the leftmost negator in the dependency grammar.

4.6 Summary

In this chapter, we provided an overview of the basic syntactic structures that commonly occur in written Afrikaans. We presented and discussed constituency and dependency parses for each structure and described the linear order of primary syntactic components for each structure by means of formal symbols. Additionally we presented parses for syntactic structures that could prove problematic to parsers (long-distance dependencies

Stellenbosch University https://scholar.sun.ac.za

and discontinuities), investigated several syntactic ambiguities encountered during the overview, and motivated a solution for each. In doing this, we established a set of linguistically motivated, formal guidelines for the annotation of an Afrikaans treebank.

Stellenbosch University https://scholar.sun.ac.za

Chapter 5

Experimental Methods

This chapter describes the empirical methods used to test **H0** and **H1**. It provides an overview of all work done to design and build the core dataset (Swartsbank), the procedures used to run experiments on it with four different syntactic parsers and the metrics used to assess the output of these experiments.

5.1 Research Design

5.1.1 Dataset

The primary dataset used in this study is Swartsbank – a purpose-built collection of 1 020 Afrikaans sentences, totalling 19,474 tokens. Building Swartsbank was necessary for two reasons: firstly, the data that it contains was required for this study, and did not exist elsewhere. Secondly, although a dependency treebank for Afrikaans was built and published[1], the quality of its annotations can be questioned[2], and it contains sentences from a single domain (government texts), which limits its ability to be representative of as many types of written Afrikaans sentences as possible.

The sentences in Swartsbank have been collected from the Leipzig Corpora Collection [41].[3] The Collection was selected because of its diversity (containing sentences from a variety of domains), accessibility (data is publicly

[1] See https://github.com/UniversalDependencies/UD_Afrikaans-AfriBooms

[2] See section A.1.

[3] We owe a word of gratitude to Professor Doctor Uwe Quasthoff (University of Leipzig) for his assistance in this regard.

Stellenbosch University https://scholar.sun.ac.za

available) and legal viability (data is licensed under Creative Commons). To compile Swartsbank, sentences were randomly selected from the following sections of the Leipzig Collection:

- News (Raw file: *afr_newscrawl_2011_100K-sentences.txt*)
- Web (Raw file: *afr-za_web_2015_300K-sentences.txt*)
- Wikipedia (Raw file: *af_wikipedia_2016-10k-sentences.txt*)

Sections were chosen to maximize the diversity of types of sentences included in Swartsbank. The selected set of sentences has the following characteristics (shown here both with and without punctuation marks):

	+punctuation	**-punctuation**
Average length	19.04	17.17
Median length	18	16
Standard deviation	8.59	7.9
Longest sentence[4]	59	54
Shortest sentence[5]	4	3

The lengths of these sentences are distributed unevenly around a median length of 18 tokens. This unevenness is caused by a minority of long sentences trailing off to the right of the distribution, as well as a minority of short sentences to its left, as seen in figure 5.1.

[4]The longest sentence in Swartsbank is Sentence 503: *"Ons maan alle Namibiërs om waaksaam te wees en ons versoek die Swapo-regering om hulle van hierdie soort vuil spel te weerhou, sodat die mense van hierdie land die party kan kies wat hulle glo hulle vir die volgende vyf jaar kan lei," het mnr. Spyskys gesê terwyl hy aan sy pyp geteug het.*

[5]The shortest sentence in Swartsbank is Sentence 135: *Dis 'n leeftyd.*

Stellenbosch University https://scholar.sun.ac.za

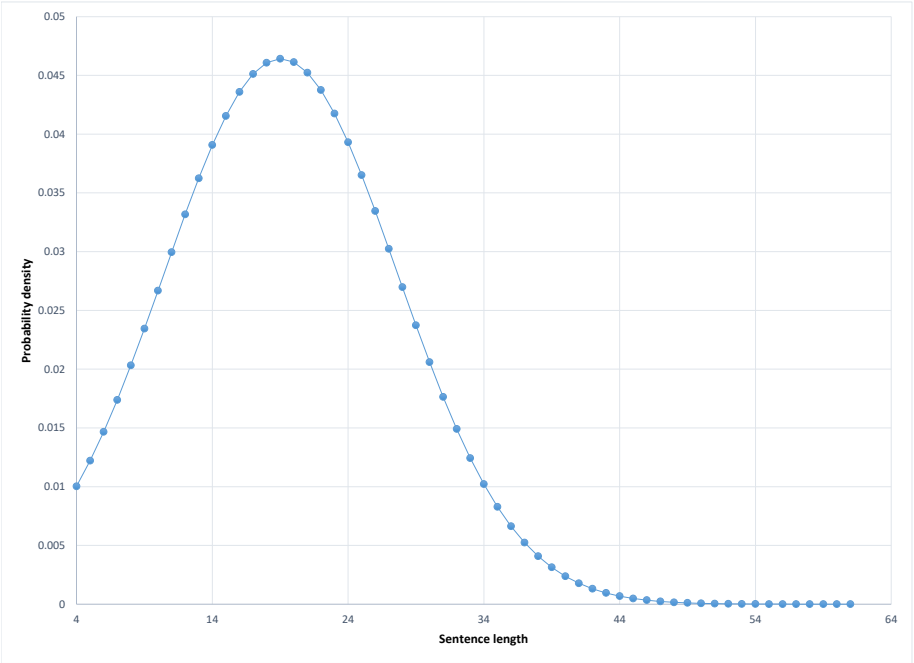
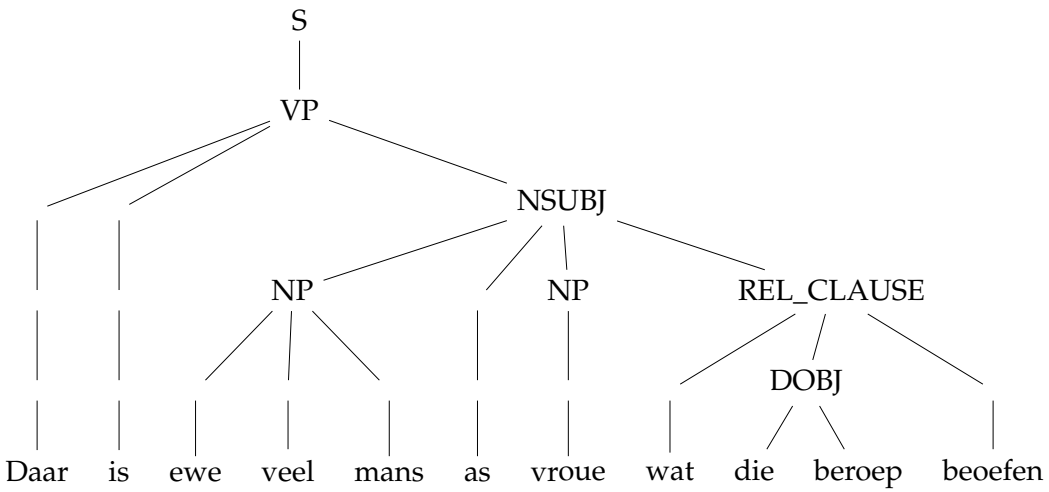


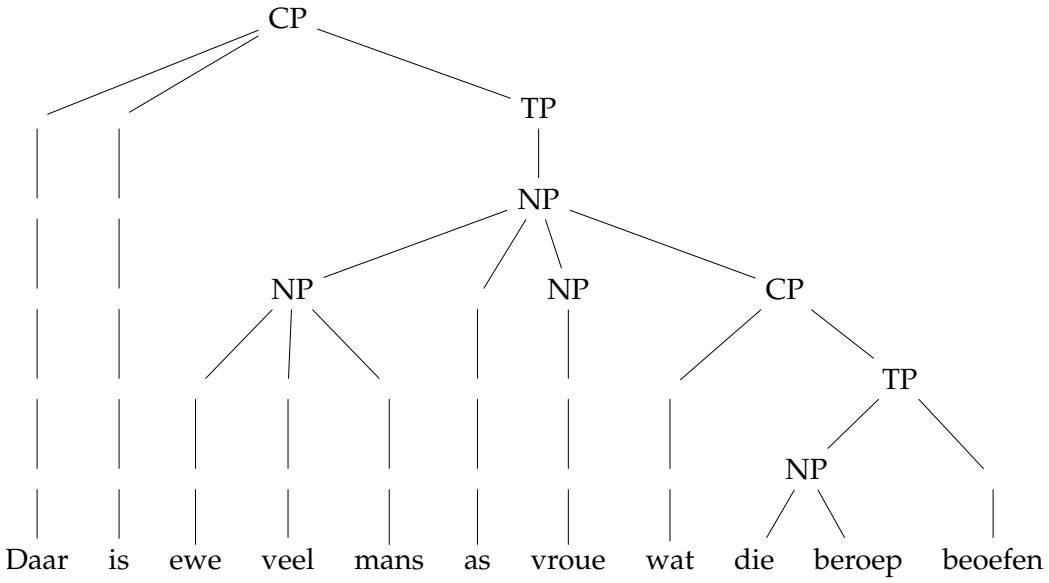
Figure 5.1: Sentence length distribution (with punctuation)

All sentences in Swartsbank have been annotated by hand. Sentences that were not well-formed were manually corrected during the annotation process. To allow the evaluation of multiple annotation schemas, the constituency section of Swartsbank has initially been annotated with annotations that differ slightly from what is presented in Chapter 4. This initial set of annotations – the "raw set" – makes more granular distinctions on a clausal and nominal level, and does not aim to be completely linguistically coherent. The information contained in the raw set, combined with the data in the dependency label set, allows us to heuristically change the structure of the treebank, depending on what type of annotation schema is required for a specific evaluation run. To illustrate: sentence 975 in Swartsbank has been annotated with the following "raw" annotations (POS tags excluded for simplicity's sake):

Stellenbosch University https://scholar.sun.ac.za



The graph above illustrates how the raw set allows a distinction between subjects, objects and different types of clauses, which is not found in the label set defined in Chapter 4. The VP is not used as a VP proper (it contains the subject, which is a syntactic violation), but rather as a marker of an inverted sentence, containing the preposed expletive *daar* and the copular verb *is*. Much of the information in the graph above is lost to us in the constituency tagset defined in Chapter 4. Having it safely stored in the raw version of Swartsbank, we can now easily convert the sentence to the syntactically "purer" version that adheres to the rules we defined:



Stellenbosch University https://scholar.sun.ac.za

Constructing Swartsbank in this way allows us maximum freedom to adjust its label set if necessary, based on the results of our experiments.

Each sentence in the raw version of Swartsbank contains:

- Part-of-speech tags for all tokens
- 1 $\times$ well formed constituency graph
- 1 $\times$ well formed dependency graph

Part of speech tags were manually added during the annotation process as parent nodes of tokens in the constituency section of Swartsbank. This information was then added programatically to the dependency section.

To ensure comparable evaluation results, constituency and dependency graphs in Swartsbank have been constructed to resemble each other as closely as possible within the bounds of each framework. This implies:

- Leaf nodes have identical part-of-speech tags
- Clausal and phrasal nodes (constituency) and catenas (dependency) span the same start and end nodes
- Nominal verbal arguments have the same labels across frameworks[6]

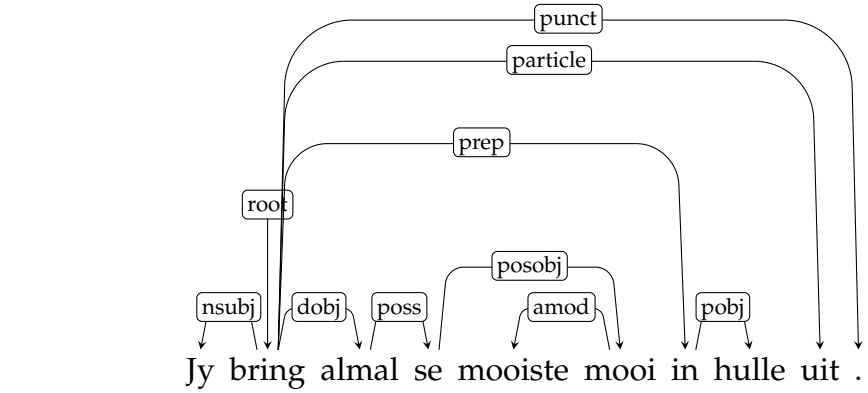
Notable exceptions to this rule of similarity are:

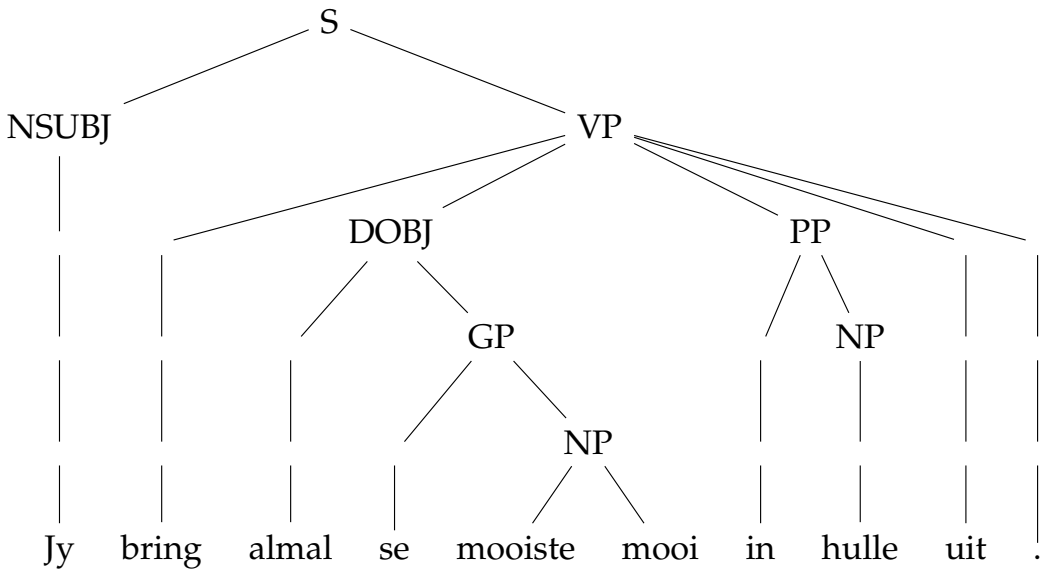
- Non-projective dependencies are not represented in constituency graphs[7]
- The hierarchical position of sentence / clausal ending punctuation marks differ between frameworks[8]
  - In dependency parses, sentence ending punctuation marks are linked as closely to the root node as possible
  - In constituency parses, sentence ending punctuation marks are linked as far away from the root node as possible.

To illustrate, consider the constituency and dependency parsers for sentence 187:

[6] *NSUBJ*, *DOBJ*, *IOBJ*, *NSUBPASS*
[7] This is a formal limitation of constituency grammar.
[8] This is an artifact of the annotation process.

Stellenbosch University https://scholar.sun.ac.za





In total, Swartsbank contains the following number of nodes and edges:

Table 5.1 Swartsbank edge and node counts

Dependency edges	18454
Constituency nodes (Excl. POS)	30745
Constituency nodes (Incl. POS)	50200

A complete list of all tags in Swartsbank is available in Appendix B.2.

Swartsbank is published as two separate files:

Table 5.2 Swartsbank filenames

Filename	**Format**	**Framework**
const.mrg	PENN Treebank combined [97]	Constituency
dep.conll	CoNLL-U format [16]	Dependency

Stellenbosch University https://scholar.sun.ac.za

5.2 Instrumentation

5.2.1 Software

Corpus building and parser evaluation software were written from scratch. This decision was made because no publicly available software packages contained the necessary features to build both a dependency and a constituency treebank from the same set of sentences. Furthermore no libraries to evaluate multiple parsers on the same set of sentences using the same set of metrics could be found.

Cobbling together a disparate set of software packages to reach the goals for this study made little sense. What was available publicly were unsupported, poorly documented, employed a variety of database systems and schemas, were written in multiple different programming languages and varied in their degrees of extensibility.

Three software packages were therefore designed and built for this study: a treebank storage tool, an annotation tool and a parser evaluation tool.

5.2.1.1 [1] Treebank – The Storage Package

To store and manipulate sentences and syntactic graphs, a package named "Treebank" was written. "Treebank" interfaces with a Cassandra database[9] which houses all data pertaining to syntactic annotations. It contains classes to represent sentences, tokens, constituency graphs and dependency graphs, methods to test the integrity of the treebank and a variety of utility classes.

5.2.1.2 [2] TreeGUI – The Annotation Package

To visually inspect, modify and annotate the sentences in Swartsbank, a graphical user interface called "TreeGUI" was written. "TreeGUI" allows the rapid annotation of sentences with dependency and constituency annotations. Its user interface is utilitarian, reflecting the need to build a treebank quickly. It makes use of keyboard shortcuts (to select labels) and colours (to indicate node and edge types) and provides features to correct spelling errors

[9] Cassandra, a NoSQL database, was chosen because the data schema was expected to expand during the course of development. This made a relational database such as MySQL impractical.

Stellenbosch University https://scholar.sun.ac.za

and incorrectly truncated sentences. It isn't the prettiest piece of software on the planet, but it gets the job done.

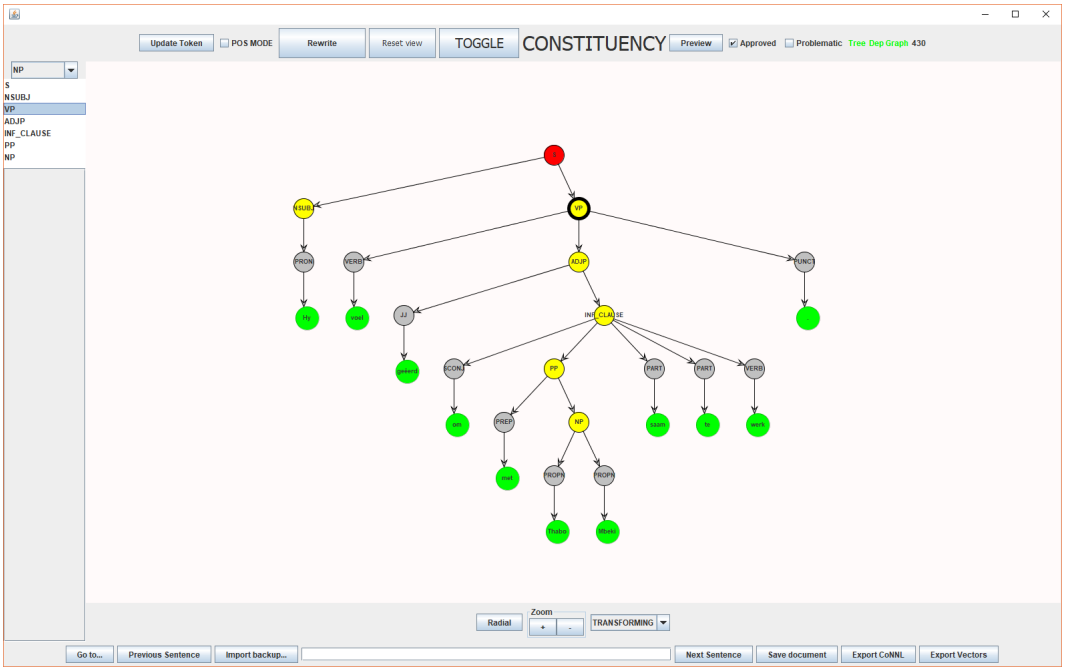


Figure 5.2: TreeGUI in action

5.2.1.3 [3] Parser – The Evaluation Package

To execute training and evaluation runs with each parser, a package named "Parser" was written. This package serves as a single point of integration for all parsers. It standardizes the interaction with each parser by providing a set of wrappers around it that is used during parser training and application phases. This allows parsers to be evaluated on identical sets of data. "Parser" contains methods to generate evaluation data based on specified parameters, train parsers on training data, evaluate parsers on evaluation data and calculate parse scores. It also allows the rule based transformation of the "raw" graphs in Swartsbank into different versions of itself.

5.2.2 Hardware

Development and parser evaluation were executed on an unremarkable desktop computer with an Intel Core i5-6500 3.20 GHz processor and 16 GB

Stellenbosch University https://scholar.sun.ac.za

of RAM.

5.3 Research Procedures and Pilot Testing

5.3.1 Testing procedure

To obtain empirical data to help determine an optimal grammatical framework for Afrikaans, four popular sentence parsers were evaluated on multiple versions of Swartsbank, each containing a different set of labels. Evaluation was done by performing k-fold cross validation (k=10) on the treebank using a procedure which can be expressed through the following pseudo code:

```
for 0 to 9 do
    for 1 to k do
        randomize order of sentences in treebank
        split randomized treebank into k parts
        for parser ∈ parsers do
            perform k-fold cross validation with parser
            log scores for parser
        end for
    end for
end for
```

This ensured that every parser has been evaluated on exactly the same sets of sentences.

Parsers were evaluated on four versions of Swartsbank: the Raw treebank and three transformed versions. This was done to account for the different types of annotations found in each half of Swartsbank – Constituency annotations describe syntactic categories; dependency annotations describe syntactic functions. These versions are:

Stellenbosch University https://scholar.sun.ac.za

Label set	Description
Raw	The initial annotation set used to construct Swartsbank
Generative	The linguistically motivated label set with which Swartsbank is distributed with
Functional	A transformed version of Swartsbank where constituency annotations have been changed to functionally match dependency annotations (advcl, acomp, xcomp et cetera)
Categorical	A transformed version of Swartsbank where dependency annotations have been changed to match syntactic categories (NP, VP, ADJP et cetera)

Results for each parser were logged on four levels, with each level capturing a different dimension of the results. Data points for each level were written in separate files in CSV format and imported and analyzed in Microsoft Excel.

Log level	Description
Totals	The average score of each evaluated section k, calculated as the average of all sentence scores in the evaluated section.
Labels	The average score of each node label (constituency) or edge label (dependency), calculated as the average score of each label in the evaluated section
Sentences	The score of every single sentence evaluated in every section during every evaluation run, along with the length of the sentence.
Mistakes	The details of each mistake made on each sentence during each evaluation run.

5.3.2 Parsers

The following sentence parsers were tested on account of their widespread availability, their reputation for high precision and the fact that they are all written in the same language (Java), which made integration into our own software easier:

- Berkeley Parser (Constituency) [79]
- Bikel Parser (Constituency) [10]

Stellenbosch University https://scholar.sun.ac.za

- Maltparser (Dependency) [73]
- Stanford Dependency Parser (Dependency) [18]

With the exception of the Berkeley Parser, all parsers are able to read POS tags together with tokens as input. Swartsbank's POS tags were therefore provided as part of the input during each evaluation run.

We summarize the characteristics of each parser, as well as the settings used to train them, below:

5.3.2.1 Berkeley Parser

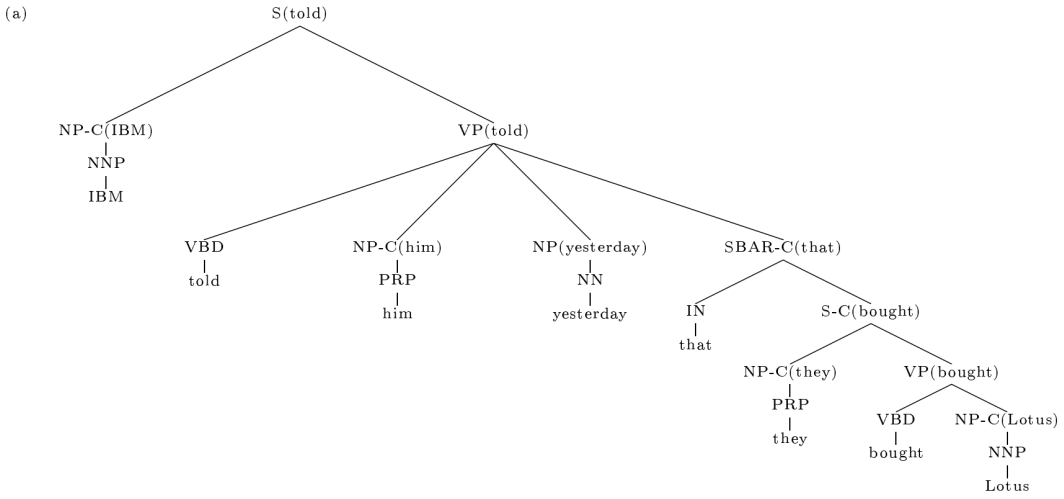
The Berkeley parser, named after the University of California, Berkeley where it was developed, is a probabilistic context free grammar (PCFG) parser that uses "an automatic approach to tree annotation in which basic nonterminal symbols are alternately split and merged to maximize the likelihood of a training treebank" [79, p. 433]. Here, splitting refers to the separation of existing labels into new ones based on observed linguistic trends. For example: "the symbol NP might be split into the subsymbol NP^S in subject position and the subsymbol NP^VP in object position." [79, p. 433] Merging refers to the parser's tendency to regroup split labels which would result in significant performance penalties with little benefit to the overall model.

The Berkeley parser is language agnostic, and does not require additional configuration to learn and parse Swartsbank. We therefore used its default settings in our experiments.

5.3.2.2 Bikel Parser

The Bikel parser is an implementation of Collins' head-driven statistical models [26] written by Daniel Bikel [10]. Collins' models "extend methods from probabilistic context-free grammars to lexicalized grammars, leading to approaches in which a parse tree is represented as the sequence of decisions corresponding to a head-centered, top-down derivation of the tree." [26, p. 589]. A grammar induced by the Bikel parser is parameterized with lexical heads, as shown in the following graph from Collins' PhD: [25, p. 18]:

Stellenbosch University https://scholar.sun.ac.za



The aim of a lexicalized grammar, according to Collins, is to

> choose an order of decomposition that allows a parameterization that reflects the local influence of lexical heads. This leads to an immediate constraint on the decomposition of the tree: a lexical head must be generated before all structure that is dependent upon it . . . This constraint leads us to a head-centered derivation of the tree [25, p. 18][p. 19].

Bikel parser is described as "multilingual" [10, p. 479]. No configuration changes were needed to induce a grammar for Afrikaans. We therefore used its default settings in our experiments.

5.3.2.3 Maltparser

Maltparser is a dependency parser developed by Nivre et al. at Linnaeus University that makes use of a "deterministic classifier approach" [76, 98]. It can be used "to induce a parser for a new language from a treebank sample in a simple yet flexible manner" [76, p. 95], and is claimed to obtain good results even with sparse datasets. Maltparser is different from both Berkeley and Bikel parsers in that it performs disambiguation of candidate parses deterministically, rather than probabilistically. It does so by

> using a greedy parsing algorithm that approximates a globally optimal solution by making a series of locally optimal choices, guided by a classifier trained on gold standard derivation sequences derived from a treebank. [76, p. 97]

Stellenbosch University https://scholar.sun.ac.za

The advantage of this approach is that it is fast, with parsing time being "linear or at worst quadratic in the size of the input" [76, p. 97]. Moreover, Nivre et al. explains, a deterministic parser has a steeper learning curve than a probabilistic parser and

> may in fact give higher accuracy with small training data sets. This is a natural consequence of the fact that the deterministic model has a much smaller parameter space, where only the mode of the distribution for each distinct history needs to be estimated, whereas a traditional generative model requires a complete probability distribution. [76][pp. 97–98]

Maltparser is language independent. To account for the non-projective constructions found in Afrikaans, we used its *stacklazy* algorithm, which allows the parser to learn and parse these types of constructions.

5.3.2.4 Stanford Parser

The Stanford Dependency parser is a transition based dependency parser developed by Stanford University's Natural Language Processing Group. Stanford parser "aims to predict a transition sequence from an initial configuration to some terminal configuration, which derives a target dependency parse tree" and uses a neural network classifier "to predict the correct transition based on features extracted from the configuration" [18, p. 741]. To overcome two problems associated with sparse transition based dependency parsers (poorly estimated feature weights and manually designed sets of feature templates), Stanford parser represents "words, POS tags and arc labels as dense vectors" and models their interactions "through a novel cube activation function". This allows the parser to "automatically learn the most useful feature conjunctions for making predictions". [18, p. 749] In experiments ran on large English and Chinese datasets, Stanford parser achieves a score that is 2% better than that of Maltparser [18, p. 748].

Stanford parser is language independent, and does not require special settings to learn and parse Afrikaans. We used 2000 training iterations to train its neural network, rather than the parser's default 20 000 iterations,

Stellenbosch University https://scholar.sun.ac.za

as preliminary experiments indicated that a significant increase in iterations was too expensive in terms of time.[10]

5.3.3 Runs & Durations

For each variation of Swartsbank, k-fold cross validation was performed ten times per parser, with k set to 10. Thus a total of 400 evaluations runs (100 per parser) were performed on each version of the treebank, adding up to a total of 1600 evaluation runs overall.

Parser	K	Repetitions	Variations	Total runs
Berkeley	10	10	4	**400**
Bikel	10	10	4	**400**
Malt	10	10	4	**400**
Stanford	10	10	4	**400**
			Total evaluation runs	**1600**

The average run duration of a single run differs, with Malt Parser being the fastest and Stanford Parser being the slowest. Data for the Raw variation of the treebank (which remains essentially the same for all variations) is shown below:

Table 5.3 Average duration for k (Raw evaluation)

	Seconds	Minutes
Berkeley	103.350	1.72
Bikel	38.471	0.64
Malt	6.417	0.11
Stanford	880.268	14.67

[10] Increases in iterations correlated strongly with increases in the duration of evaluation runs (0.99991), but moderately with increases in parser precision (0.36973). A single evaluation run k (using $k = 2$), with the parameter -maxIter set to 2000 took approximately 9 minutes to complete. Setting -maxIter to 20 000 increased this duration to more than 2 hours, but resulted in a precision increase of only 1.90 percentage points. This time cost was impractical, and thus we decided on 2000 iterations. (See figure 5.3 on page 135 for a visual overview of this impracticality.)

Stellenbosch University https://scholar.sun.ac.za

5.3.4 Metrics

Parsers have been scored on their ability to accurately reproduce syntactic graphs on unseen sentences. For constituency parsers, this meant the ability to correctly predict *node* spans and labels; for dependency parsers, the ability to correctly predict *edge* spans and labels.

Constituency parsers have been scored using the well-known Parseval metric [11]. (The Leaf Ancestor metric, an alternative to Parseval proposed by Sampson [90], has been considered but discarded early on due to reasons outlined in Appendix A.2.) Dependency parsers have been scored by calculating Labeled Attachment Scores (LAS), as outlined in [58]. Scores were calculated and logged on a macro (sentential) and micro (edge, node) level.

Constituency scores are calculated as the harmonic mean of both precision[11] and recall[12], expressed as its F1 score[13], with $F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$. Dependency scores are calculated differently: sentence level scores are determined by calculating precision, while edge label scores are calculated using the F1 metric mentioned above.

This slight difference in approaches is necessitated by the different structures generated by each type of parser. Constituency parsers are not guaranteed to predict ("recall") all nodes in a constituency tree, and therefore precision alone does not provide a true indication of their efficacy. Dependency parsers, conversely, start out with all graph nodes readily available. The node recall rate of a dependency parser is always 100%, making precision the only measurement required. However, it is still entirely possible for edge labels to be recalled incorrectly, and therefore F1 scores are calculated for individual edge labels, just as they are calculated for individual node labels during constituency parser evaluation.

Lastly there is the question of how to deal with graph roots. The root node of a constituency graph is not in the input data, and has to be generated as a prediction, while the root node of a dependency graph is a terminal node that already exists in the input data. Because of this, we measure roots differently. Both types of root nodes are measured with a precision score, but the meaning of this score differs slightly between frameworks. Root precision

[11] Number of correctly predicted nodes divided by number of nodes in candidate tree

[12] Number of correctly predicted nodes divided by number of nodes in gold tree

[13] This metric is formally called the Sørensen–Dice coefficient, but this name is not often used in NLP circles, where references to the "F1 score" are common.

Stellenbosch University https://scholar.sun.ac.za

for dependency parsers is expressed as a precision score for root nodes. Root precision for constituency parsers is expressed as a precision score for the immediate children of the root node. (This is because all root nodes in the Raw version of Swartsbank are labeled *S* by default, and precision of 100% for *S* would not teach us anything useful or interesting.)

Table 5.4 Summary of metrics used during evaluation

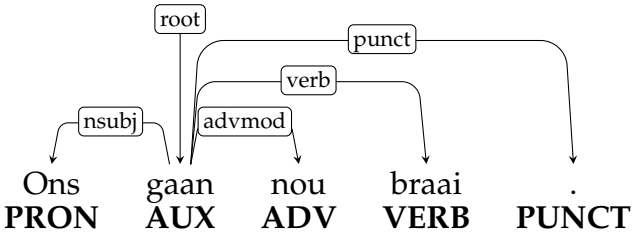
Framework	Level	Metric
Constituency	Sentence	Parseval (F1)
	Label	Parseval (F1)
	Root	Precision
Dependency	Sentence	LAS (Precision)
	Label	LAS (F1)
	Root	Precision

5.4 Evaluation pre-processing

5.4.1 Part of speech nodes

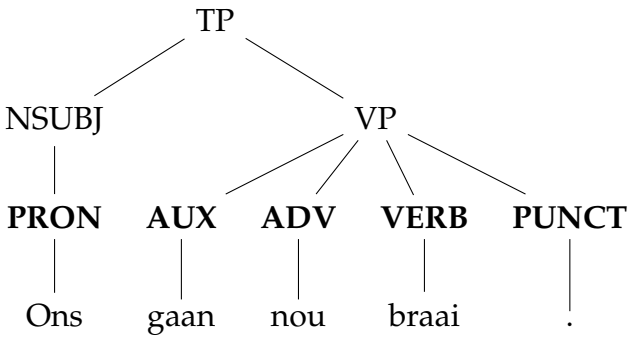
In a dependency graph, part of speech tags are node features, as illustrated in the dependency parse for the following sentence:

(102) Ons gaan nou braai
We will now barbeque
'We will barbeque sometime in the near future'



In constituency graphs, POS tags are tree nodes themselves.

Stellenbosch University https://scholar.sun.ac.za



This raises the question of how to deal with them during evaluation. Considering that almost 40% of all constituency labels in Swartsbank are part of speech tags, and that these labels are not part of what is predicted by dependency parsers, we decide to remove part of speech nodes from gold and candidate constituency trees before F1 scores are calculated. In doing so we get a clearer idea of each parser's ability to accurately predict syntactic categories.

5.4.2 Punctuation

To account for the different ways in which punctuation marks are handled in each framework, they are also ignored during the calculation of parser scores.

5.5 Summary

In this chapter we provided an overview of the experimental methods used to obtain empirical data for this study. We have done so by describing key features of the core dataset (Swartsbank), the software written to build, analyse and measure experimental results, the metrics with which we measured parser efficacy as well as the parsers employed during treebank evaluation. We also referred the reader to critiques of Afribooms and the Leaf Ancestor metric in appendices A.1 and A.2.

Stellenbosch University https://scholar.sun.ac.za

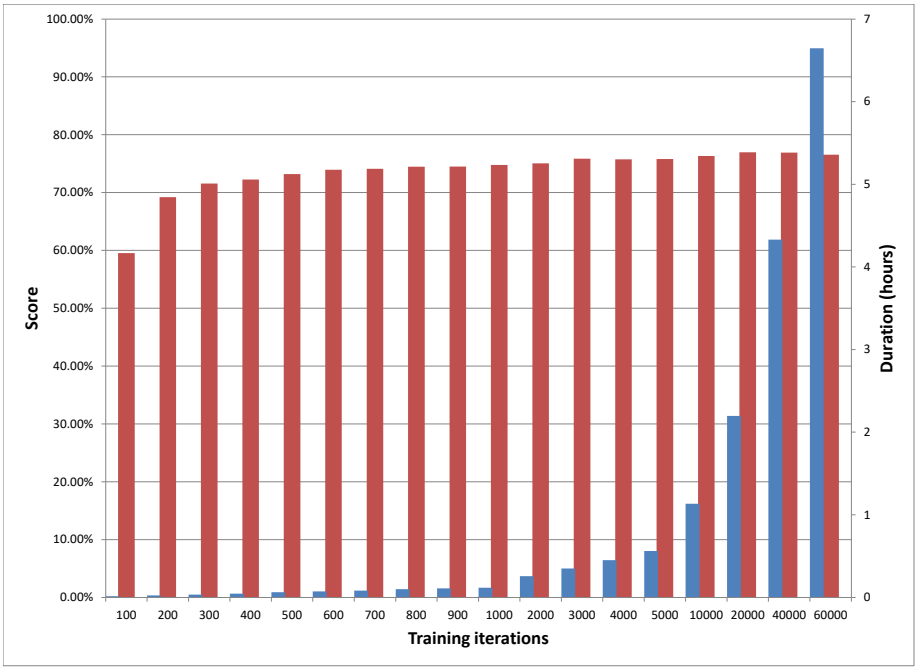


Figure 5.3: The impact of training iterations on parser score (left axis) and duration of evaluation runs (right axis) for the Stanford Parser

Stellenbosch University https://scholar.sun.ac.za

Chapter 6

Data Analysis

To test the veracity of **H0** and **H1**, we performed the experiments described in Chapter 5 and analysed the following aspects of the data logged by our evaluation software:

Table 6.1 Evaluation aspects

Aspect	Description
Parser score	The average score per parser
Root score	The average score for graph roots
Label score	The average score per label per parser
Mistake shares	The proportional make-up of types of mistakes
Unmatched elements	The nodes and edges most frequently mispredicted by each parser
Labeling errors	The node and edge labels most frequently mispredicted by each parser
Length correlation	Correlations between parser scores and sentence lengths
Frequency correlations	Correlations between label scores and label frequencies

We present the data per variation of Swartsbank that was used during these experiments: the Raw, Generative, Functional and Categorical variations. As mentioned in Chapter 5, a small subset of results are ignored because of their lack of usefulness to our investigation. These include part of speech tags, dependency labels for which the corollary in the constituency portion of Swartsbank are part of speech tags, and punctuation.

Data for scores and mistakes are presented in tables in which we visually highlight the results of the best performing parser to ease reading (see

Stellenbosch University https://scholar.sun.ac.za

for example table 6.23 on page 155). In cases where constituency and dependency parsers measure aspects unique to each framework, we visually indicate both the highest constituency and the highest dependency score. (See for example table 6.24 on page 159).

Correlation data is presented in tables and illustrated visually via scatterplots (see for example figure 6.14 on page 152) and heatmaps (see for example figure 6.10 on page 150.

Sections in which parser mistakes are highlighted are augmented by visual representations of parse trees predicted during evaluation to highlight common mistakes committed by each type of parser. In these representations, we use colour in candidate parses to distinguish between correctly predicted nodes or edges (green) and incorrectly predicted nodes or edges (red).

We begin our analysis with scores achieved by parsers on the Raw variation of Swartsbank.

6.1 Raw evaluation results

6.1.1 Parser scores

Dependency parsers outperform constituency parsers when evaluated on the Raw variation of Swartsbank. Dependency parsers (Malt, Stanford) score in the region of 80%, while constituency parsers (Berkeley, Bikel) score in the region of 70%. The best dependency result (Malt, 83.43%) is on average 10 percentage points higher than the best constituency result (Bikel 73.46%).

Table 6.2 Parser scores – Raw evaluation

Parser	Score
Berkeley	69.76%
Bikel	73.46%
Malt	83.43%
Stanford	79.32%

6.1.2 Root scores

Dependency parsers obtain higher scores than constituency parsers when predicting sentence roots, with Maltparser winning out overall. (See table

Stellenbosch University https://scholar.sun.ac.za

6.3 on page 140.) [1]

6.1.3 Label scores

6.1.3.1 Nominal labels

Dependency parsers generally predict nominal labels shared across frameworks better than constituency parsers. Malt parser obtains the highest scores overall.

Nominal subjects are predicted correctly the most (Malt: 71.03%), followed by direct objects (Malt score: 46.41%) and passive subjects (Malt: 8.14%). Parsers all but fail to predict indirect objects, with Bikel wining a small victory with a score of 0.14%. (See table 6.4 on page 140.)

Nominal labels unique to the dependency section of Swartsbank are predicted most correctly by Malt parser. Prepositional objects (*pobj*) score highest with 78.69%, followed by nominal modifiers (*nn*) with 18.32%. Rarer labels *posobj* (possessive object) and *appos* (apposition) receive F1 scores smaller than 10%. (See table 6.5 on page 140.)

The noun phrase (*NP*), the single nominal label unique to the Raw constituency section of Swartsbank, is predicted most correctly by Bikel parser. Bikel achieves a score of 71.65%, beating Berkeley by 2.7 percentage points. (See table 6.6 on page 140.)

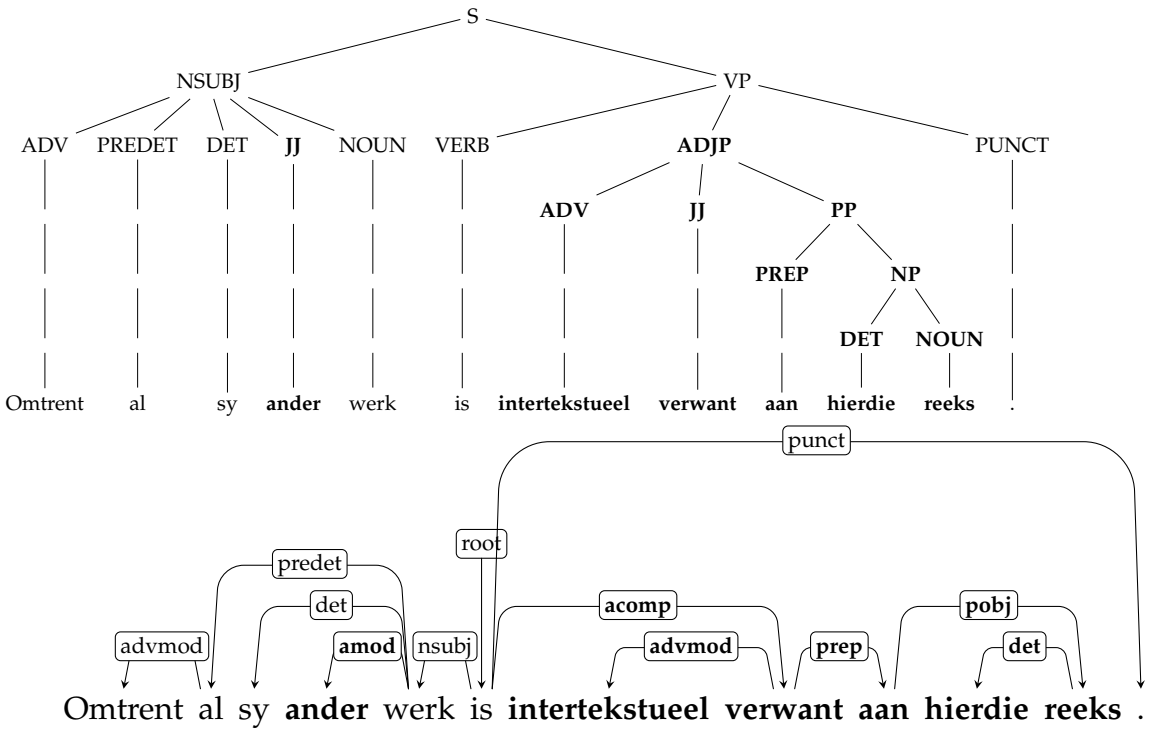
6.1.3.2 Adjectival labels

Parsers struggle to identify adjectival labels across frameworks. Scores for the dependency label *acomp* and the constituency label *ADJP* (both indicating predicative adjectival phrases) are low. The dependency label *amod* (which indicates the roots of attributive adjectives embedded inside nominal constructions) scores in the region of 49%. (See table 6.7 on page 140).

This discrepancy in adjectival scores likely has to do with *amod* being situated within the bounds of nominal phrases, while *acomp* and *ADJP* are usually indicative of more complicated structures, as illustrated with reference to sentence 366 from Swartsbank, which contains both types:

[1] The reader is reminded that this result in itself is not an indication of the efficacy of constituency parsers, since we calculate roots differently for each framework. Refer back to section 5.3.4 for details.

Stellenbosch University https://scholar.sun.ac.za



6.1.3.3 Adpositional labels

Constituency parsers obtain slightly higher scores when predicting prepositional phrases (Bikel: 67.84%), compared to dependency parsers predicting prepositional dependencies (Malt: 63.16%). (See table 6.8 on page 140).

Postpositions (labeled in Swartsbank's dependency section as *adp*) are very rare, occurring only 6 times in the treebank. This is reflected in its dependency scores, which approach zero for all parsers. (See table 6.9 on page 140.)

6.1.3.4 Possessive labels

Scores for genitive particles (dependency) and genitive phrases (constituency) are low. Malt parser obtains the highest score (9.42%) by a small margin. (See table 6.10 on page 140.)

6.1.3.5 Clausal labels

Clausal label scores are low in both frameworks. Constituency scores for different types of clauses range from 0.65% to 13.11%. Dependency scores

Stellenbosch University https://scholar.sun.ac.za

Table 6.3 Root scores – Raw evaluation

	Berkeley	Bikel	Malt	Stanford
root	—	—	92.63%	91.58%
ROOT	75.46%	77.68%	—	—

Table 6.4 Nominal Scores – Raw evaluation

	Berkeley	Bikel	Malt	Stanford
nsubj	66.41%	66.13%	71.03%	68.78%
dobj	31.14%	36.51%	46.41%	44.22%
iobj	0.10%	0.14%	0.12%	1.42%
nsubjpass	2.35%	1.71%	8.14%	5.73%

Table 6.5 Nominal scores – Raw evaluation (Dep)

	Malt	Stanford
pobj	79.50%	76.25%
nn	18.32%	17.61%
posobj	9.07%	8.87%
appos	3.49%	3.56%
npadvmod	0.35%	1.42%

Table 6.6 Nominal scores – Raw evaluation (Const)

	Berkeley	Bikel
NP	68.95%	71.65%

Table 6.7 Adjectival scores – Raw evaluation

	Berkeley	Bikel	Malt	Stanford
amod	—	—	49.57%	49.71%
acomp	—	—	15.42%	12.89%
ADJP	5.38%	12.13%	—	—

Table 6.8 Prepositional scores – Raw variation

	Berkeley	Bikel	Malt	Stanford
prep	—	—	64.02%	61.48%
PP	65.24%	67.76	—	—

Table 6.9 Postpositional scores – Raw evaluation

	Malt%	Stanford%
adp	0.72%	0.00%

Table 6.10 Possessional scores – Raw evaluation

	Berkeley	Bikel	Malt	Stanford
poss	—	—	9.42%	8.71%
GP	7.59%	6.82%	—	—

Stellenbosch University https://scholar.sun.ac.za

range from 0.06% to 17.78%. Berkeley obtains slightly higher scores than Bikel for the constituency framework. Malt obtains slightly higher scores than Stanford in the dependency framework. Labels with very low occurrences in the treebank ("MAIN_CLAUSE", "advcl", "csubj") are responsible for the worst scores which – in some cases – approach 0%. (See tables 6.11 and 6.12 on page 142.)

6.1.4 Mistakes

Most mistakes logged by parsers during Raw evaluation are matching errors – nodes or edges in gold data which are not accounted for in candidate data because they are not there, or their start and end nodes do not correlate. These account for approximately four out of every five mistakes. Labelling errors – nodes or edges with correct start and end nodes but incorrect labels – account for less than a fifth of errors. Incorrectly predicated graph roots are the rarest type of mistake (1% – 3%). (See figure 6.1 on page 142).

6.1.4.1 Unmatched nodes and edges

Constituency parser's unmatched node errors are dominated by missing nominal and prepositional phrases, and to a lesser extent clauses. Nominal constituents account for approximately 28% of all unmatched errors, prepositional constituents for approximately 20% and clausal constituents for between 13% and 16%. This pattern holds for Berkeley as well as Bikel, as noted in tables 6.13 (page 145) and 6.14 (page 145). To illustrate these mistakes, note the candidate parse assigned to sentence 780 in which the boundaries of prepositional and nominal phrases are misjudged:

(103) Bird Custard is later as custardpoeier in winkels verkoop
Bird Custard was later as custard-powder in shops sold
'Bird Custard was later sold as custard powder in shops'

Stellenbosch University https://scholar.sun.ac.za

Table 6.11 Constituency clausal scores – Raw evaluation

	Berkeley	Bikel
SUB_CLAUSE	14.27%	13.05%
COORD_CLAUSE	11.51%	12.14%
INF_CLAUSE	9.28%	8.28%
REL_CLAUSE	10.37%	9.59%
MAIN_CLAUSE	0.83%	0.81%

Table 6.12 Dependency clausal scores – Raw evaluation

	Malt	Stanford
ccomp	17.80%	15.38%
xcomp	10.78%	13.76%
rcmod	12.50%	11.18%
advcl	2.30%	2.14%
csubj	0.00%	0.14%

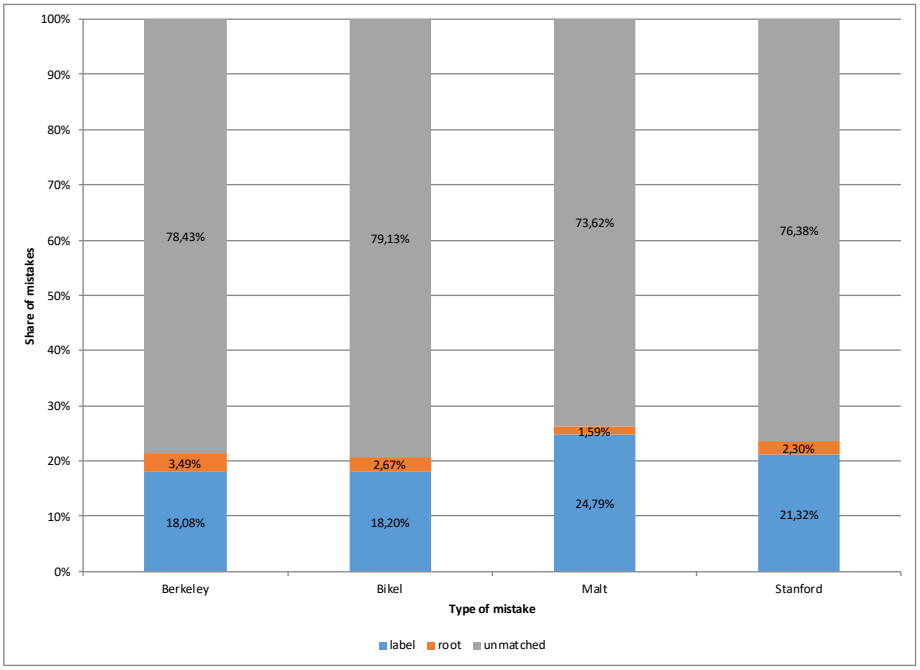


Figure 6.1: Distribution of mistakes in Raw evaluation runs

Stellenbosch University https://scholar.sun.ac.za

Figure 6.2: Sentence 780 – Gold standard, constituency

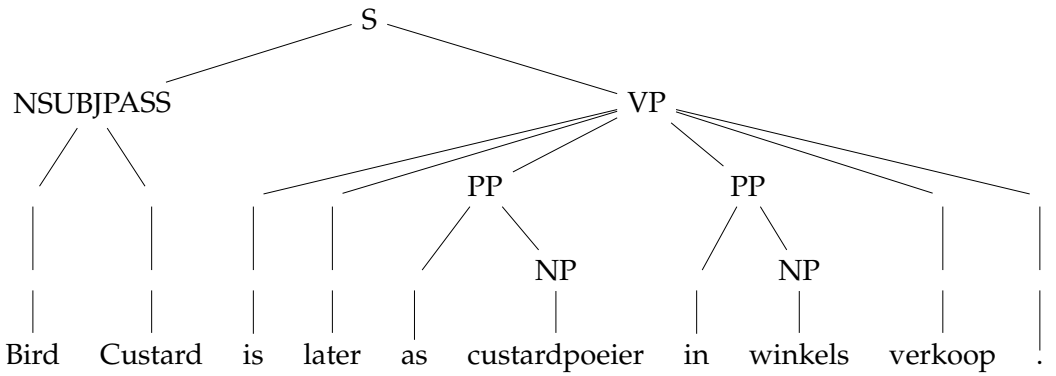
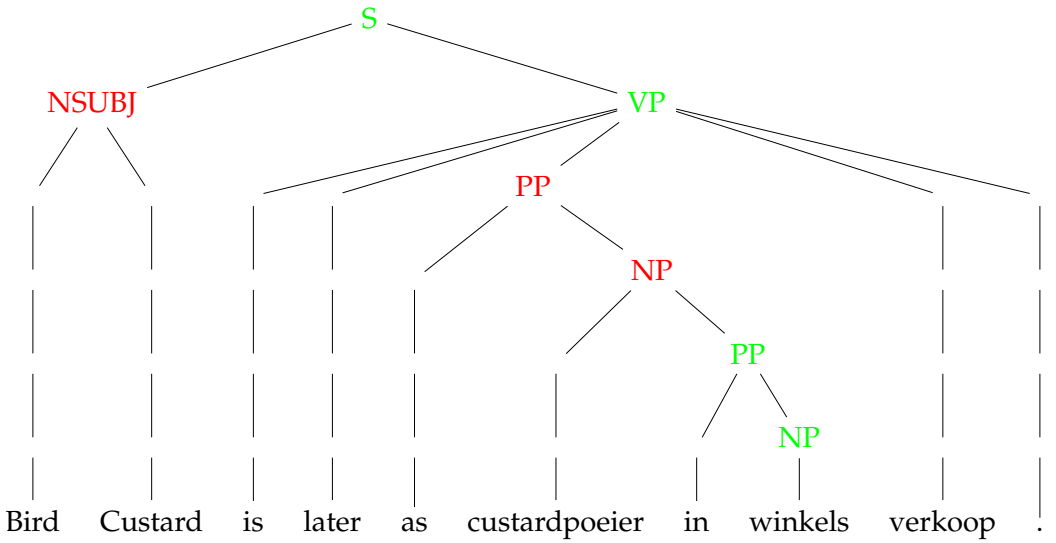


Figure 6.3: Sentence 780 – Candidate parse, Bikel



Dependency parsers make different types of unlabeled mistakes than constituency parsers. The majority of errors are missed governors of prepositional catenas (*prep*), adverbial modifiers (*advmod*) and coordinated graph nodes (*conj*). Nominal and clausal nodes are problems to a lesser extent. Notably, clause markers (*mark*) are also in the list of top 10 matching errors, accounting for just over 4% of wrongly matched edges. Malt and Stanford parser mistakes share a high degree of similarity, with the exception of verbal and negation particles (*particle*) in Stanford parser results which account for just over 5% of wrongly matched edges. (See table 6.15 on page 145). This

Stellenbosch University https://scholar.sun.ac.za

is illustrated in a candidate parse for sentence 908, in which edges labeled *advmod* and *prep* are attached to the wrong graph nodes:

(104) Hierdie toename in plantegroei maak die water minder
This increase in plant-growth make the water less
deursigtig, sodat die sonlig nie tot op die bodem kan
transparent so-that the sunlight not to the bottom can permeate
deurdring nie.
not
'This increase in vegetation makes the water less transparent, so sunlight cannot reach the bottom'

Figure 6.4: Sentence 908 – Gold standard, dependency

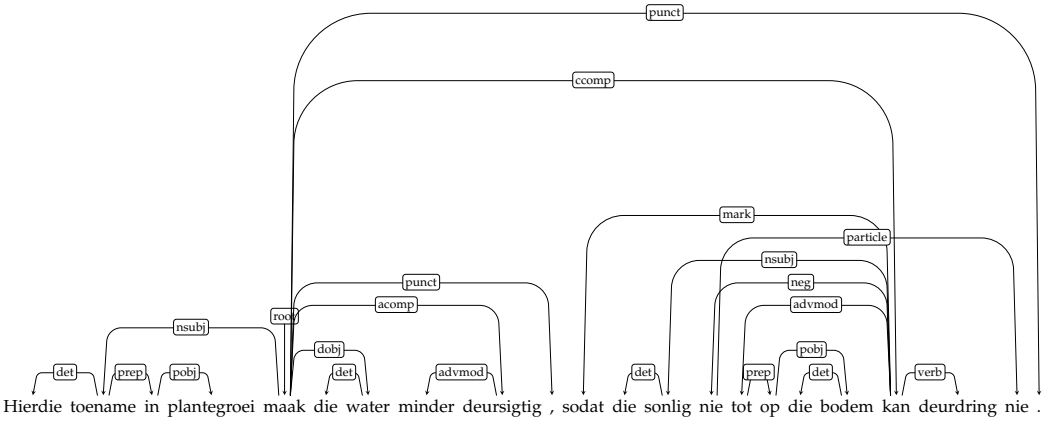
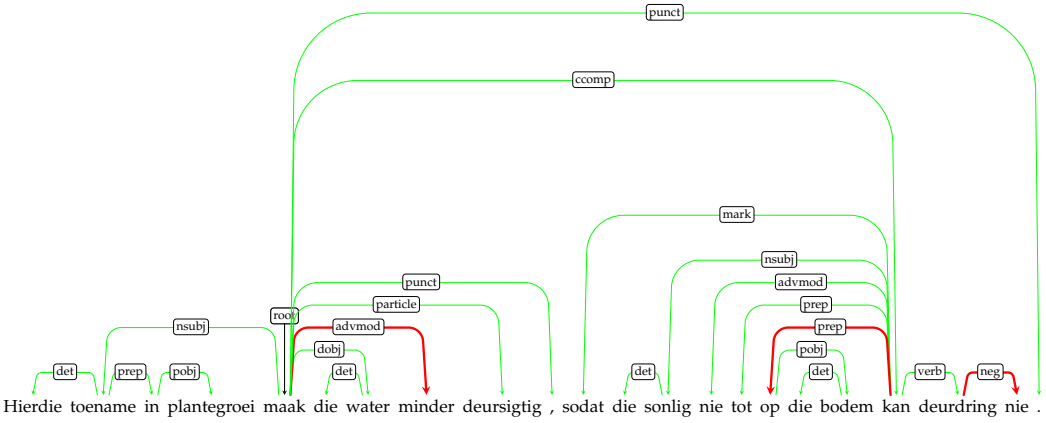


Figure 6.5: Sentence 908 – Candidate parse, Stanford



Stellenbosch University https://scholar.sun.ac.za

Table 6.13 Berkeley parser top 10 unmatched node types (Raw evaluation)

Missed node	% of match mistakes
NP	28.14%
PP	20.08%
DOBJ	11.66%
VP	7.59%
NSUBJ	7.56%
ADJP	6.80%
SUB_CLAUSE	4.60%
REL_CLAUSE	4.54%
COORD_CLAUSE	2.27%
INF_CLAUSE	2.22%

Table 6.14 Bikel parser top 10 unmatched node types (Raw evaluation)

Missed node	% of match mistakes
NP	28.98%
PP	19.61%
DOBJ	9.14%
NSUBJ	6.94%
VP	6.14%
ADJP	6.09%
SUB_CLAUSE	5.76%
REL_CLAUSE	5.19%
INF_CLAUSE	3.05%
COORD_CLAUSE	2.73%

Table 6.15 Malt parser top 10 unmatched edge types (Raw evaluation)

Missed node	% of match mistakes
prep	21.85%
advmod	9.71%
conj	7.40%
dobj	5.79%
pobj	5.24%
nsubj	4.82%
ccomp	4.81%
xcomp	4.57%
mark	4.34%
rcmod	4.32%

Stellenbosch University https://scholar.sun.ac.za

6.1.4.2 Labelling mistakes

The most frequently occurring labelling mistakes in constituency parser results are mispredicted nominal labels. In these cases one type of commonly occurring nominal label is mistaken for another type of commonly occurring label. This specific type of error is so common that it accounts for more than 65% of all labelling mistakes. (See the example of sentence 596) below, in which WH-movement trips up Bikel parser's ability to correctly identify nominal labels.) The second most common type of mistake – clausal labels being confused with each other – barely reaches 10% of all labelling errors. The remainder of labelling errors do not provide any significant insights for improving the input data for either constituency parser. (See tables 6.18 and 6.17 on page 147.)

(105) Nou wat was dit dan?
Now what was it then?
'So what was that, then?'

Figure 6.6: Sentence 596 – Gold standard, constituency

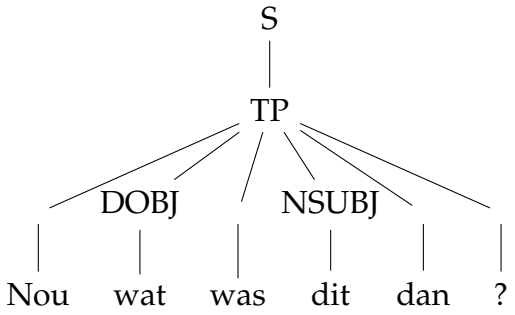
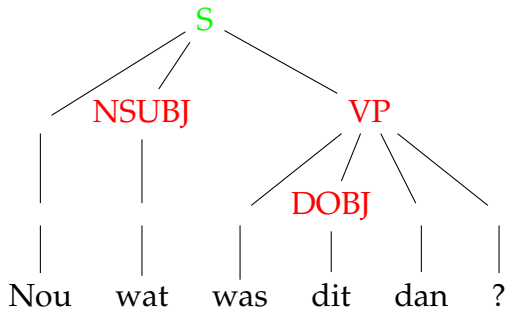


Figure 6.7: Sentence 596 – Candidate parse, Bikel



Stellenbosch University https://scholar.sun.ac.za

Table 6.16 Stanford parser top 10 unmatched edge types (Raw evaluation)

Missed node	% of match mistakes
prep	18.82%
advmod	8.98%
conj	7.16%
dobj	5.77%
pobj	5.59%
particle	5.09%
ccomp	5.00%
nsubj	4.82%
mark	4.81%
xcomp	4.42%

Table 6.17 Berkeley parser top 10 label mistakes (Raw evaluation)

Correct	Guess	% of node mistakes
PATIENT	NSUBJ	16.04%
NSUBJ	PATIENT	10.17%
DOBJ	NSUBJ	7.63%
NSUBJ	DOBJ	5.97%
DOBJ	NP	5.80%
NP	DOBJ	3.70%
NSUBJ	NP	3.65%
COORD_CLAUSE	SUB_CLAUSE	3.19%
NP	NSUBJ	2.90%
SUB_CLAUSE	COORD_CLAUSE	2.75%

Table 6.18 Bikel parser top 10 label mistakes (Raw evaluation)

Correct	Guess	% of node mistakes
NSUBJ	DOBJ	14.74%
PATIENT	NSUBJ	14.06%
DOBJ	NSUBJ	9.55%
NP	DOBJ	6.10%
DOBJ	NP	5.90%
NSUBJ	NP	5.76%
NSUBJ	PATIENT	4.82%
NP	NSUBJ	4.47%
SUB_CLAUSE	COORD_CLAUSE	3.83%
COORD_CLAUSE	SUB_CLAUSE	3.09%

Stellenbosch University https://scholar.sun.ac.za

Dependency parsers commit the same types of mistakes, but in different proportions. Stanford also struggles to correctly label negative adverbs (confusing them for adverbial modifiers) and somewhat struggles to correctly label closed clausal complements (confusing them with verbal dependencies). (See table 6.19 on page 150 and table 6.20 on page 150.) An example of Stanford's struggle to correctly label negators can be found in a candidate parse for sentence 869:

(106) Hy het nooit getrou nie
He has never married not
'He never got married'

Figure 6.8: Sentence 869 – Gold standard

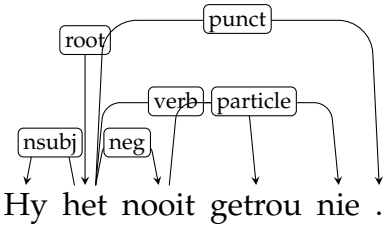
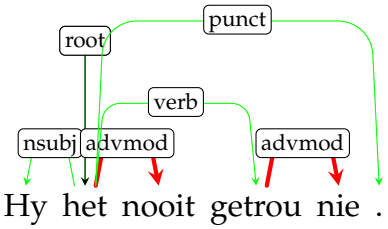


Figure 6.9: Sentence 869 – Candidate parse, Stanford



6.1.5 Correlations

6.1.5.1 Sentence lengths

Sentence lengths have a negative impact on parser results, as is evident from the length / score correlation coefficient for candidate parses. This correlation is most pronounced in the results of Bikel parser. Longer sentences are therefore more likely to receive lower scores – a result which is not surprising within the sphere of NLP.

Stellenbosch University https://scholar.sun.ac.za

Table 6.21 Sentence length / score correlation coefficients (Raw)

Parser	Coefficient
Berkeley	-0.19131
Bikel	-0.25626
Malt	-0.2056
Stanford	-0.19801

Heatmaps of scores versus sentence lengths confirm this visually, as seen in figures 6.10 (page 150), 6.11 (page 151), 6.12 (page 151) and 6.13 (page 154).

6.1.5.2 Label frequency

Strong correlations between label scores and label frequencies exist in all parser results. This correlation is slightly stronger in dependency parsers than in constituency parsers:

Table 6.22 Label frequency / score correlation coefficient (Raw variation)

Parser	Coefficient
Berkeley	0,84659
Bikel	0,84118
Malt	0,91071
Stanford	0,90845

This correlation is illustrated visually in figures 6.14 and 6.15.

Stellenbosch University https://scholar.sun.ac.za

Table 6.19 Malt parser top 10 label mistakes (Raw evaluation)

Correct	Guess	% of label mistakes
nsubj	dobj	11.12%
xcomp	ccomp	9.08%
dobj	nsubj	7.66%
patient	nsubj	6.69%
ccomp	xcomp	6.04%
nsubj	patient	4.45%
xcomp	rcmod	2.83%
npadvmod	dobj	2.53%
det	amod	2.21%
prep	dobj	2.17%

Table 6.20 Stanford parser top 10 label mistakes (Raw evaluation)

Correct	Guess	% of label mistakes
neg	advmod	8.67%
nsubj	dobj	8.42%
patient	nsubj	8.41%
dobj	nsubj	7.64%
xcomp	ccomp	3.81%
nsubj	patient	3.65%
ccomp	xcomp	2.86%
ccomp	verb	2.37%
det	amod	2.04%
mark	nsubj	1.74%

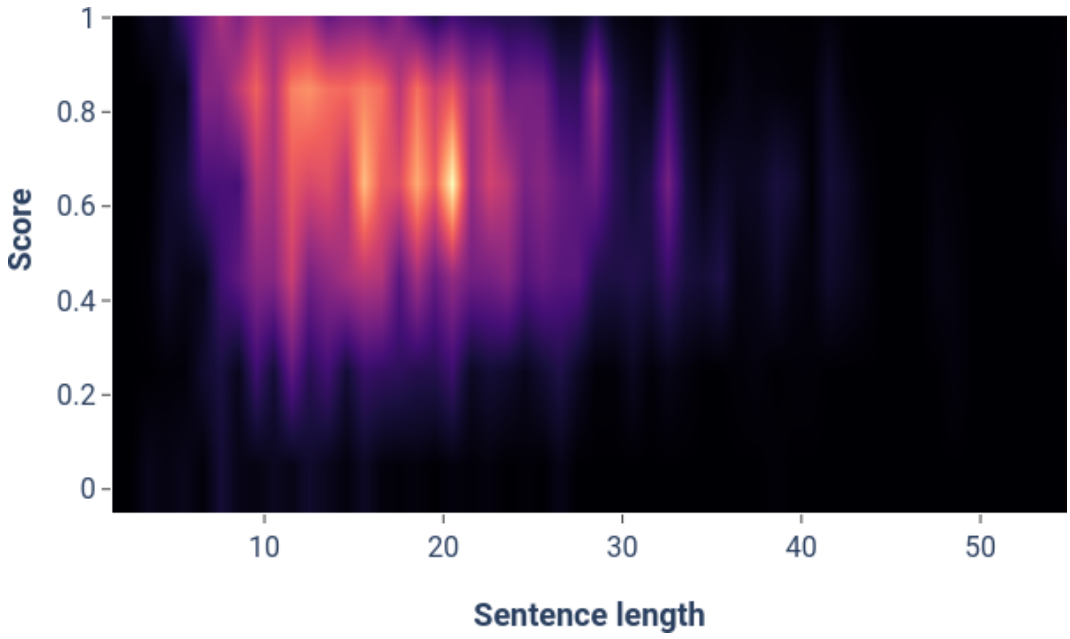


Figure 6.10: Scores versus lengths (Berkeley – Raw)

Stellenbosch University https://scholar.sun.ac.za

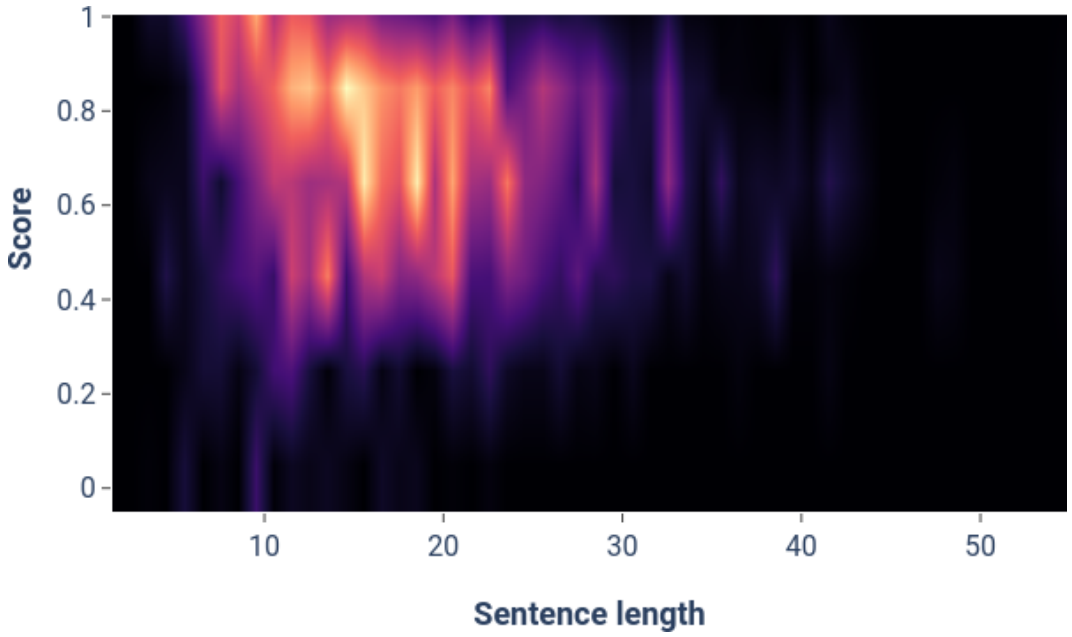


Figure 6.11: Scores versus lengths (Bikel – Raw)

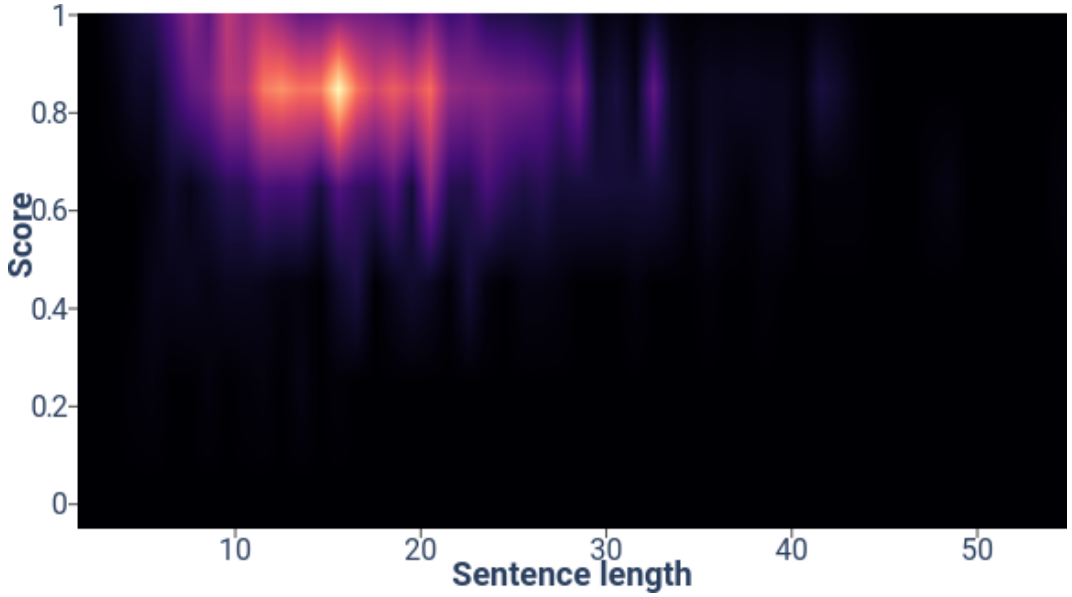


Figure 6.12: Scores versus lengths (Malt – Raw)

Stellenbosch University https://scholar.sun.ac.za

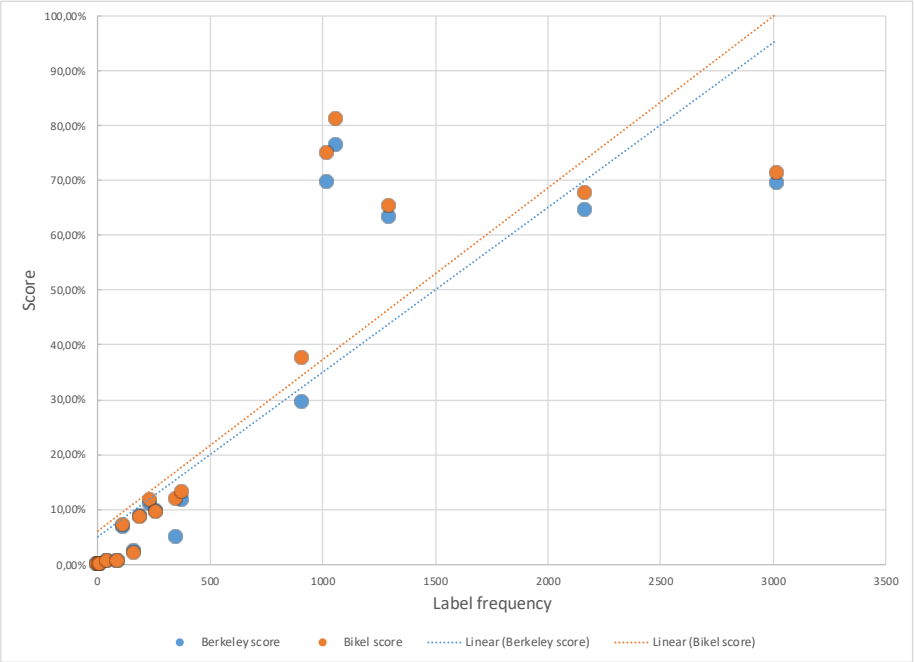


Figure 6.14: Label frequency impact on scores (Constituency, Raw)

Stellenbosch University https://scholar.sun.ac.za

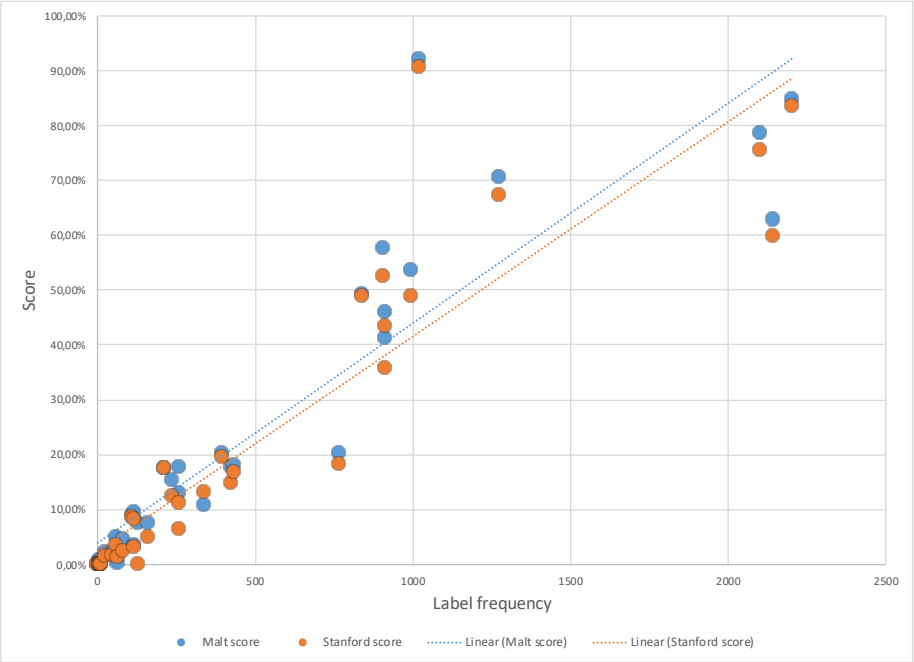


Figure 6.15: Label frequency impact on scores (Dependency, Raw)

6.1.6 Summary

Dependency parsers obtain better scores than constituency parsers when evaluated on the Raw variation of Swartsbank. About 4 out of every 5 parser mistakes are the incorrect prediction of nodes (dependency) or edges (constituency). The remaining 1/5th is incorrect label and root predictions. The frequency of the types of nodes and edges in Swartsbank strongly correlates with each parser's ability to predict each of these nodes and edges. A weak negative correlation exists between sentence lengths and parser accuracy.

Stellenbosch University https://scholar.sun.ac.za

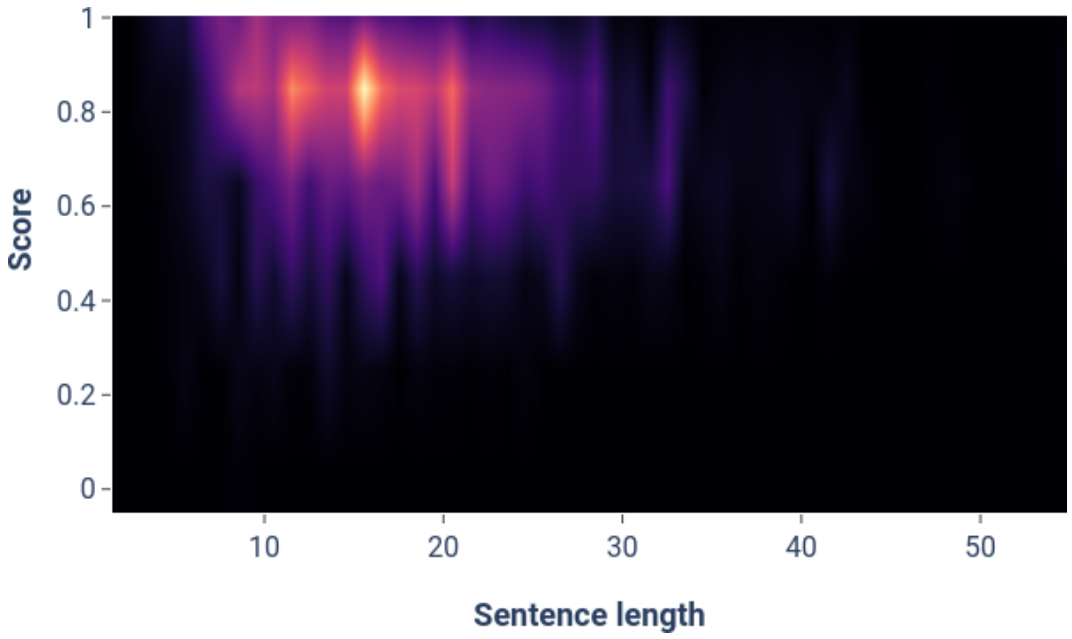


Figure 6.13: Scores versus lengths (Stanford – Raw)

Stellenbosch University https://scholar.sun.ac.za

6.2 Generative evaluation results

6.2.1 Parser scores

Converting the constituency section of Swartsbank to the Generative format specified in Chapter 4 does not significantly alter constituency scores. Dependency parsers still perform better than constituency parsers, and Malt parser outperforms all other parsers. (Berkeley parser does manage an approximately 2% increase over its Raw score, but it is not clear whether this is a statistically significant change.)

Table 6.23 Parser scores – Standard variation

Parser	Score
Berkeley	71.69%
Bikel	72.63%
Malt	84.65%
Stanford	80.23%%

The dependency section of Swartsbank is not altered during Generative conversion, and therefore dependency parser scores do not change during Generative evaluation. We therefore only list constituency results in the sections below.

6.2.2 Root scores

Constituency parsers are not able to beat dependency parsers at the prediction of root nodes. Even so, the Generative structure seems to have a noticeable effect on their efficacy. Berkeley parser performs slightly better compared to Raw evaluation, and Bikel parser performs slightly worse. (See table 6.24 on page 159.)

6.2.3 Label scores

6.2.3.1 Nominal labels

The Generative variation of Swartsbank has only one nominal label, namely *NP*. Results for this label show significant improvement when compared to those in Raw evaluation. Both parsers achieve scores of more than 80%,

Stellenbosch University https://scholar.sun.ac.za

with Bikel parser achieving the highest score (85.92%). This increase of 15 percentage points in the scores of both parsers likely has to do with the fact that multiple labels used for the same set of nodes in the Raw variation (indicating subjects, objects and passive agents) are all merged into a single label (NP) in the generative variation. This increases its frequency, which increases its scores. (See table 6.25 on page 159.)

6.2.3.2 Adjectival labels

Scores for labels indicating adjectives do not significantly change when compared to Raw evaluation. (See table 6.26 on page 159).

6.2.3.3 Adpositional labels

Dependency scores for adpositional labels do not significantly change during Generative evaluation. (See table 6.27 on page 159).

6.2.3.4 Possessive labels

No significant change in possessive labels are encountered in the constituency section when compared to Raw evaluation. (See table 6.28 on page 159.)

6.2.3.5 Clausal labels

The two labels indicating clausal structures in the Generative variation (*CP*, *TP*) are predicted correctly about half of the time. Berkeley parser is approximately 5 percentage points better at predicting CP than Bikel parser. Conversely, Bikel parser is approximately 6 percentage points better than Berkeley parser at predicting TP. (See tables 6.29 on page 159.) The much higher scores for clausal structures (compared to Raw evaluation) is likely due to clausal nodes being described by only two annotations, rather than the five different annotations used in the Raw variant. It might also be possible that the slightly more hierarchical structure of CP + TP happens to be favoured by the mechanics of constituency parsers.

Stellenbosch University https://scholar.sun.ac.za

6.2.4 Mistakes

Errors made by constituency parsers during Generative evaluation contain more unmatched nodes (9 out of 10) than Raw evaluation (8 out of 10). This is likely because of the smaller label set, which results in less labelling errors. (See figure 6.16 on page 160).

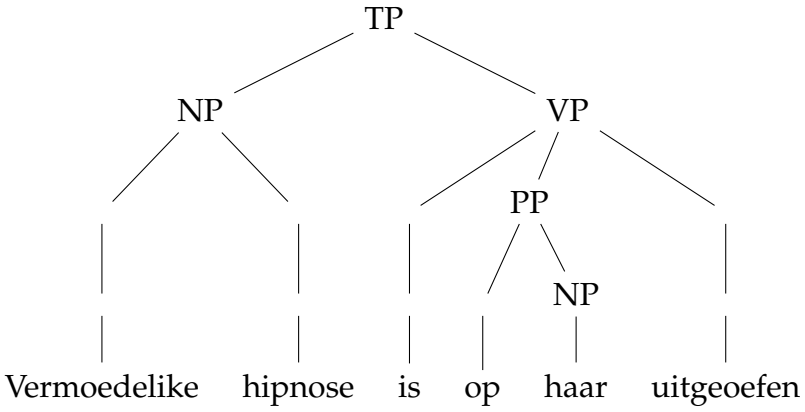
6.2.4.1 Unmatched nodes and edges

Unmatched nodes in generative evaluation exhibit the same pattern as those in Raw evaluation. (See tables 6.30 on page 159 and 6.31 on page 160.) The pattern holds for both Berkeley and Bikel parsers, albeit in different proportions than in Raw evaluation results. The single most missed node is *NP* (Berkeley 46.37% of errors, Bikel 42.34% of errors) followed by *PP* (Berkeley 19.29%, Bikel 17.73%) and *TP* (Berkeley 11.60%, Bikel 16.15%) and *CP* (Berkeley 8.77%, Bikel 11.66%). We hypothesise that the different proportions is a side effect of the smaller constituency label set found in the Generative variation of Swartsbank.

The Berkeley parser often predicts boundaries that are too wide for NPs. See for example the candidate parse for sentence 317 in which the boundary of the *PP* and right most *NP* (*haar*) is stretched to include the verb (*uitgeoefen*):

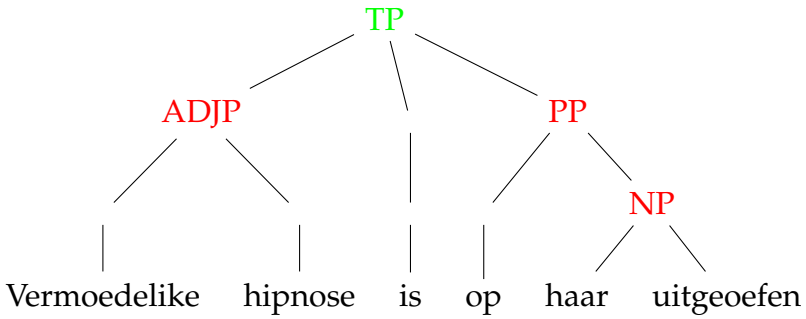
(107) Vermoedelike hipnose is op haar uitgeoefen
Suspected hypnosis PAST-be on her practiced
'Suspected hypnosis has been exercised on her'

Figure 6.17: Sentence 317: Generative gold standard, constituency



Stellenbosch University https://scholar.sun.ac.za

Figure 6.18: Sentence 317: Berkely Generative candidate



6.2.4.2 Labelling mistakes

There is some overlap between the labelling errors made by constituency parsers. Both parsers confuses *CP* for *TP*, and *NP* for *PP*. Other similarities also exist, but do not tell us anything significant or interesting. Rather, it is an unintended side-effect of the Generative variation's much smaller label set, in which the odds of labels being confused should be higher purely because there are less of them. (See tables 6.32 on page 161 and 6.33 on page 161.)

6.2.5 Correlations

6.2.5.1 Sentence lengths

As during Raw evaluation, sentence lengths correlate negatively with sentence scores.

Table 6.34 Sentence length / score correlation coefficients (Generative)

Parser	Coefficient
Berkeley	-0.15641
Bikel	-0.23567

Heatmaps of scores versus sentence lengths confirm this visually, as seen in figures 6.10 (page 150), 6.11 (page 151), 6.12 (page 151) and 6.13 (page 154).

Stellenbosch University https://scholar.sun.ac.za

Table 6.24 Root scores – Generative variation

	Berkeley	Bikel	Malt	Stanford
root	—	—	92.67%	91.20%
ROOT	78.01%	71.73%	—	—

Table 6.25 Nominal scores – Generative evaluation (Const)

	Berkeley	Bikel
NP	82.78%	85.92%

Table 6.26 Adjectival scores – Generative evaluation

	Berkeley	Bikel
ADJP	5.63%	11.85%

Table 6.27 Prepositional scores – Generative evaluation

	Berkeley	Bikel
PP	66.41%	68.02%

Table 6.28 Possessional scores – Generative evaluation

	Berkeley	Bikel
GP	8.10%	7.23%

Table 6.29 Constituency clausal scores – Generative evaluation

	Berkeley	Bikel
CP	49.36%	44.24%
TP	55.91%	50.72%

Table 6.30 Berkeley parser top 10 unmatched node types (Generative evaluation)

Missed node	% of match mistakes
NP	46.37%
PP	19.29%
TP	11.60%
CP	8.77%
ADJP	7.32%
VP	3.68%
ADVP	1.98%
GP	0.91%
EXPL	0.07%

Stellenbosch University https://scholar.sun.ac.za

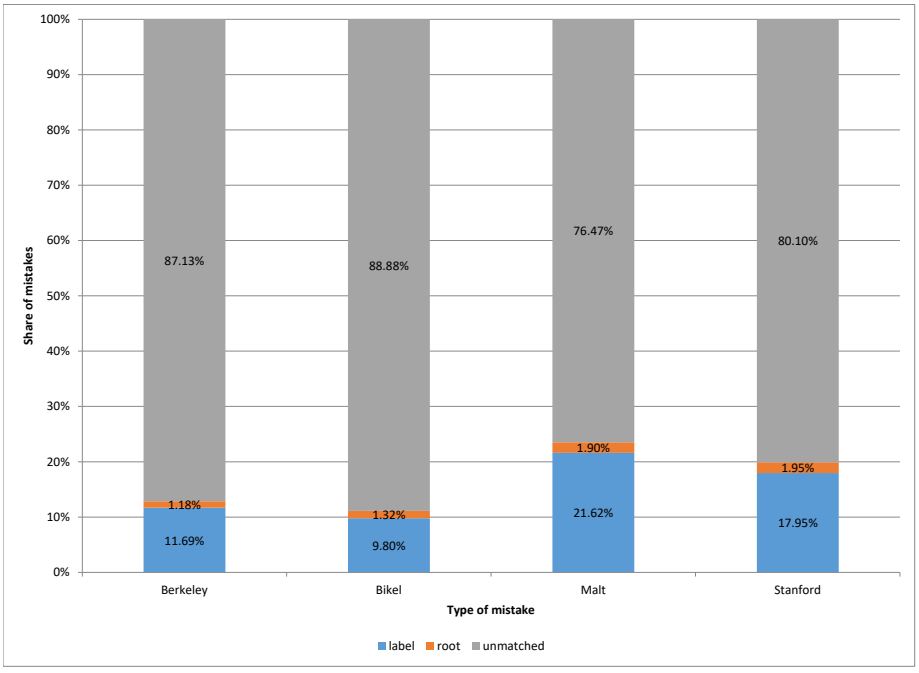


Figure 6.16: Distribution of mistakes in Generative evaluation runs

Table 6.31 Bikel parser top 10 unmatched node types (Generative evaluation)

Missed node	% of match mistakes
NP	42.34%
PP	17.73%
TP	16.15%
CP	11.66%
ADJP	5.59%
VP	3.02%
ADVP	2.21%
GP	1.23%
EXPL	0.07%

Stellenbosch University https://scholar.sun.ac.za

Table 6.32 Berkeley parser top 10 label mistakes (Generative evaluation)

Correct	Guess	% of node mistakes
TP	CP	12.88%
ADJP	NP	9.75%
PP	NP	8.78%
CP	TP	8.45%
NP	PP	8.19%
NP	ADJP	6.22%
NP	TP	6.10%
TP	NP	5.36%
VP	TP	4.50%
TP	VP	4.24%

Table 6.33 Bikel parser top 10 label mistakes (Generative evaluation)

Correct	Guess	% of node mistakes
PP	NP	15.99%
TP	CP	12.58%
CP	TP	11.48%
NP	TP	7.80%
TP	NP	6.57%
VP	TP	5.95%
NP	PP	5.89%
TP	VP	5.77%
CP	NP	3.93%
NP	CP	2.79%

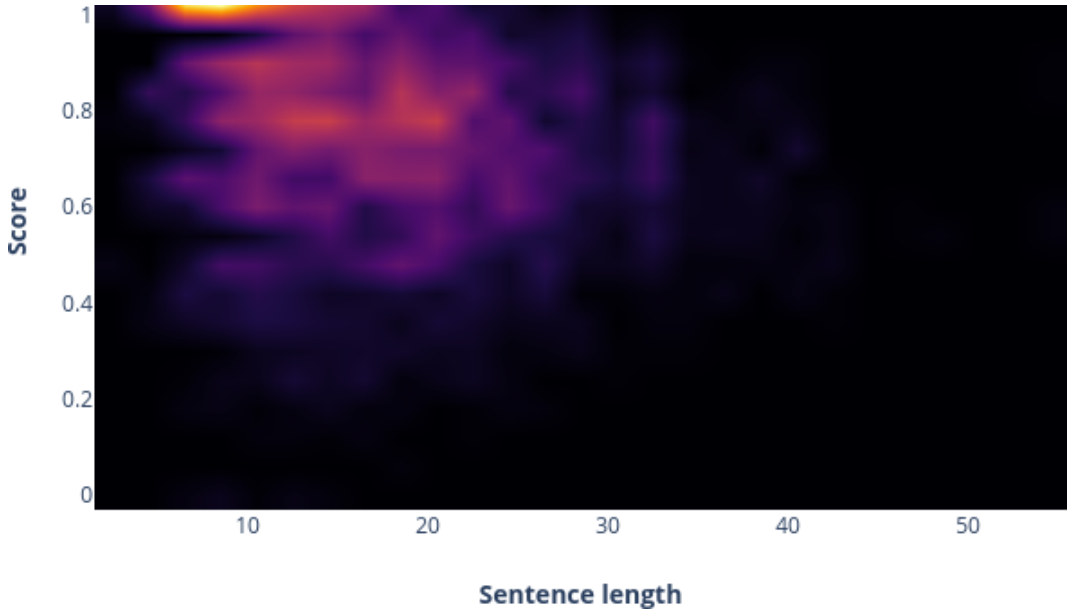


Figure 6.19: Scores versus lengths (Berkeley – Generative)

Stellenbosch University https://scholar.sun.ac.za

6.2.5.2 Label frequency

Strong correlations between label scores and label frequencies exist in Generative evaluation results:

Table 6.35 Label frequency / score correlation coefficient (Raw variation)

Parser	Coefficient
Berkeley	0.84017
Bikel	0.85466

This correlation is illustrated visually in figure 6.21.

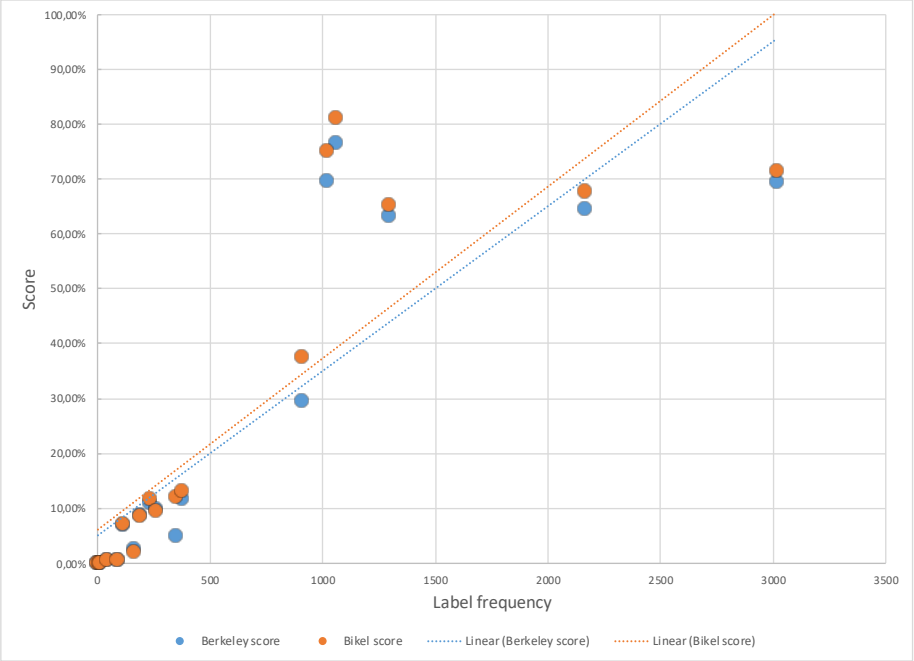


Figure 6.21: Label frequency impact on scores (Constituency, Raw)

6.2.6 Summary

The generative label set allows Berkeley and Bikel parsers to predict *NP*s and *PP*s better. Even so, it does not provide better overall scores when compared to dependency parser results. Unmatched graph nodes make up a bigger

Stellenbosch University https://scholar.sun.ac.za

proportion of the errors committed by each constituency parser. We note that this is likely because of the Generative variation's smaller label set. As during Raw evaluation, a weak negative correlation exists between sentence lengths and parser scores.

Stellenbosch University https://scholar.sun.ac.za

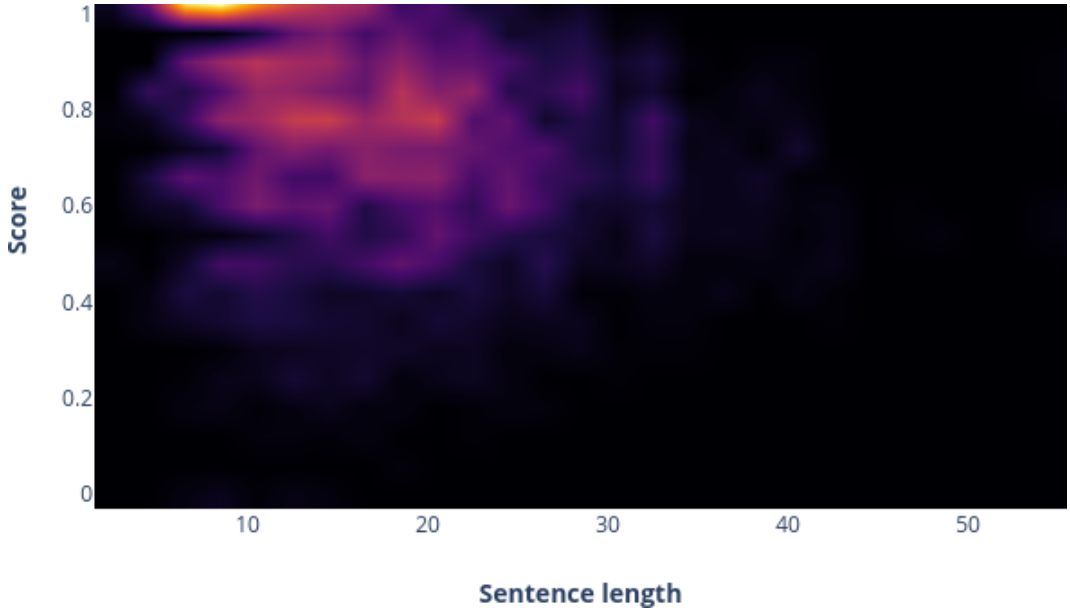


Figure 6.20: Scores versus lengths (Bikel – Generative)

Stellenbosch University https://scholar.sun.ac.za

6.3 Functional evaluation results

6.3.1 Parser scores

Converting the constituency labels in Swartsbank to the functional labels found in the dependency section does not improve the results of constituency parsers in comparison to the scores obtained in Raw or evaluation. It has, in fact, a slightly detrimental effect on constituency scores.[2]

Table 6.36 Parser scores – Functional evaluation

Parser	Raw	Functional
Berkeley	69.76%	64.86%
Bikel	73.46%	67.90%

6.3.2 Root scores

The Bikel root node score slightly decreases (-1.75%) during Functional evaluation. It is unclear whether this decrease is statistically significant. No real effect on Berkeley parser's root scores is observed. (See table 6.37 on page 169.) Neither parser sees any drastic improvements to their ability to predict the immediate children of root nodes. Given that functional evaluation does not change their labels, this outcome is not surprising.

6.3.3 Label Scores

6.3.3.1 Nominal labels

Functional nominal labels in the constituency label set follow the same trend as in the Raw version of Swartsbank: less frequently occurring labels obtain lower scores. Bikel scores better than Berkeley overall, but both still fare worse than their dependency counterparts, and are unable to best Malt parser. (See table 6.38 on page 169). The most correctly predicted label is *pobj*.

[2]Malt and Stanford parsers experience no change, since they are evaluated on the same sets of data.

Stellenbosch University https://scholar.sun.ac.za

6.3.3.2 Adjectival labels

Functionally adjusted adjectival labels in the constituency label set suffer the same fate as their nominal counterparts: after adjustment, dependency parsers still perform better. As during Raw evaluation, attributive labels (*amod*) fare better than predicative labels (*acomp*). (See table 6.39 on page 169).

6.3.3.3 Adpositional labels

Functional conversion does not affect prepositional or adpositional labels, and therefore has no effect on its scores.

6.3.3.4 Possessional labels

Functional conversion does not affect possessional labels, and therefore has no effect on its scores.

6.3.3.5 Clausal labels

Scores for clausal labels remain low, and no clear winner among the frameworks emerge. Constituency parsers have a slight upper hand over dependency parsers for the scores of the frequently occurring open clausal complement (*xcomp*), as well as the infrequently occuring clausal subject (*csubj*). Malt Parser performs slightly better on closed clausal complements (*ccomp*) and relative clauses (*rcmod*). (See table 6.40 on page 169.)

6.3.4 Mistakes

A functional label set does not impact the composition of error types, which retain roughly the same proportions of the mistakes made during Raw evaluation as indicated in figure 6.22 on page 167.

Stellenbosch University https://scholar.sun.ac.za

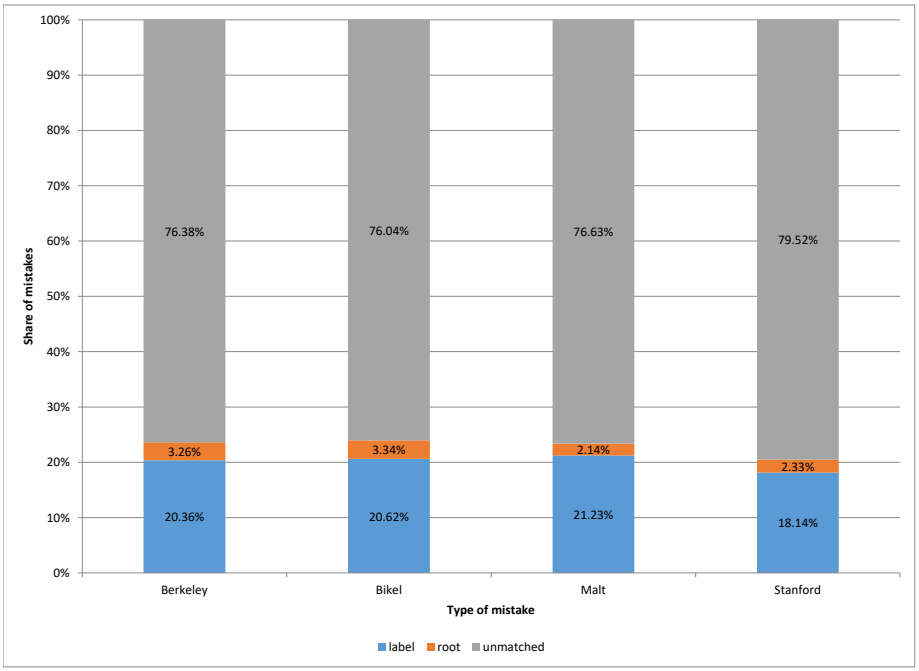


Figure 6.22: Distribution of mistakes in Functional evaluation runs

6.3.4.1 Unmatched nodes and edges

Prepositional, nominal and clausal nodes still account for the largest share of missed nodes during Functional evaluation, except that there are fewer nominal errors in total, and *PP* now tops the chart. This outcome is expected, given that *NP* has essentially been split into various sub-labels.) Prepositional objects (POBJ) account for the biggest share of missed nominal nodes. (See table 6.41 on page 169 and table 6.42 on page 172). An example of missed prepositional nodes is found in the Bikel candidate parse for sentence 1007 (notice also the missed NPs in the Berkeley candidate parse):

(108) U moet egter self vir hierdie gees in u daarvoor vra.
You must however self for this spirit in thou it-for ask.
'You must however ask the spirit in yourself for this.'

Stellenbosch University https://scholar.sun.ac.za

Figure 6.23: Sentence 1007 – Gold standard, dependency

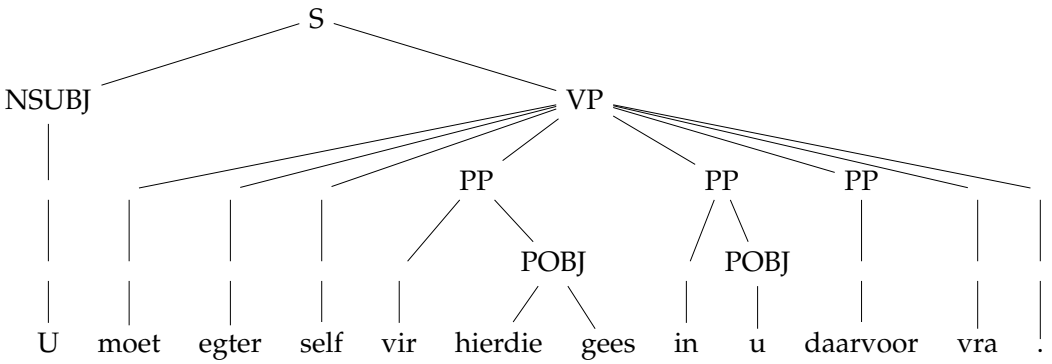


Figure 6.24: Sentence 1007 – Bikel parse

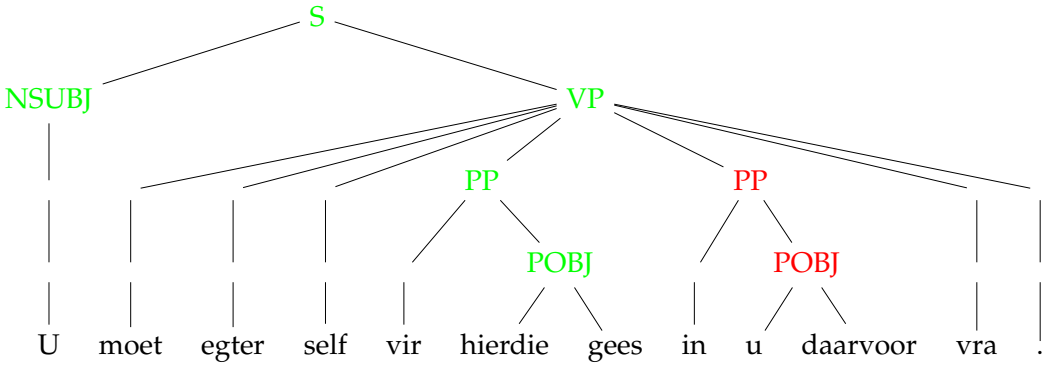
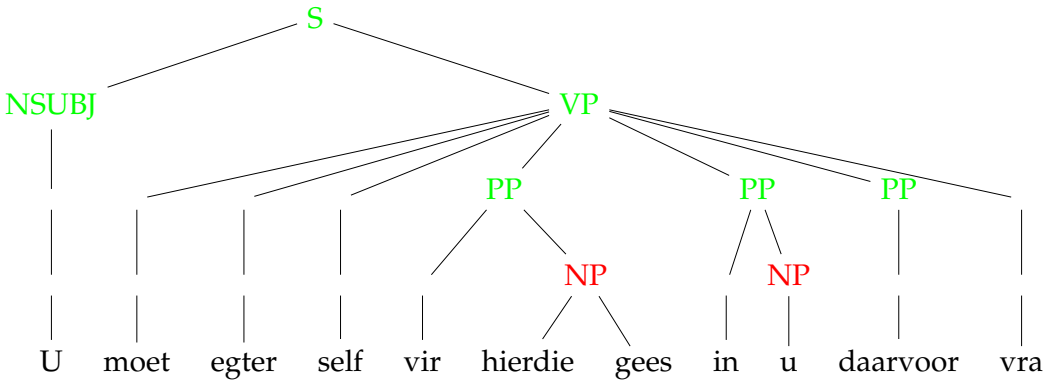


Figure 6.25: Sentence 1007 – Berkeley parse



The label set for Malt and Stanford evaluation data does not change during Functional transformation, and therefore their mistakes are not discussed in this section.

Stellenbosch University https://scholar.sun.ac.za

Table 6.37 Root scores – Functional evaluation

	Berkeley	Bikel
ROOT	69.20%	73.34%

Table 6.38 Nominal Scores – Functional evaluation

	Berkeley	Bikel	Malt	Stanford
pobj	65.50%	69.66%	78.84%	75.18%
nsubj	62.30%	65.20%	71.22%	67.90%
dobj	28.77%	37.12%	46.51%	43.18%
posobj	6.59%	6.91%	9.13%	8.49%
patient	3.25%	2.26%	7.80%	5.11%
appos	2.06%	3.32%	3.49%	3.23%
npadvmod	1.01%	1.00%	0.37%	1.20%
iobj	0.02%	0.11%	0.14%	0.00%

Table 6.39 Adjectival scores – Functional evaluation

	Berkeley	Bikel	Malt	Stanford
acomp	5.11%	10.99%	15.30%	12.74%
amod	0.59%	1.35%	49.09%	48.89%

Table 6.40 Clausal scores – Functional evaluation

	Berkeley	Bikel	Malt	Stanford
ccomp	13.03%	16.50%	17.51%	14.88%
xcomp	15.27%	15.57%	10.85%	13.55%
rcmod	9.27%	9.24%	12.88%	11.33%
advcl	1.26%	0.45%	2.20%	1.89%
csubj	0.78%	0.02%	0.35%	0.09%

Table 6.41 Bikel parser top 10 unmatched edge types (Functional evaluation)

Missed node	% of match mistakes
PP	20.18%
POBJ	16.55%
DOBJ	9.50%
NP	8.25%
NSUBJ	7.30%
XCOMP	6.04%
VP	5.71%
RCMOD	5.43%
CCOMP	5.08%
ACOMP	5.00%

Stellenbosch University https://scholar.sun.ac.za

6.3.4.2 Labelling mistakes

Constituency labelling mistakes consist largely of wrongly predicted nominal and clausal labels. Labels that are predominantly mispredicted are active and passive subjects, objects and open and closed clausal compliments. (See table 6.43 on page 172.) This also holds for Bikel parser (see table 6.44 on page 172). These problems are illustrated in Bikel candidate parses for sentences 530 and 775:

(109) Muskietnette is ook voorsien
Mosquito-nets PAST-be also provide
'Mosquito nets are also provided'

Figure 6.26: Sentence 530 – Gold standard, constituency

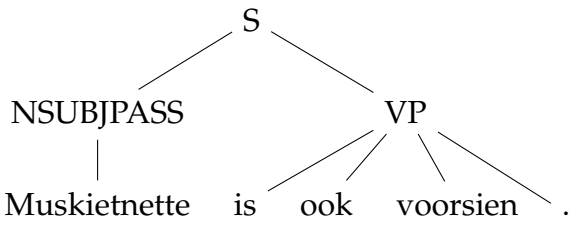
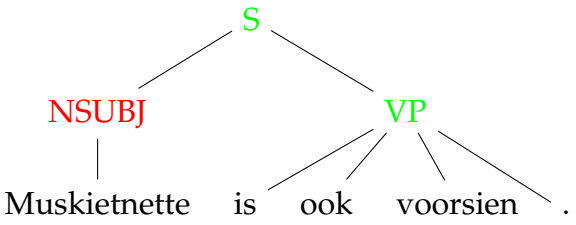


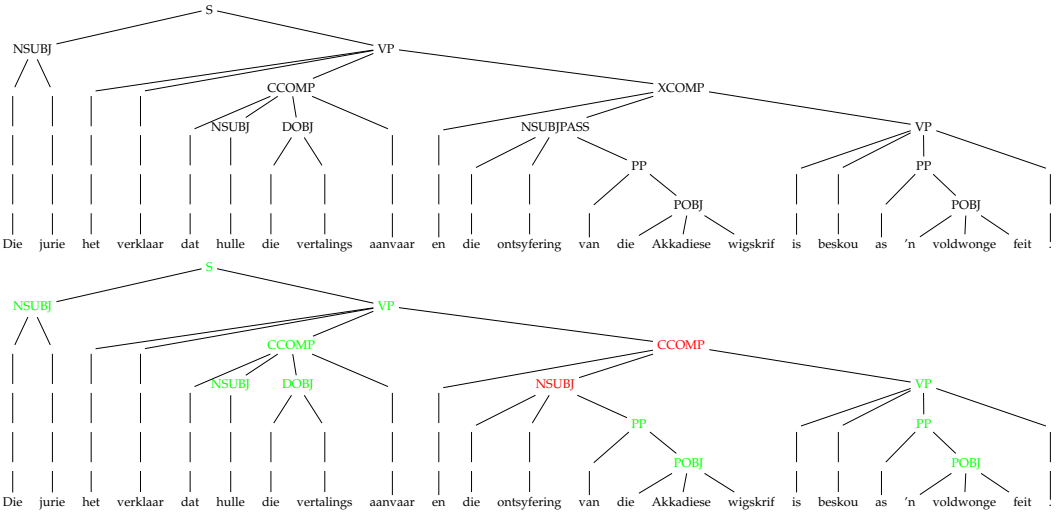
Figure 6.27: Sentence 530 – Bikel Functional candidate, constituency



(110) Die jurie het verklaar dat hulle die vertalings aanvaar en die
The jury has declare that they the translations accept and the
ontsyfering van die Akkadiese wigskrif is beskou as 'n
decyphering of the Akkadian cuneiform is regard as an
voldwonge feit
accepted fact

Stellenbosch University https://scholar.sun.ac.za

'The jury declared that they accept the translations and the decyphering of the Akkadian cuneiform has been regarded as fait accompli'



These mistakes are not very different from those made by their dependency counterparts.

6.3.5 Correlations

6.3.5.1 Sentence lengths

A functional label set has a negligible effect on the length/score correlation coefficients for constituency results as compared to Raw evaluation. Correlations remain negative. (See figures 6.28 on page 173 and 6.29 on page 173.)

Table 6.45 Sentence length / score correlation coefficient (Functional evaluation)

Parser	Coefficient
Berkeley	-0.18889
Bikel	-0.26581

6.3.5.2 Label frequency

Functional evaluation exhibits the same strong correlation between label frequency and label scores. (See figure 6.30.)

Stellenbosch University https://scholar.sun.ac.za

Table 6.42 Berkeley parser top 10 unmatched edge types (Functional evaluation)

Missed node	% of match mistakes
PP	20.04%
POBJ	17.32%
DOBJ	11.69%
NP	7.98%
NSUBJ	7.78%
VP	7.57%
ACOMP	5.48%
XCOMP	4.75%
RCMOD	4.59%
CCOMP	3.77%

Table 6.43 Berkeley parser top 10 label mistakes (Functional evaluation)

Correct	Guess	% of node mistakes
PATIENT	NSUBJ	12.00%
NSUBJ	PATIENT	8.88%
DOBJ	NSUBJ	6.19%
NSUBJ	DOBJ	6.17%
XCOMP	CCOMP	3.95%
NP	POBJ	3.10%
POBJ	NP	3.03%
CCOMP	XCOMP	2.89%
DOBJ	PP	2.64%
DOBJ	POBJ	2.48%

Table 6.44 Bikel parser top 10 label mistakes (Functional evaluation)

Correct	Guess	% of node mistakes
NSUBJ	DOBJ	13.46%
PATIENT	NSUBJ	11.02%
DOBJ	NSUBJ	8.20%
NSUBJ	PATIENT	5.53%
XCOMP	CCOMP	4.67%
NP	POBJ	4.09%
NP	DOBJ	4.01%
CCOMP	XCOMP	3.49%
POBJ	NP	2.73%
PATIENT	DOBJ	2.48%

Stellenbosch University https://scholar.sun.ac.za

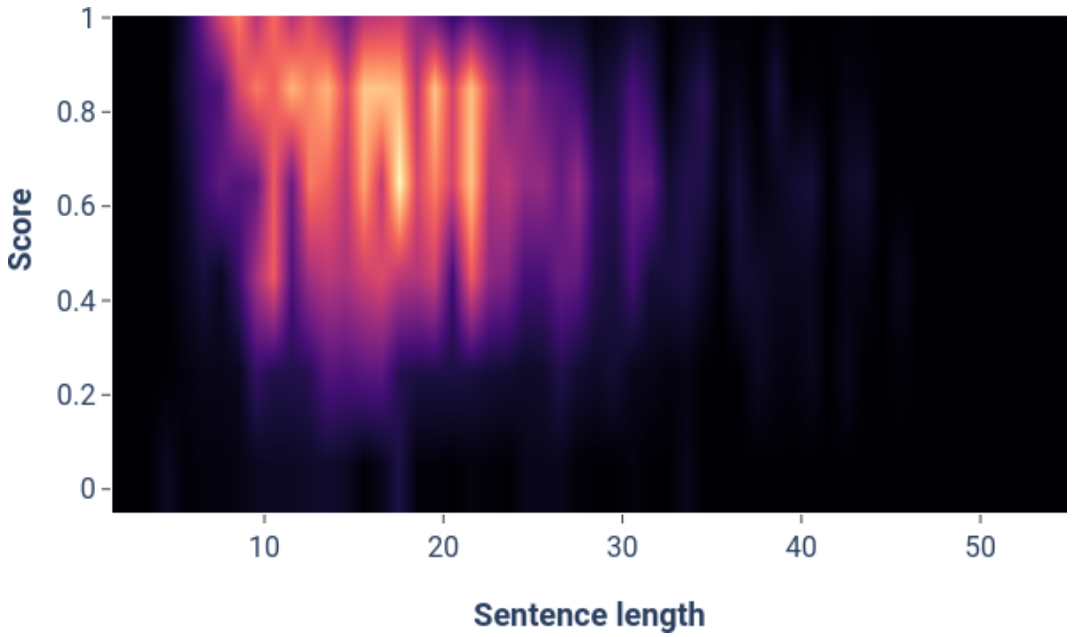


Figure 6.28: Sentence scores versus lengths (Berkeley – Functional)

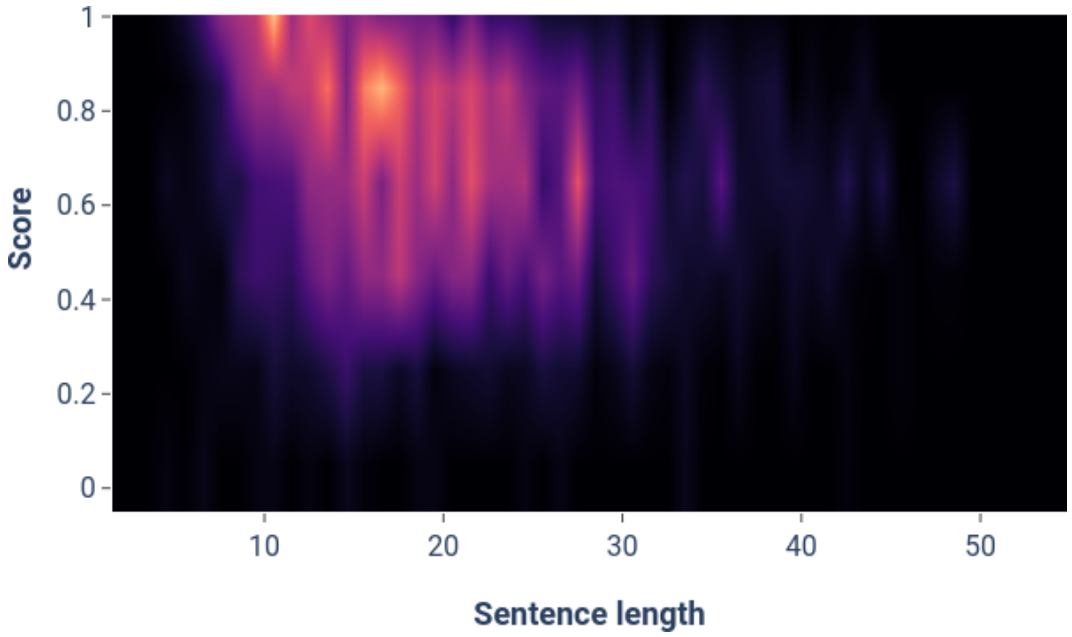


Figure 6.29: Sentence scores versus lengths (Bikel – Functional)

Stellenbosch University https://scholar.sun.ac.za

Table 6.46 Label frequency / score correlation coefficient (Functional variation)

Parser	Coefficient
Berkeley	0.89587
Bikel	0.90048

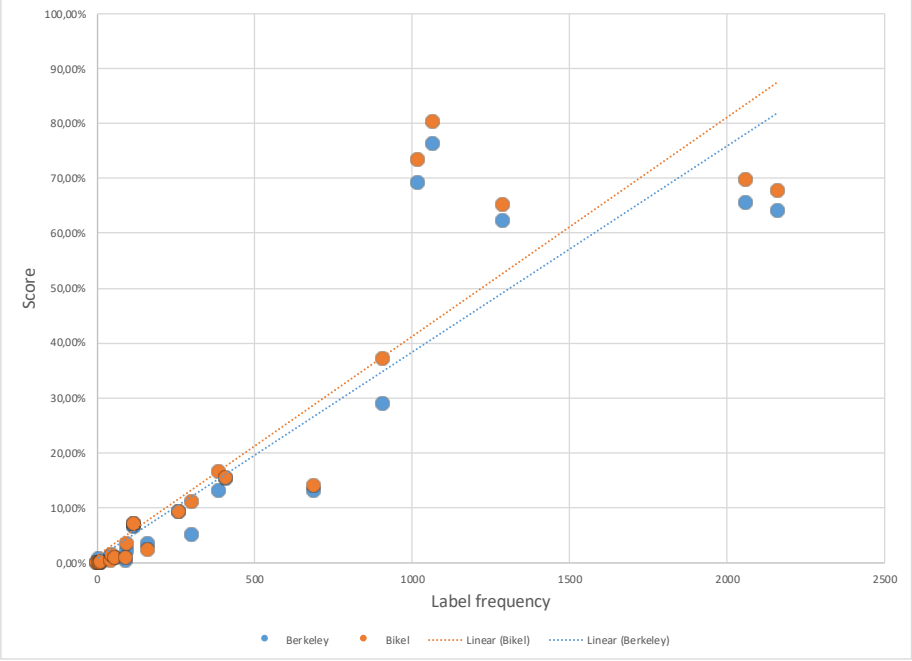


Figure 6.30: Label frequency impact on scores (Constituency, Functional)

6.3.6 Summary

Conversion of constituency labels to functional versions of themselves does not improve constituency parser results. No significant change in the types of labels that obtained low scores in Raw evaluation, such as clauses and indirect objects, is achieved. Correlations remain similar to that of Raw and Generative evaluations.

The slightly underwhelming results of Functional evaluation is not surprising. Just as the smaller Generative label set resulted in slightly better

Stellenbosch University https://scholar.sun.ac.za

performance for constituency parsers, the larger Functional label set results in fewer of each label type, which translates to slightly lower label scores for constituency parsers.

6.4 Categorical evaluation results

6.4.1 Parser scores

Conversion of Swartsbank to a Categorical label set (mapping the Generative constituency labels onto the dependency half of the treebank) doesn't notably increase or decrease the dependency scores in comparison to those generated during Raw evaluation. Only slight improvements are incurred (Malt +2.64%, Stanford +1.22%). Malt parser remains the highest scoring parser with an LAS of 86,07%.

Table 6.47 Parser scores – Categorical treebank

Parser	Score (Raw)	Score (Categorical)
Malt	83.43%	86.07%
Stanford	79.32%	80.54%

The Categorical variation is essentially a simplified version of the Dependency label set. To achieve this simplicity, nominal, adjectival and clausal labels are merged into single labels – *nomen*, *adj* and *clause*. The label *nomen* aligns with the Generative label *NP*. The label *clause* represents the labels *CP* and *TP*. (These two labels have to be merged into one label, because the dependency framework has only a single edge available with which to indicate a clausal dependency, as opposed to the CP-TP hierarchy which is built from two nodes.) The label *adj* replaces all instances of the dependency labels *amod* (attributive adjectives) and *acomp* (predicative adjectives).

Stellenbosch University https://scholar.sun.ac.za

6.4.2 Root scores

Categorical evaluation does not change dependency root scores. Dependency parsers remain better at predicting roots than constituency parsers. (See table 6.48 on page 180.)

6.4.3 Label scores

6.4.3.1 Nominal labels

Categorical simplification of the dependency label set results in increased nominal label scores. The label *nomen* consistently scores above 80% for both dependency parsers, with Malt dominating at 88.38%. (See table 6.49 on page 180.)

6.4.3.2 Adjectival labels

Adjectival scores are several percentage points higher for dependency parsers than those achieved in Raw and Generative evaluation. This improvement is due to the merging of multiple labels into a single one (*adj*), and therefore does not teach us anything interesting. Malt parser achieves the highest score (see table 6.50 on page 180).

6.4.3.3 Adpositional labels

No statistically significant change is found in prepositional scores. This is unsurprising, since the only structural change in the dependency label set is a very small number of postpositions (6) which are grouped with the existing set of prepositions into the label *adp*. Thus dependency parsers still do not outperform constituency parsers for adpositional labels. Malt parser achieves the highest score of the dependency parsers (see table 6.51 on page 180).

6.4.3.4 Clausal labels

When clausal labels in Swartsbank are merged into a single label, dependency scores for this label more than triple in comparison to those in Raw evaluation. Malt parser reaches the highest score at 42%. (See table 6.52 on page 180.) Here, again, grouping different types of clauses together under one label

Stellenbosch University https://scholar.sun.ac.za

results in a more frequently occurring label, which correlates with a higher score.

6.4.4 Mistakes

The Categorical label set has a notable effect on the composition of errors for dependency parsers. Similar to Generative evaluation, Categorical evaluation sees an increase in the proportion of unmatched nodes for dependency parsers, with approximately 9 out of 10 errors falling in this category.

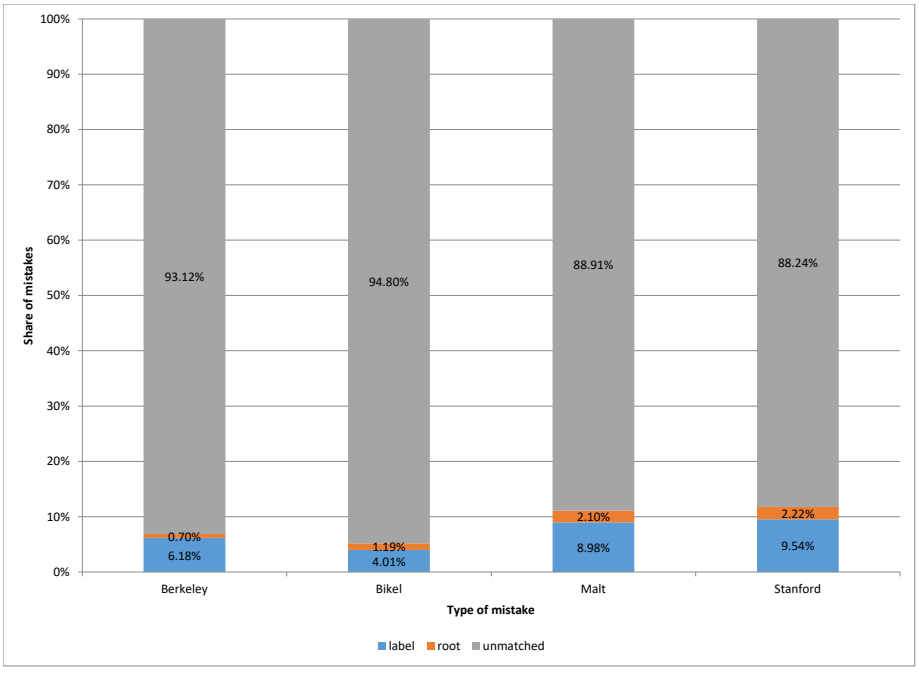


Figure 6.31: Distribution of mistakes in Categorical evaluation runs

6.4.4.1 Unmatched nodes and edges

Missed nominal edges are the most common mistake in Malt and Stanford parser results, followed by adpositional and clausal dependencies. (See tables 6.53 and 6.53 on page 180.) This patterns conforms to what is observed

Stellenbosch University https://scholar.sun.ac.za

in Raw and Generative evaluation runs, and can likely be explained by reference to label frequency.

For examples of these mistakes, see the candidate parses for sentence 749 below.

(111) Sy buitelandse beleid het veral op goeie betrekkinge met die
His foreign policy has especially on good relations with the
voormalige vyande Rusland en Sjina asook die
former enemies Russia and China as-well-as the
Arabies-Israeliese konflik gefokus
Arabic-Isreali conflict focused
'His foreign policy focused especially on good relations with former enemies Russia and China, as well as the Arabic-Isreali conflict'

Figure 6.32: Sentence 749 – Gold standard (dependency)

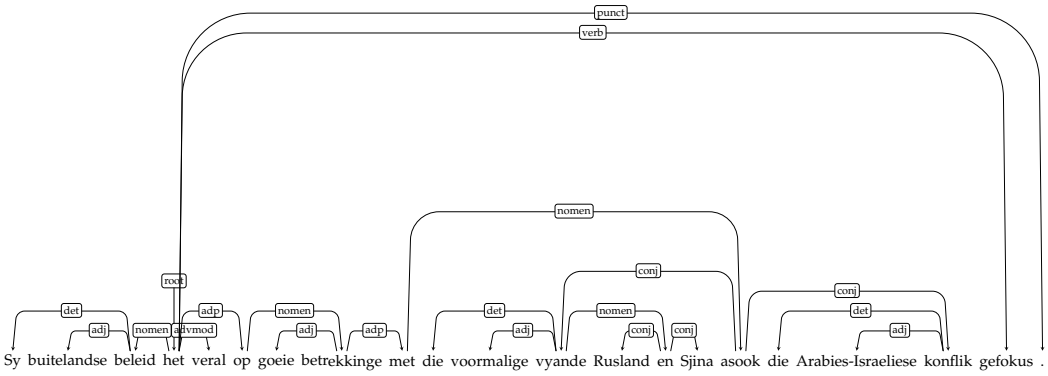
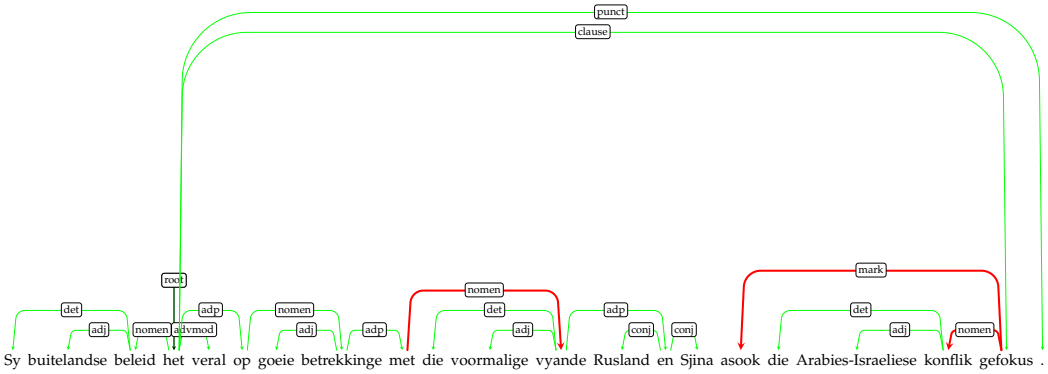
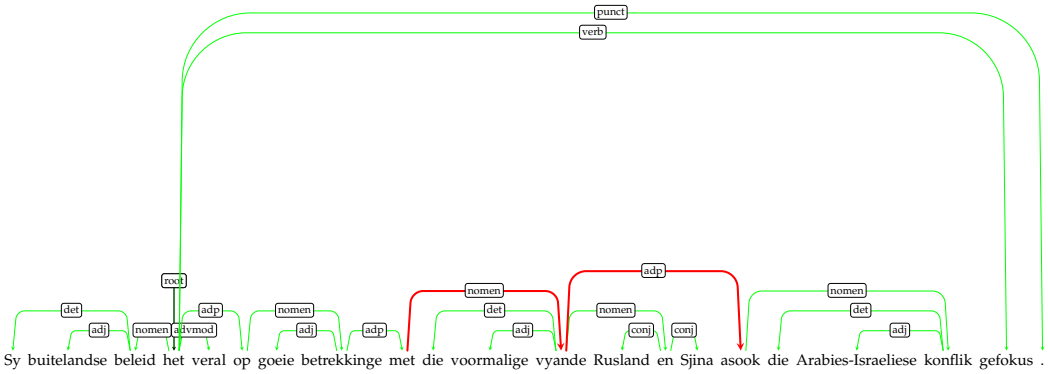


Figure 6.33: Sentence 749 – Malt candidate



Stellenbosch University https://scholar.sun.ac.za

Figure 6.34: Sentence 749 – Stanford candidate



6.4.4.2 Labelling mistakes

Not much useful (in the sense that we can adjust Swartsbank for better results) can be learned from observing labelling mistakes made during Categorical evaluation. We include them for the sake of completeness in tables 6.55 (page 181) and 6.56 (page 181).

6.4.5 Correlations

6.4.5.1 Sentence lengths

A negative correlation between sentence lengths and scores is observed in Categorical evaluation results:

Table 6.57 Sentence length / score correlation coefficient (Categorical evaluation

Parser	Coefficient
Malt	-0.22426
Stanford	-0.22905

This correlation is indicated visually in heatmaps 6.35 and 6.36.

6.4.5.2 Label frequency

As in prior evaluation runs, a high correlation between label occurrences and label frequencies is also observed in Categorical evaluation data:

Stellenbosch University https://scholar.sun.ac.za

Table 6.48 Root scores – Categorical evaluation

	Malt	Stanford
root	93.06%	90.81%

Table 6.49 Nominal scores – Categorical evaluation

	Malt	Stanford
nomen	88.38%	84.85%

Table 6.50 Adjectival scores – Categorical evaluation

	Malt	Stanford
adj	57.00%	54.93%

Table 6.51 Adpositional scores – Categorical evaluation

	Malt	Stanford
adp	64.20%	60.85%

Table 6.52 Clausal scores – Categorical evaluation

	Malt	Stanford
clause	42.60%	35.94%

Table 6.53 Malt parser top unmatched node types (Categorical evaluation)

Missed node	% of match mistakes
nomen	20.74%
adp	18.06%
clause	12.01%
advmod	9.00%
conj	6.31%
adj	3.30%
mark	3.23%
det	2.57%
num	2.38%

Stellenbosch University https://scholar.sun.ac.za

Table 6.54 Stanford parser top unmatched node types (Categorical evaluation)

Missed node	Count	% of match mistakes
nomen	19.91%	
adp	15.42%	
clause	11.62%	
advmod	8.38%	
conj	5.84%	
mark	4.21%	
particle	4.17%	
adj	3.14%	
verb	2.74%	

Table 6.55 Malt parser top 10 label mistakes (Categorical evaluation)

Correct	Guess	Count	% of node mistakes
adp	nomen	324	12.05%
clause	verb	200	7.44%
verb	clause	178	6.62%
det	adj	168	6.25%
nomen	num	157	5.84%
mark	nomen	120	4.46%
num	nomen	106	3.94%
adj	nomen	79	2.94%
nomen	clause	75	2.79%
particle	inf	72	2.68%

Table 6.56 Stanford parser top 10 label mistakes (Categorical evaluation)

Correct	Guess	Count	% of node mistakes
clause	verb	275	6.98%
verb	clause	231	5.86%
adp	nomen	229	5.81%
advmod	mark	182	4.62%
conj	nomen	169	4.29%
det	adj	164	4.16%
mark	nomen	164	4.16%
num	nomen	162	4.11%
adj	det	131	3.32%
nomen	num	117	2.97%

Stellenbosch University https://scholar.sun.ac.za

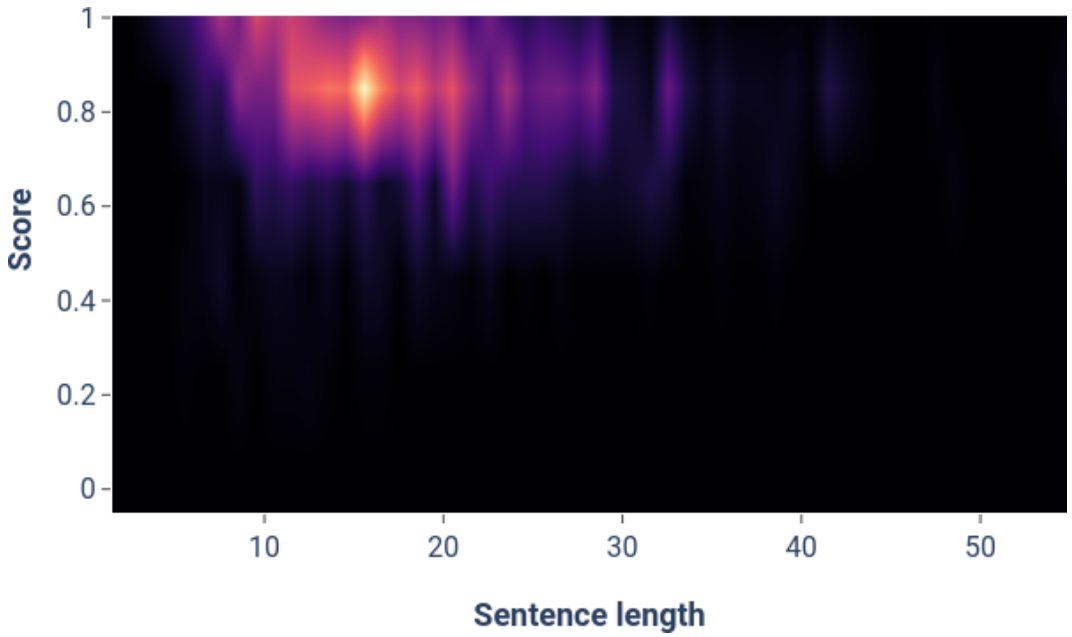


Figure 6.35: (Sentence scores versus lengths (Malt – Categorical)

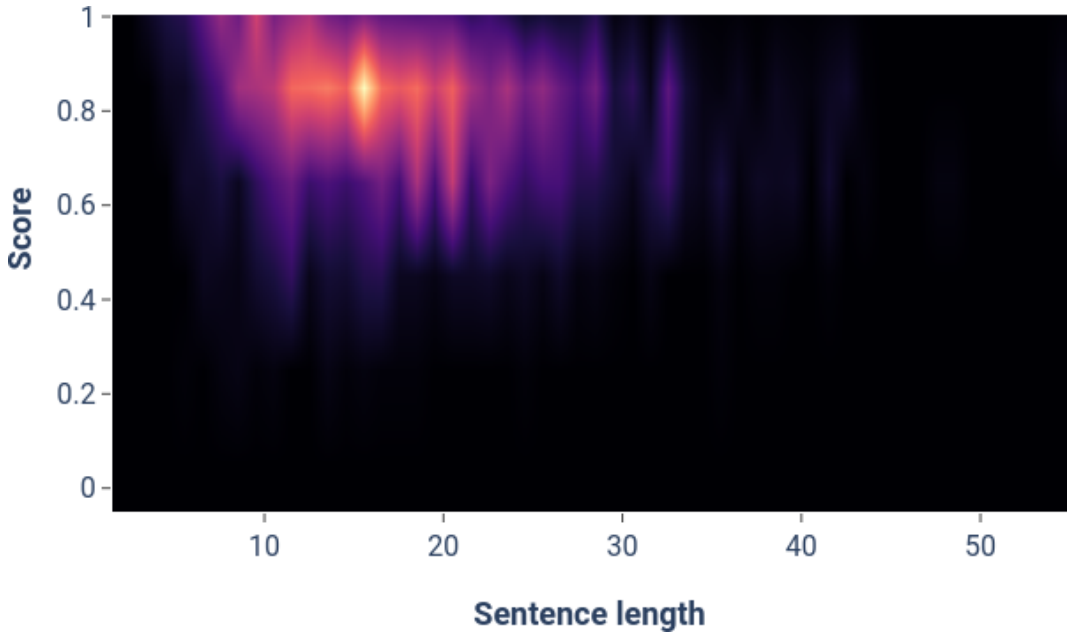


Figure 6.36: Sentence scores versus lengths (Stanford – Categorical

Stellenbosch University https://scholar.sun.ac.za

Table 6.58 Label frequency / score correlation coefficient (Categorical evaluation)

Parser	Coefficient
Malt	0.68035
Stanford	0.68290

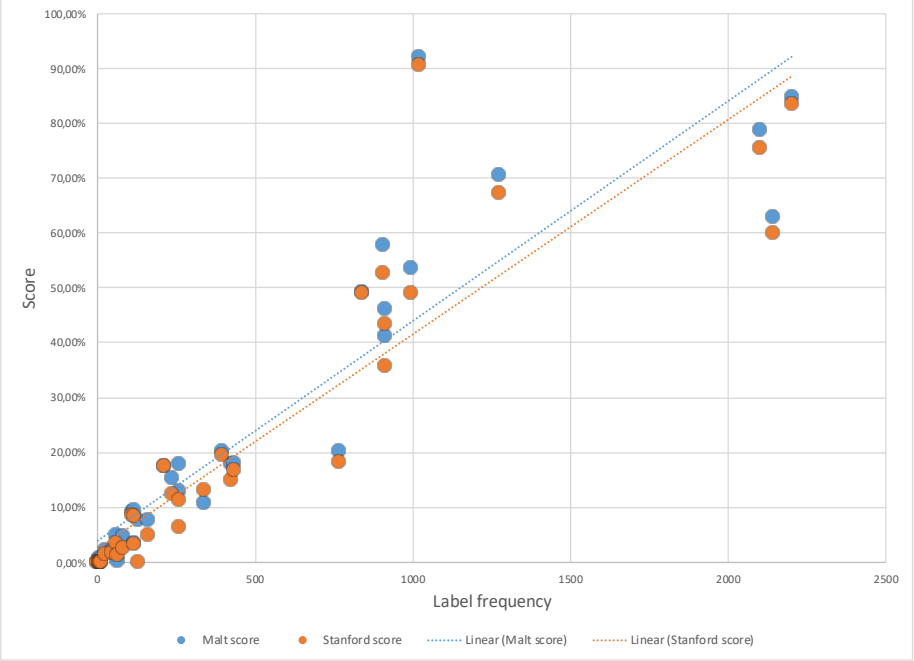


Figure 6.37: Label frequency impact on scores (Dependency, Categorical)

6.5 Summary

We make the following observations from the data presented above.

Observation 1: dependency parsers generally perform better than constituency parsers when predicting the syntactic constructions found in Swartsbank, regardless of the size of the label set employed. (The exception to this observation is the prediction of prepositional constructions and clausal

Stellenbosch University https://scholar.sun.ac.za

nodes in the Generative variation, which constituency parsers do slightly better).

Observation 2: The frequency of a label in Swartsbank correlates strongly with each parser's ability to successfully predict it. This can be a consequence of a dataset that is too small, or a side-effect of the way parsers learn and make predictions. To test this, we repeated the same experiments ran on Swartsbank on much larger, publicly available treebanks for other languages, using Maltparser. We observed the same pattern as during parser evaluation with Swartsbank: less frequently occurring labels receive lower scores than more frequent ones, and label frequencies and label scores strongly correlate with one another (see table 6.5 below). This leads us to believe that our own results cannot solely be attributed to data sparsity. Our suspicion is that this pattern persists as a side-effect of the statistical nature of parsers – given a dataset with imbalanced classes (such as a treebank for a human language), it is reasonable to assume that a classifier trained on this dataset will predict more of the over-represented classes, and less of the under-represented classes.[3]

Table 6.59 Frequency / Score correlation coefficients for larger treebanks

Language	Data source	Sentences	Correlation coefficient
Russian	SynTagRus	61 888	0.90342
German	German UD Treebank	14 118	0.92884
English	English Web Treebank	12 543	0.68245
Dutch	Alpino	12 269	0.85463
Swedish	Talbanken	5 330	0.94194

Observation 3: Predictive errors made by parsers share a large number of similarities. All parsers struggle with the syntactic ambiguity found in complex prepositional phrases. Mistaken candidate parses are often syntactically correct even when they do not match the expected gold parse. (See for example the Bikel parse for sentence 749 on page 143.) It is not

[3]This can be explained in terms of Bayes' Theorem

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)}$$

in which the probability of a label predicted as a A given that it has certain features F is directly proportional to $P(A)$ – the probability (relative frequency) of A in the treebank.

Stellenbosch University https://scholar.sun.ac.za

immediately clear how this problem should be solved (since prepositional ambiguity trips up even humans easily) but we posit that it could be addressed in part by including additional semantic information in Swartsbank.

Different types of clausal and nominal labels are also confused with one other, especially in the constituency framework. This confusion could be indicative of a label set that is too complicated, a dataset that is not big enough or simply of the morphological impoverishment of Afrikaans, which implies a lack of case marking and verbal inflection which can contribute to parser predictions. Reaching a verdict on why this is the case is hard without additional data.

Observation 4: A label set describing Afrikaans from a functional perspective performs slightly better during parser evaluation than a label set describing Afrikaans from a categorical perspective. We do not believe this is enough evidence to consider one perspective superior to the other (such a view would suffer the same bias we criticize the leaf ancestor metric for in section A.2). Nonetheless, it is useful to know this, and might help inform the future expansion of Swartsbank.

Observation 5: Maltparser provides superior results when evaluated on Swartsbank, both in terms of accuracy and speed. We did not anticipate this result, but given Maltparser's steeper learning curve and tendency to perform better on smaller datasets (refer to section 5.3.2.3), it is not surprising.

Stellenbosch University https://scholar.sun.ac.za

Chapter 7

Conclusion

The data presented in Chapter 6 invalidates our null hypothesis (**H0**)[1] and validates our research hypothesis (**H1**)[2]. It does so by showing that grammatical frameworks do have an impact on the efficacy of representing and computationally parsing written Afrikaans sentences, and that dependency grammar seems more efficient at these tasks than constituency grammar. The answer to our research question:

> What is an optimal grammatical framework for the computational representation and sparsing of Afrikaans sentences?

is therefore:

> A dependency grammar with functionally oriented syntactic labels

We arrived at this conclusion using the theoretical and empirical knowledge gained in this study.

In Chapter 4, we found that dependency grammar handles Afrikaans' non-projective constructions more elegantly than constituency grammar. We also demonstrated how dependency grammar allows for richer analyses of composite verbal structures found in Afrikaans, such as complex initials. Based on this, we conclude a priori that a dependency framework is preferable when modelling the syntactic structure of written Afrikaans.

[1] "Types of grammatical frameworks do not impact the efficacy of representing and parsing Afrikaans sentences."

[2] "Written Afrikaans sentences are most accurately represented and parsed by dependency parsers."

Stellenbosch University https://scholar.sun.ac.za

In Chapter 6 we analysed the data from parser evaluation on four versions of Swartsbank. According to this data, dependency parsers consistently fare better when predicting the syntactic structures of written Afrikaans than constituency parsers. This outcome seems to hold true no matter the complexity of the label set employed for each framework, as indicated in the combined set of parser scores in the table below:

Table 7.1 Parser scores – all variations

Parser	Raw	Generative	Functional	Categorical
Berkeley	69.76%	71.69%	64.86 %	—
Bikel	73.46%	72.63%	67.90 %	—
Malt	**83.43%**	**84.65%**	—	**86.07%**
Stanford	79.32%	80.23%	—	80.54%

We arrived at these results by describing the formal properties of constituency and dependency grammars, after which we described the core syntactic components in written Afrikaans from a constituency and dependency perspective *[Objective 1]*.

In doing so, we have contributed the first large scale dependency representation of written Afrikaans to the body of academic work on the language. This representation informed the construction of label sets and annotation guidelines for both types of formalisms, which allowed the first theoretically grounded Afrikaans treebank to be constructed out of sentences from the Leipzig Corpora collection *[Objective 2]*.

The resultant treebank (Swartsbank) was used to evaluate four state-of-the-art syntactic parsers (Berkeley, Bikel, Malt and Stanford) *[Objective 3]*.

The results of these experiments, combined with the theoretical knowledge gained through completing the first objective, suggest that written Afrikaans is better suited towards dependency parsing *[Objective 4]*.

We present these results with the acknowledgement of a number of limitations. Firstly, we do not know to what extent our dataset qualifies as sparse. Compared to the likes of the Penn Treebank (3 million syntactically annotated tokens), TüBa-D/Z (1.7 million) the Prague Dependency Treebank (1.5 million) and others, Swartsbank is small. It is possible that a larger

Stellenbosch University https://scholar.sun.ac.za

treebank might provide different results. This presents future research opportunities.

Secondly, we have considered the selection of an annotation framework for Afrikaans to be a zero sum game. Based on the research by Hall et al. [44] referred to in Chapter 2, a hybrid framework in which dependency and constituency information are combined in a single graph could also have been investigated. (It is not apparent to us, however, how such a hybrid would deal with non-projective constructions.)

Thirdly, we have not measured the effect of any form of post-processing (heuristically driven changes to labels after parsing) on the output of constituency or dependency parsers. We suspect this could improve the low scores for some infrequently occurring labels which follow a regular pattern such as clauses and passive subjects. It might also provide a mechanism to deal with discontinuities in constituency grammar. This, too is an avenue for future research.

Fourthly, our dataset has been created by a single annotator. Great care has been taken during this process to adhere to the annotation guidelines set out in Chapter 4 and an extended set of integrity tests have been added to "TreeGUI" and "Treebank" to ensure that as many errors as possible are eliminated. Even so, it is entirely possible that some mistakes might lurk in Swartsbank. Furthermore, human language is by definition ambiguous, and additional annotators might have agreed on a slightly different (and equally valid) set of constituency and dependency graphs for the same set of sentences.

7.1 Suggestions for future research

Corpus expansion — We recommend that Swartsbank be expanded to 10 times its current size, containing 20 000 syntactically annotated sentences. This will put it on par with larger public datasets such as the German UD Treebank (∼20 000 sentences) and The Alpino UD Treebank (∼12 000 sentences).

Minimum viable treebank size — It is not clear what treebank size is necessary to achieve state of the art parser results for Afrikaans. Future research with an expanded version of Swartsbank could determine this.

Stellenbosch University https://scholar.sun.ac.za

Corpus enrichment — We hypothesise that nominal scores could be improved by adding more information about verbs (valency, aspect, voice et cetera) and using this as extra parser features to help overcome Afrikaans' morphological impoverishment. Adjectival scores could likely also be improved by extracting inflectional information as a preprocessing step and providing this information to parsers. (Predicative adjectives in Afrikaans largely follows the same inflectional pattern.)

Label set optimisation — The smaller dependency label set used for Categorical evaluation resulted in improved dependency scores. This hints at the value of refining the labels in the existing dataset to improve nominal, prepositional and clausal accuracy.

Parser parameter tweaking — Parsers were used with default settings during this study, since the degree to which each can be tweaked vary. Further research could investigate the effect of tweaking the multitude of parameters available in each parser to see whether higher scores are achievable.

Maltparser and other languages — An unanticipated finding from our experiments is that Malt parser performs better than other parsers, and that this is likely because of the relatively small size of Swartsbank and the parser's relatively steep learning curve. Research endeavours for other resource scarce languages (especially in the South African context) could benefit from knowing this. We recommend that the broader NLP community in South Africa investigate this potential benefit in more detail.

Stellenbosch University https://scholar.sun.ac.za

Appendix A

Critiques

This chapter contains two critiques that contribute to the central thesis of this study, but are not suited to the flow of the main document.

A.1 Critique of Afribooms

Unless stated otherwise, this critique is based on the publicly available version of Afribooms which has been released in Universal Dependency format, utilizing a subset of the Stanford tagset.[1]

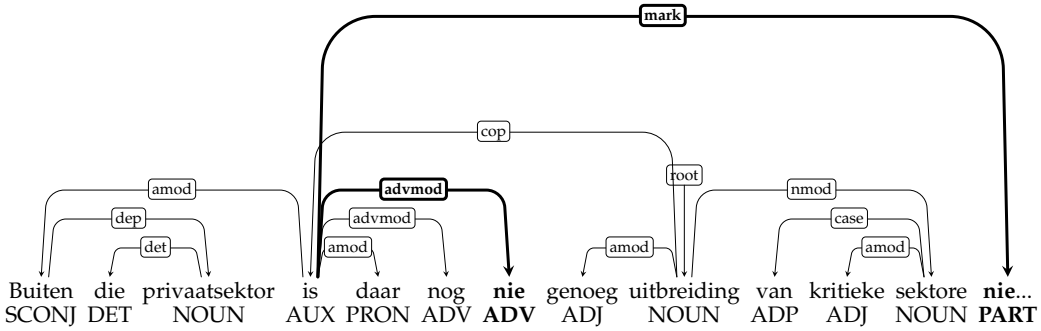
Afribooms, a treebank of 1740 Afrikaans sentences sourced from South African government texts, signifies the first attempt at building an Afrikaans treebank [5]. Afribooms was built by converting Afrikaans sentences to Dutch, parsing these converted sentences with a Dutch dependency parser, and then converting the resultant sentences back into Afrikaans. As a final step, the sentences were presented to a human annotator for inspection and correction. During the course of this study, we manually inspected Afribooms to see if we could use it to bootstrap Swartsbank. However, we observed a number of problems with the treebank which led us to discard this idea. The main problems we encountered were as follows:

1. Afribooms' annotation guidelines prohibits non-projective constructions [5][678], thereby violating the dependency structure of discontinuous constructions found in Afrikaans, such as double negatives

[1] *Available at* https://github.com/UniversalDependencies/UD_Afrikaans-AfriBooms

Stellenbosch University https://scholar.sun.ac.za

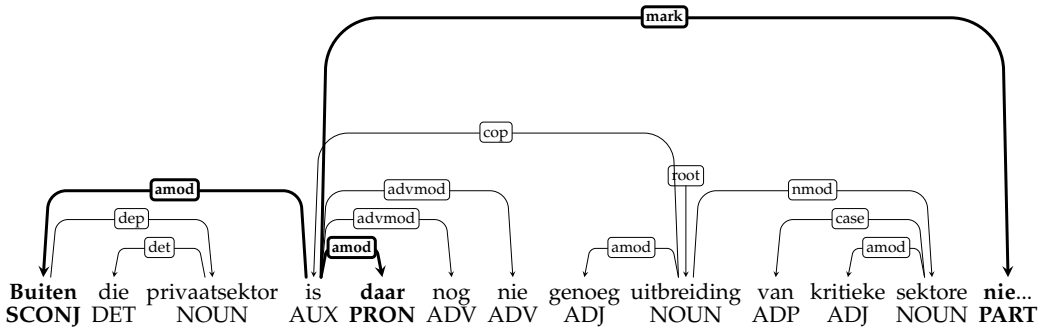
and discontinuous relative clauses. An example of this can be seen in Afribooms sentence "test-s250" in which a negation particle is governed by the auxiliary verb *is*, rather than its negator *nie*:



Not allowing non-projectivity in an Afrikaans treebank has a serious impact on its ability to accurately represent the structure of the language. To illustrate the severity of this impact, consider that just over 19% of the sentences in Swartsbank contains non-projective constructions. This design decision alone means 1 in every 5 sentences would not be accurately represented.

2. Afribooms contains syntactically incongruous combinations of POS tags and edge labels. Consider again sentence "test-s250", in which:

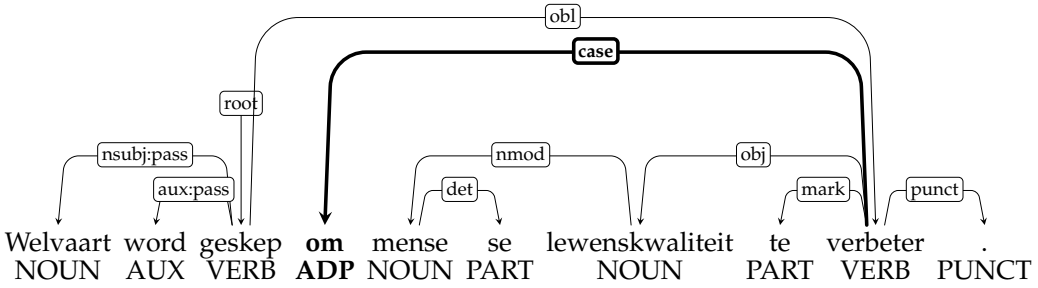
   - the preposition *buiten* is mistakenly categorized as a subordinating conjunction (*SCONJ*) yet governed in an adjectival dependency (*amod*)
   - the expletive *daar* is mistakenly categorized as a pronoun (*PRON*) yet governed in an adjectival (*amod*) dependency
   - the negation particle *nie* is (correctly) indicated as *PART*, but governed by a clausal marker dependency (*mark*)



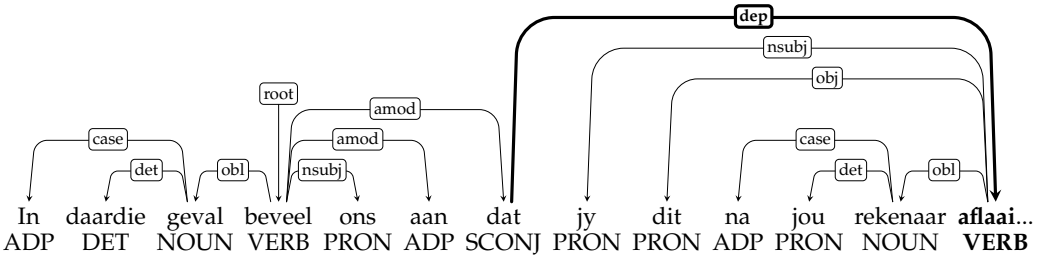
Stellenbosch University https://scholar.sun.ac.za

The mistaken POS tags are especially problematic, as they are bound to add faults into any model deduced by a parser from Afribooms.

3. Afribooms does not seem to distinguish between "om" as adposition and "om" as clausal marker, accounting only for the former. This is illustrated in Afribooms sentence "test-s256":

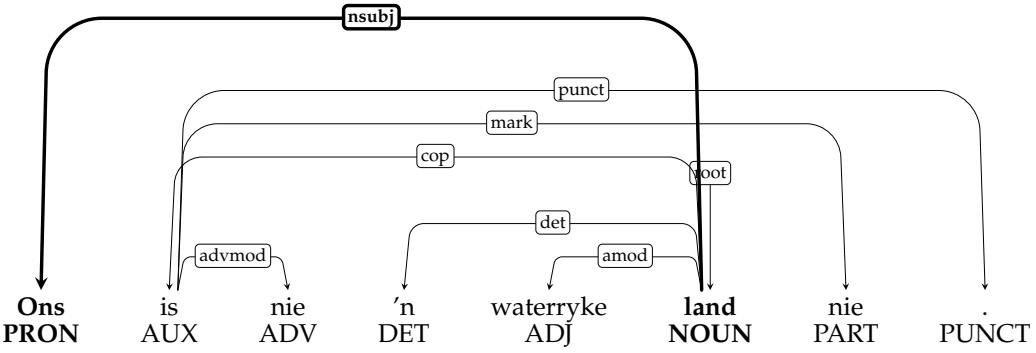


4. Afribooms contains 1455 instances of edges simply labeled as *dep* ("dependency"), thereby losing the syntactic information that would have been present if annotators had chosen a proper syntactic label. From what we can observe, these annotations do not seem particularly ambiguous or hard to decide on. Consider sentence "train-s42" in which a verb is governed by a *dep* (sentence truncated for illustrative purposes):

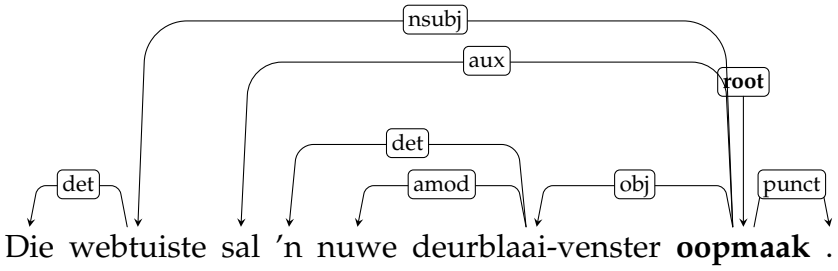


5. Afribooms seems to be at odds with its own annotation guidelines. While non-projectivity is not allowed in Afribooms, it does contain non-projective structures. These structures seem to rarely represent actual non-projective dependencies that occur in Afrikaans. As an example, consider Afribooms sentence "test-s133" in which a negation particle is not connected to its negator (presumably because of the projectivity constraint), yet a pronoun is connected to a noun via a non-projective dependency:

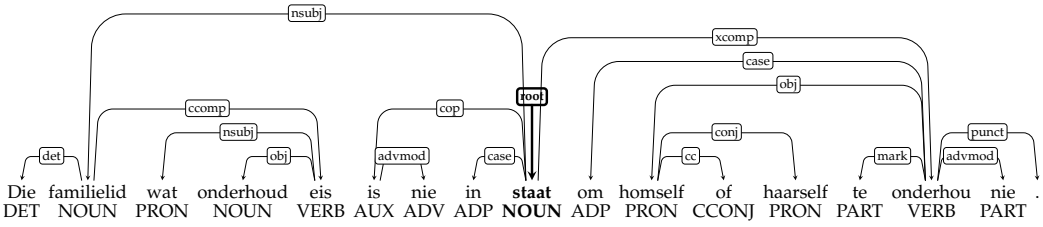
Stellenbosch University https://scholar.sun.ac.za



6. Afribooms does not treat finite auxiliary verbs as graph roots. Instead, in cases where finite auxiliary verbs are present in a sentence, infinite lexical verbs are used for this purpose. An example of this is found in sentence "train-s28":

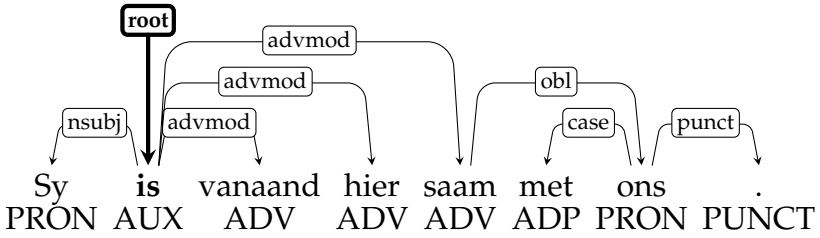


7. Afribooms does not treat copular verbs as graph roots, opting rather to assign this role to "content words" such as nouns and adjectives. Consider for example sentence "train-s79", in which a prepositional object (*staat*) is marked as the root:



This anomaly, presumably, is the result of not merely basing labels in Afribooms on Stanford Typed Dependencies [5][678], but also following their linguistically unsound practice of regarding "content words" hierarchically more important than others. Yet even this principle is not properly followed by Afribooms, as illustrated in sentence "train-s161", in which a copular verb contradictorily **is** the graph root:

Stellenbosch University https://scholar.sun.ac.za



In total, we count 43 sentences in Afribooms that breaks this rule.

8. Afribooms does not contain any labels for relative clauses.

The low quality of the Afribooms edge labels and part of speech tags is likely a side effect of how the treebank was constructed. From a manual inspection of the dependency graphs contained in Afribooms, it seems that its designers did not try to theoretically ground their understanding of what they were building, relying rather on gut feel. This seems to have led to linguistically dubious annotations and therefore – while we applaud its creators for taking the first step towards building an Afrikaans treebank – we do not consider Afribooms useful for sustained future use.

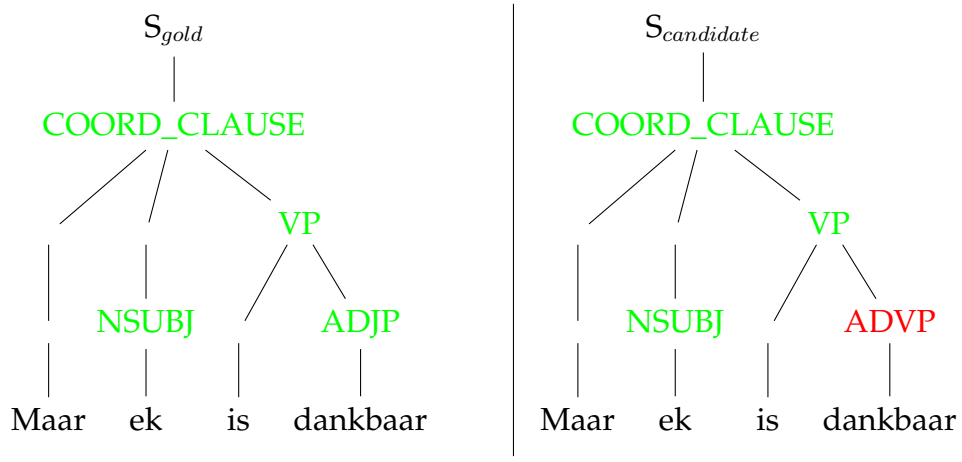
Stellenbosch University https://scholar.sun.ac.za

A.2 Critique of the Leaf Ancestor metric

The Leaf Ancestor Metric (LA) has been developed as an alternative to the Parseval metric. Parseval (in use in NLP circles since the early 1990s) measures the correctness of a candidate parse by comparing matching constituents in its corresponding gold parse. To be considered correctly parsed, a constituent has to exhibit the following characteristics:

- The start and end positions of the candidate constituent are the same as those of the gold constituent.
- The syntactic label of the candidate constituent is the same as that of the gold constituent.

From these constituents, an F1-score can be calculated for the sentence as a whole, as well as for each node type. The example gold and candidate parses of sentence 87 in the Raw version of Swartsbank[2] illustrate this visually, with red indicating an incorrectly parsed constituent:



In the example above the parser has correctly identified the boundaries of 4 nodes (Recall = 4/4 = 1.0) and correctly predicted the labels of 3 of the 4 candidate constituents (precision = 3/4 = 0.75). This yields an F1 score of 85.71%[3]

The Leaf Ancestor metric has been promoted in NLP circles by Babarczy and Sampson as an alternative to Parseval [90] since the early 2000s

[2] *Maar ek is dankbaar* ("But I am thankful")

[3] $F1 = 2 \times \frac{p \times r}{p+r} = 2 \times \frac{0.75 \times 1}{0.75+1} \approx 0.8571$

Stellenbosch University https://scholar.sun.ac.za

on the basis that Parseval is too coarse a metric to account for partially correctly predicted constituents. Although Babarczy and Sampson do not provide a formal definition of what a partially correct constituent entails, the examples they provide are of candidate constituents that have almost (but not precisely) correctly predicted start and end nodes and almost (but not precisely) correctly predicted node labels. They describe this as follows (regarding boundary nodes):

> If, for instance, in the gold standard, words 5 to 14 are identified as a noun phrase, then a candidate parse which identifies a noun phrase beginning at word 5 but ending at word 13, or word 15, should in our view be given substantial though not full credit; under Parseval it is given no credit. [91][219]

and as follows (regarding node labels):

> ...partial credit is given for mistaking, say, a noun phrase for an N-bar, which is surely a lesser error than mistaking it for a subordinate clause. [91][225]

LA therefore relies on fuzzy matching of candidate and gold nodes. It achieves this with a modified version of the Levenshtein distance algorithm[4] that compares serialized versions of candidate and gold parses, but weighs partially correct labels more heavily than partially correct constituent boundaries. This is calculated as follows: $1 - \frac{Lv(c,g)}{len(c)+len(g)}$[5]. In this way, slightly incorrect bracketing and labels still count towards the overall score of a candidate parse, rather than being marked as incorrect. Babarczy and Sampson find this more desirable as it allows for a metric that more closely mirrors human intuition [91][220]. We disagree with them on the basis of two observations.

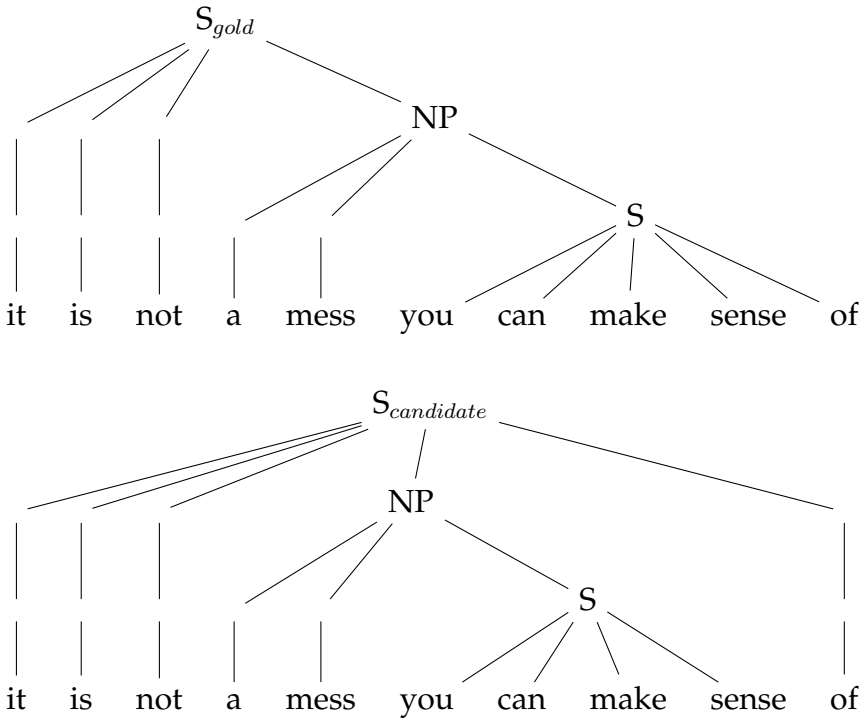
[4] An algorithm to calculate the similarity of two strings based on the number of character inserts and deletes that has to be performed to transform the first string into the second string

[5] Lv = Levenshtein distance, len = length, c = serialized candidate tree and g = serialized gold tree.

Stellenbosch University https://scholar.sun.ac.za

A.2.0.1 Observation 1 - Confirmation bias

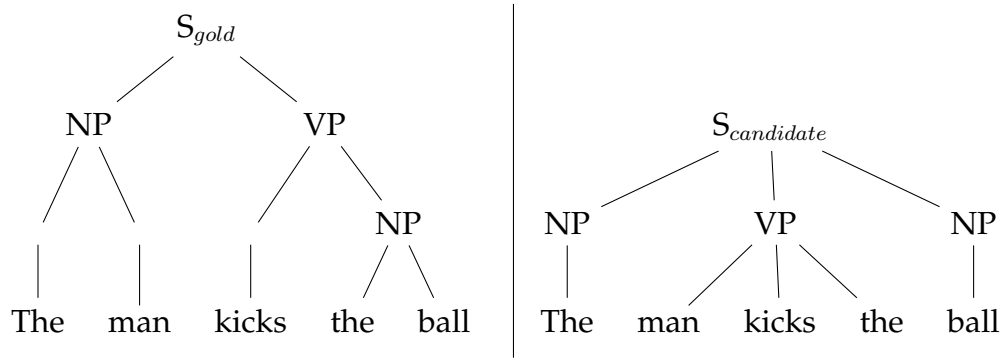
The argument in favour of fuzzy constituent boundaries allows demonstrably nonsensical parses to contribute higher scores to a parse tree. Consider the following gold and candidate trees presented by Sampson to illustrate the advantage of using LA over Parseval [90][7]



According to Babarczy and Sampson, the Parseval score for the candidate parse is approximately 0.33%, which to them seems excessively low for a parse in which only one leaf node boundary is misplaced. LA, on the other hand, scores the candidate parse with an LA score of 95,2% because – barring the dangling participle – the constituents in the candidate parse seem largely accurate to human intuition.

The argument is easily refuted with a counter-example:

Stellenbosch University https://scholar.sun.ac.za



The candidate parse in this example exhibits the same "almost-but-not-quite" characteristics of Babarczy and Sampson's example. The boundary of the first NP is a mere one word out from the gold standard, the VP two words and the second NP, again, only one word. Despite the constituents being semantically and syntactically nonsensical, LA's fuzziness dictates that the parser being evaluated should be rewarded for each of its predictions.

In presenting only examples favoured by the fuzzy approach and not considering counter-examples such as the one above, Babarczy and Sampson make themselves guilty of confirmation bias.

A.2.0.2 Observation 2 - Anthropomorphic bias

Just as partial constituent boundaries in candidate parses count towards a higher LA score, so do partially correct constituent labels. The labels that fit the "almost-but-not-quite" approach are those that are not the same in the candidate parse as in the gold parse, but still feel intuitively correct to a human because it looks similar. Babarczy and Sampson explains:

> The present experiment sets the cost of replacing a symbol by an unrelated symbol at 2, but the cost of a replacement where both symbols share the same first character at 0.5; thus partial credit is given for mistaking, say, a noun phrase for an N-bar, which is surely a lesser error than mistaking it for a subordinate clause. [90][6]

This claim reveals an incorrect understanding of probabilistic and deterministic parsers, and can be falsified using Babarczy and Sampson's own example: [91][221]

Stellenbosch University https://scholar.sun.ac.za

(G) [S [N1 two [N1 tax revision] bills] were passed]
(C) [S [NP two tax revision bills] were passed]

In this example, Babarczy and Sampson maintains, the mislabelling of *N1* in the gold parse as *NP* in the candidate parse is a lesser error than labelling it something less intuitive (say, *VP* or *XBAR*) – because the string still starts with an "N". While this approach might seem reasonable, it mistakenly relies on the characteristics of the specific label set that happens to be employed. To illustrate the problem, consider a slightly modified tagset that makes a more granular (yet grammatically tenable) distinction between subjects and objects, which could yield this version of Sampson's example:

(G) [S [SUBJ two [NP tax revision] bills] were passed]

(C) [S [DOBJ two tax revision bills] were passed]

This modification changes the string similarity scores, which changes the candidate parse's LA score, rendering the metric inconsistent and limiting it to use on label sets where the nomenclature happens to be that assumed in Babarczy and Sampson's original example. Worse, it relies on a seemingly flawed understanding of how most sentence parsers extract features and build grammars during their training phases. Consider that we could change our tag set even further, replacing our linguistically motivated labels with entirely random substitutes, resulting in the following gold tree:

(G) [R [X42 two [X42 tax revision] bills] were passed]

As long as these substitutions are made consistently to all graphs in a treebank[6], it would make no difference to either a probabilistic or a deterministic parser, since the resultant tagset would still have the same size and distribution of labels. The features used as training input would yield the same consistent output. Parseval scores would remain consistent; LA scores would fluctuate.

A constituency parser is not conscious. It does not "know" or "understand" that one label seems more similar to a human than another label. It simply

[6]That is to say, all labels of type *S* are renamed to *R*, all labels of type *N1* are renamed to *X42*

Stellenbosch University https://scholar.sun.ac.za

maps a model derived from a set of training data onto input data. The argument that LA is useful or better than Parseval because it better models human intuition betrays a seeming anthropomorphic bias on the part of Babarczy and Sampson.

Based on **Observation 1** and **Observation 2** we choose not to employ LA to evaluate the output of constituency based parsers used in this study's experiments, on the grounds that the metric yields nonsensical and inconsistent results.

On the matter of whether LA has any value at all, we will let Babarczy and Sampson themselves have the final word:

> If two alternative methods for measuring the same property yield uncorrelated results, one might have supposed that at least one of the methods could never be taken seriously at all. [90][7]

Stellenbosch University https://scholar.sun.ac.za

Stellenbosch University https://scholar.sun.ac.za

Appendix B

Appendices

B.1 Pilon Afrikaans POS Tagset

Voorbeeld	Waarde	Etiket	Intermediêre etiket
hierdie	aanwysend	PA	P000000100000
niemand	onbepaald	PO	P000000200000
myne	eerste/enkelvoud/besitlik	PEEB	P101000300000
joune	tweede/enkelvoud/besitlik	PTEB	P201000300000
syne	derde/manlik/enkelvoud/besitlik	PDHEB	P311000300000
hare	derde/vroulik/enkelvoud/besitlik	PDVEB	P321000300000
dit is ***ons*** *boek*	eerste/meervoud/besitlik	PEMB	P102000300000
dit is ***julle*** *boek*	tweede/meervoud/besitlik	PTMB	P202000300000
dit is ***hulle*** *boek*	derde/meervoud/besitlik	PDMB	P302000300000
wie	vraend	PV	P000000400000
ek sny ***myself***	eerste/enkelvoud/wederkerend	PEEW	P101000500000
jy sny ***jouself***	tweede/enkelvoud/wederkerend	PTEW	P201000500000
hy sny ***homself***	derde/manlik/enkelvoud/wederkerend	PDHEW	P311000500000
sy sny ***haarself***	derde/vroulik/enkelvoud/wederkerend	PDVEW	P321000500000
dit hou ***sigself*** *besig met...*	derde/onsydig/enkelvoud/wederkerend	PDOEW	P331000500000
ons sny ***onsself***	eerste/meervoud/wederkerend	PEMW	P102000500000
julle sny ***julleself***	tweede/meervoud/wederkerend	PTMW	P202000500000
hulle sny ***hulleself***	derde/meervoud/wederkerend	PDMW	P302000500000
ek *slaap*	eerste/enkelvoud/ongemarkeerd /persoonlik	PEENP	P101008600000
jy *slaap*	tweede/enkelvoud/ongemarkeerd /persoonlik	PTENP	P201008600000
hy *slaap*	derde/manlik/enkelvoud/ongemarkeerd /persoonlik	PDHENP	P311008600000
sy slaap	derde/vroulik/enkelvoud/ongemarkeerd /persoonlik	PDVENP	P321008600000
dit *slaap*	derde/onsydig/enkelvoud/ongemarkeerd /persoonlik	PDOENP	P331008600000
ons *slaap*	eerste/meervoud/persoonlik	PEMP	P102000600000
julle *slaap*	tweede/meervoud /persoonlik	PTMP	P202000600000
hulle *slaap*	derde/meervoud/persoonlik	PDMP	P302000600000
jy sien ***my***	eerste/enkelvoud/gemarkeerd/persoonlik	PEEDP	P101009600000
ek sien ***jou***	tweede/enkelvoud/gemarkeerd /persoonlik	PTEDP	P201009600000
ek sien ***hom***	derde/manlik/enkelvoud/gemarkeerd /persoonlik	PDHEDP	P311009600000
sy sien ***haar***	derde/vroulik/enkelvoud/gemarkeerd /persoonlik	PDVEDP	P321009600000
ons help ***mekaar***	wederkerig	PW	P000000700000
die man ***wat*** *daar loop*	betreklik	PB	P000000800000

Tabel 9: WS-etikette vir Afrikaanse voornaamwoorde

Stellenbosch University https://scholar.sun.ac.za

B.2 Swartsbank Label sets

Table B.1 Raw Constituency Labelset

Label	Description	Occurences
NP	Noun phrase	3010
PP	Prepositional phrase	2158
NSUBJ	Nominal subject	1290
VP	Verb phrase	1064
S	Sentence root	1020
DOBJ	Direct object	908
SUB_CLAUSE	Subordinate clause	373
ADJP	Adjectival phrase	350
REL_CLAUSE	relative clause	260
COORD_CLAUSE	coordinated clause	232
INF_CLAUSE	infinitival clause	192
NSUBJPASS	passive subject	163
GP	Genitive phrase	116
ADVP	Adverbial phrase	90
MAIN_CLAUSE	Declarative clause (not preceded by conjunction)	44
IOBJ	Indirect object	12
CP	Coordinate phrase	7
EXPL	Expletive phrase	2

Stellenbosch University https://scholar.sun.ac.za

Table B.2 Raw Dependency Labelset

Label	Description	Occurences
det	determiner	2207
prep	preposition	2143
pobj	prepositional object	2100
punct	punctuation	1907
nsubj	nominal subject	1275
root	graph root	1020
mark	clausal marker	996
advmod	adverbial modifier	913
dobj	direct object	909
verb	lexical verb	908
amod	attributive adjectival modifier	837
conj	coordinated conjunct	766
nn	nominal modifier	433
ccomp	closed clausal complement	422
num	numeral / number	394
xcomp	open clausal complement	338
particle	verbal or negation particle	259
rcmod	relative clause	258
acomp	predicative adjectival complement	238
inf	infinitive particle	213
patient	passive subject	162
neg	negator	127
poss	possessive	115
appos	apposition	114
posobj	possessive object	112
aux	auxiliary verb (non-finite)	79
npadvmod	nominal adverbial modifier	63
expl	expletive	58
advcl	adverbial clause	49
predet	predeterminer	24
iobj	indirect object (not governed by preposition)	12
cc	conjunction	9
csubj	clausal subject	8
adp	adposition (used postpositionally)	6

Stellenbosch University https://scholar.sun.ac.za

Bibliography

[1] Abbas, Q. (2012). Building a hierarchical annotated corpus of Urdu: The URDU.KON-TB treebank. *Lecture notes in computer science*, vol. 7181, pp. 66–79.

[2] Abney, S. (1995). Dependency grammars and context-free grammars. Manuscript. Presented at meeting of Linguistic Society of America.

[3] Adger, D. (2003). *Core Syntax: a minimalist approach*. Core linguistics. Oxford University Press, Oxford, United Kingdom.

[4] Ágel, V. (2006). *Dependency and valency: an international handbook of contemporary research*. Walter de Gruyter, Berlin, Deutschland.

[5] Augustinus, L., Dirix, P., Van Niekerk, D., Schuurman, I., Vandeghinste, V., Van Eynde, F. and Van Huyssteen, G. (2016). Afribooms: an online treebank for Afrikaans. In: Calzolari, N. (ed.), *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pp. 23–28. European Language Resources Association, Portorož, Slovenia.

[6] Barnard, E., Davel, M.H. and Van Heerden, C.J. (2009). ASR corpus design for resource-scarce languages. In: *Proceedings of the 10th Annual Conference of the International Speech Communication Association (Interspeech 2009)*, pp. 2847–2850. Brighton, United Kingdom.

[7] Berovic, D., Agic, Z. and Tadic, M. (2012). Croatian dependency treebank: recent development and initial experiments. In: *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC 2012)*, pp. 1902–1906. European Language Resources Association (ELRA).

Stellenbosch University https://scholar.sun.ac.za

[8] Biberauer, T. (2002). Verb second in Afrikaans: is this a unitary phenomenon? *Stellenbosch Papers in Linguistics*, vol. 34, pp. 19–69.

[9] Biberauer, T. (2003). *Verb second (V2) in Afrikaans : a minimalist investigation of word order variation*. Ph.D. thesis, The University of Cambridge.

[10] Bikel, D.M. (2004). Intricacies of Collins' parsing model. *Computational Linguistics*, vol. 30, no. 4, pp. 479–511.

[11] Black, E., Abney, S.P., Flickenger, D., Gdaniec, C., Grishman, R., Harrison, P., Hindle, D., Ingria, R., Jelinek, F., Klavans, J.L., Liberman, M., Marcus, M.P., Roukos, S., Santorini, B. and Strzalkowski, T. (1991). A procedure for quantitatively comparing the syntactic coverage of English grammars. In: *Proceedings of the workshop on Speech and Natural Language (HLT 1991)*, pp. 306–311. Morgan Kaufmann, California, United States of America.

[12] Borges, J.L. (1946). Del rigor en la ciencia. *Los Anales de Buenos Aires 1.3*, vol. 2, p. 53.

[13] Borsley, R.D. (2014). *Syntactic theory : a unified approach*. Routledge, London, United Kingdom.

[14] Bos, J., Bosco, C. and Mazzei, A. (2009). Converting a dependency treebank to a categorial grammar treebank for Italian. In: Passarotti, M., Przepiorkowski, A., Raynaud, S. and van Eynde, F. (eds.), *Proceedings of the Eighth International Workshop on Treebanks and Linguistic Theories (LREC 2012)*, pp. 27—38. Milano, Italia.

[15] Brants, S., Dipper, S., Hansen, S., Lezius, W. and Smith, G. (2002). Tiger treebank. In: *Proceedings of The First Workshop on Treebanks and Linguistic Theories*, pp. 24—-42. Sozopol, Bulgaria.

[16] Buchholz, S. and Marsi, E. (2006). Conll-X shared task on multilingual dependency parsing. In: *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, pp. 149–164. Association for Computational Linguistics, New York, United States of America.

Stellenbosch University https://scholar.sun.ac.za

[17] Candito, M., Perrier, G., Guillaume, B., Ribeyre, C., Fort, K., Seddah, D. and de la Clergerie, E. (2014). Deep syntax annotation of the Sequoia French treebank. In: Calzolari, N. (ed.), *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2014)*, pp. 2298—2305. Reykjavik, Ísland.

[18] Chen, D. and Manning, C. (2014). A fast and accurate dependency parser using neural networks. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 740–750. Association for Computational Linguistics, Doha, Qatar.

[19] Cheng, Y., Asahara, M. and Matsumoto, Y. (2005). Chinese deterministic dependency analyzer: Examining effects of global features and root node finder. In: *Proceedings of the 4th SIGHAN Workshop on Chinese Language Processing*. Jeju Island, Korea.

[20] Chiang, D. and Bikel, D.M. (2002). Recovering latent information in treebanks. In: *Proceedings of the 19th International Conference on Computational Linguistics*, vol. 1 of *COLING '02*, pp. 1–7. Association for Computational Linguistics, Taipei, Taiwan.

[21] Chomsky, N. (1957). *Syntactic structures*. Mouton, The Hague.

[22] Chomsky, N. (1965). *Aspects of the Theory of Syntax*. The MIT Press, Cambridge.

[23] Chomsky, N. (2005). *The minimalist program*. The MIT Press, Cambridge, Massachusetts.

[24] Civit, M., Martí, A. and Bufí, N. (2006). Cat3lb and cast3lb: From constituents to dependencies. In: *Proceedings of the 5th International Conference on NLP (FinTAL 2006)*, pp. 141–152. Turku, Suomi.

[25] Collins, M.J. (1999). *Head-driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, Philadelphia, PA, USA.

[26] Collins, M.J. (2003). Head-driven statistical models for natural language parsing. *Computational Linguistics*, vol. 29, no. 4, pp. 589–637.

Stellenbosch University https://scholar.sun.ac.za

[27] Davel, M.H. and Martirosian, O. (2009). Pronunciation dictionary development in resource-scarce environments. In: *Proceedings of the 10th Annual Conference of the International Speech Communication Association (Interspeech 2009)*, pp. 2851–2854. Brighton, United Kingdom.

[28] De Marneffe, M.C. and Manning, C.D. (2008). Stanford typed dependencies manual.
Available at: http://nlp.stanford.edu/software/dependencies_manual.pdf

[29] De Vos, M. (2005). *The syntax of verbal pseudo-coordination in English and Afrikaans*. LOT, Amsterdam, Nederland.

[30] De Wet, F., Kleynhans, N., Van Compernolle, D. and Sahraeian, R. (2017). Speech recognition for under-resourced languages: data sharing in hidden Markov model systems. *South African Journal of Science*, vol. 113, pp. 1–9.

[31] Delmonte, R. (2009). Treebanking in VIT: from phrase structure to dependency representation. In: Nirenburg, S. (ed.), *Language Engineering for Lesser Studied Languages*, pp. 51–80. IOS Press, Amsterdam, Nederland.

[32] Den Besten, H. and Van der Wouden, T. (2012). *Roots of Afrikaans: selected Writings of Hans den Besten*. Creole language library. John Benjamins Publishing Company, Amsterdam, Nederland.

[33] Deumert, A. (2004). *Language standardization and language change: the dynamics of Cape Dutch*. IMPACT: Studies in Language and Society. John Benjamins Publishing Company.

[34] Donaldson, B.C. (1993). *A grammar of Afrikaans*. Walter de Gruyter.

[35] Drach, E. (1940). *Grundgedanken der deutschen Satzlehre*. Diesterweg, Frankfurt am Main, Deutschland.

[36] Einarsson, J. (1976). *Talbankens skriftspråkskonkordans*. Institutionen för nordiska språk, Lunds universitet, Lunds, Sverige.

Stellenbosch University https://scholar.sun.ac.za

[37] Einarsson, J. (1976). *Talbankens talspråkskonkordans*. Institutionen för nordiska språk, Lunds universitet, Lunds, Sverige.

[38] Eiselen, R. and Puttkammer, M. (2014). Developing text resources for ten South African languages. In: Calzolari, N. (ed.), *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2014)*, pp. 26–31. Reykjavik, Ísland.

[39] Engel, U. (1996). *Deutsche Grammatik*. J. G. Verlag, Heidelberg, Deutschland.

[40] Frege, G., Geach, P.T. and Black, M. (1951). On concept and object. *Mind*, vol. 60, no. 238, pp. 168–180.

[41] Goldhahn, D., Eckart, T. and Quasthoff, U. (2012). Building large monolingual dictionaries at the Leipzig corpora collection: From 100 to 200 languages. In: *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC 2012)*. İstanbul, Türkiye.

[42] Groß, T. and Osborne, T. (2015). The dependency status of function words: auxiliaries. In: *Proceedings of the Third International Conference on Dependency Linguistics (DepLing 2015)*, pp. 111–120. Uppsala University, Uppsala, Sweden.

[43] Grover, A.S., Van Huyssteen, G.B. and Pretorius, M.W. (2011). The South African human language technology audit. *Language Resources and Evaluation*, vol. 45, pp. 271—-288.

[44] Hall, J., Nivre, J. and Nilsson, J. (2007). A hybrid constituency-dependency parser for Swedish. In: *Proceedings of the 16th Nordic Conference of Computational Linguistics (NODALIDA 2007)*, pp. 284–287. Tartu, Estonia.

[45] Han, C., Han, N., Ko, E. and Palmer, M. (2002). Development and evaluation of a Korean treebank and its application to NLP. In: *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC 2002)*. Las Palmas, Gran Canaria, España.

Stellenbosch University https://scholar.sun.ac.za

[46] Helbig, G. (1992). *Probleme der Valenz- und Kasustheorie*. Konzepte Der Sprach- Und Literaturwissenschaft. Niemeyer Max Verlag GmbH, Tübingen, Deutschland.

[47] Heringa, H. and de Vries, M. (2008). Een semantische classificatie van appositie. *Nederlandse Taalkunde*, vol. 13, pp. 60–87.

[48] Hudson, R. (1990). *English word grammar*. Blackwell, Oxford, United Kingdom.

[49] Institut für Maschinelle Sprachverarbeitung (1995). Stuttgart/Tübinger tagsets (STTS). Accessed: 2015-01-05.
Available at: http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/stts.asc

[50] Jakovljevic, B., Kovacevic, A., Secujski, M. and Markovic, M. (2014). A dependency treebank for Serbian: Initial experiments. In: *SPECOM*, vol. 8773 of *Lecture Notes in Computer Science*, pp. 42–49. Springer.

[51] Kahane, S. (1997). Bubble trees and syntactic representations. In: *Proceedings of the 5th Meeting of the Mathematics of the Language, DFKI*, pp. 70–76. Saarbrucken, Deutschland.

[52] Kahane, S. and Gerdes, K. (2009). Speaking in piles: Paradigmatic annotation of French spoken corpus. In: *Proceedings of the 5th Corpus Linguistics Conference*, pp. 1–15. Liverpool, United Kingdom.

[53] Kiperwasser, E. and Goldberg, Y. (2016). Simple and accurate dependency parsing using bidirectional LSTM feature representations. *TACL*, vol. 4, pp. 313–327.

[54] Koutsoudas, A. (1971). Gapping, conjunction reduction, and coordinate deletion. *Foundations of Language*, vol. 7, no. 3, pp. 337–386.

[55] Kreß, C. (2009). *A Comprehensive Analysis of WH-Movement in Interrogative Sentences in English*. GRIN Verlag, München, Deutschland.

[56] Kruger, E. (2011). *The nanosyntactic structure of the Afrikaans passive participle*. Master's thesis, Stellenbosch University.

Stellenbosch University https://scholar.sun.ac.za

[57] Kübler, S. and Prokić, J. (2006). Why is German dependency parsing more reliable than constituent parsing? In: *Proceedings of the 5th Workshop on Treebanks and Linguistic Theories (TLT)*, pp. 7–18. Prague, Czech Republic.

[58] Kübler, S., R. McDonald, J.N. and Graeme, G.H. (2009). *Dependency Parsing*. Morgan and Claypool Publishers, London, United Kingdom.

[59] Lavelli, A. (2011). The Berkeley parser at the EVALITA 2011 constituency parsing task. In: *Working Notes of EVALITA 2011*. Center for the Evaluation of Language and Communication Technologies (CELCT), Povo (Trento), Italia.

[60] Lavelli, A. (2013). An ensemble model for the EVALITA 2011 dependency parsing task. In: *Evaluation of Natural Language and Speech Tools for Italian*, pp. 30–36.

[61] Linguistic Data Consortium (1999). Prague dependency treebank: introduction. Last accessed 2019-02-01.
Available at: http://ufal.mff.cuni.cz/pdt2.0/doc/pdt-guide/en/html/ch01.html#a-intro-context

[62] Maamouri, M. and Bies, A. (2004). Developing an Arabic treebank: methods, guidelines, procedures, and tools. In: *Proceedings of the Workshop on Computational Approaches to Arabic Script-based Languages*, Semitic '04, pp. 2–9. Association for Computational Linguistics, Genf, der Schweiz.

[63] Matthew, G. (2015). Using technology recycling to develop a named entity recogniser for Afrikaans. *Southern African Linguistics and Applied Language Studies*, vol. 33, no. 2, pp. 199–216.

[64] Matthews, P.H. (1981). *Syntax*. Cambridge University Press, Cambridge, United Kingdom.

[65] Mazziotta, N. (2011). Coordination of verbal dependents in Old French: Coordination as a specified juxtaposition or apposition. In: Gerdes, K., Hajicova, E. and Wanner, L. (eds.), *Proceedings of the International Conference of Dependency Linguistics (DepLing)*, pp. 28—-37. DepLing, Barcelona, España.

Stellenbosch University https://scholar.sun.ac.za

[66] McCawley, J.D. (1998). *The Syntactic Phenomena of English*. 2nd edn. Chicago University Press, Chicago, United States of America.

[67] Mel'čuk, I.A. (1988). *Dependency syntax: theory and practice*. State University of New York Press, New York, United States of America.

[68] Müller, P., Ohnheiser, I., Olsen, S. and Rainer, F. (2015). *Word-formation: an international handbook of the languages of Europe*. Handbooks of Linguistics and Communication Science. De Gruyter.

[69] Narasimhan, B., Eisenbeiß, S. and Brown, P. (2007). "two's company, more is a crowd": the linguistic encoding of multiple-participant events. *Linguistics*, vol. 45, pp. 383–392.

[70] Nguyen, D.Q., Dras, M. and Johnson, M. (2016). An empirical study for Vietnamese dependency parsing. In: *Australasian Language Technology Association Workshop (ALTA)*, pp. 143–149. ACL.

[71] Nguyen, T. (2017). *Building a Treebank for Vietnamese syntactic parsing*. Ph.D. thesis, SOKENDAI (The Graduate University for Advanced Studies), Kanagawa, Japan.

[72] Nguyen, T.L., Ha, M.L., Nguyen, V., Nguyen, H. and L Phuong (2013). Building a treebank for Vietnamese dependency parsing. In: *Proceedings of the 2013 RIVF International Conference on Computing and Communication Technologies*, pp. 1–5. Hanoi, Vietnam.

[73] Nivre, J. (2003). An efficient algorithm for projective dependency parsing. In: *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pp. 149–160.

[74] Nivre, J. (2006). Constraints on non-projective dependency parsing. In: *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 73–80. Trento, Italia.

[75] Nivre, J., de Marneffe, M., Ginter, F., Goldberg, Y., Hajic, J., Manning, C.D., McDonald, R., Petrov, S., Pyysalo, S., Silveira, N., Tsarfaty, R. and Zeman, D. (2016). Universal dependencies v1: A multilingual treebank collection. In: Calzolari, N. (ed.), *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC*

Stellenbosch University https://scholar.sun.ac.za

2016), pp. 23–28. European Language Resources Association (ELRA), Portorož, Slovenia.

[76] Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryigi, G., Kübler, S., Marinov, S. and Marsi, E. (2007). MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, vol. 13, no. 2, pp. 95–135.

[77] Nivre, J., Nilsson, J. and Hall, J. (2006). Talbanken05: A Swedish treebank with phrase structure and dependency annotation. In: *Proceedings of the 5th international conference on language resources and evaluation (LREC 2006)*. European Language Resources Association (ELRA), Genova, Italia.

[78] Osborne, T. (2017). Email correspondence, 2017-01-15.

[79] Petrov, S., Barrett, L., Thibaux, R. and Klein, D. (2006). Learning accurate, compact, and interpretable tree annotation. In: *Proceedings of the 21st international conference on computational linguistics and the 44th annual meeting of the association for computational linguistics (ACL-44)*, pp. 433–440. Association for Computational Linguistics, Sydney, Australia.

[80] Petrov, S., Das, D. and McDonald, R. (2012). A universal part-of-speech tagset. In: Calzolari, N. (ed.), *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC 2012)*. European Language Resources Association (ELRA), İstanbul, Türkiye.

[81] Pilon, S. (2005). *Outomatiese Afrikaanse woordsoortetikettering*. Master's thesis, North-West University.

[82] Ponelis, F.A. (1979). *Afrikaanse sintaksis*. Van Schaik, Pretoria, Suid-Afrika.

[83] Ponelis, F.A. (1993). *The Development of Afrikaans*. Duisberg papers on research in language and culture. P. Lang, Frankfurt am Main, Deutschland.

[84] Popel, M., Marecek, D., Stepánek, J., Zeman, D.D. and Zabokrtský, Z. (2013). Coordination structures in dependency treebanks. In: *ACL (1)*, pp. 517–527. The Association for Computer Linguistics.

Stellenbosch University https://scholar.sun.ac.za

[85] Postal, P. and Perlmutter, D. (1977). Toward a Universal Characterization of Passivization. In: *BLS 3*.

[86] Przepiórkowski, A. (2016). Against the argument-adjunct distinction in functional generative description. *The Prague Bulletin of Mathematical Linguistics*, vol. 106, no. 1, pp. 5 – 20.

[87] Puttkammer, M., Eiselen, E., Hocking, J. and Koen, F. (2018). NLP web services for resource-scarce languages. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics-System Demonstrations)*, pp. 43–49. Springer, Melbourne, Australia.

[88] Rademaker, A., Chalub, F., Real, L., Freitas, C., Bick, E. and de Paiva, V. (2017). Universal dependencies for Portuguese. In: *Proceedings of the 4th International Conference on Dependency Linguistics (DepLing)*, pp. 197–206. Pisa, Italia.

[89] Rojek, T. (2009). *Dependenz und Konstituenz: zu Konvergenzen zwischen der Dependenzgrammatik, IC-Analyse und GB-Theorie*. Studien zum Polnisch-Deutschen Sprachvergleisch. Jagiellonian University, Kraków, Polska.

[90] Sampson, G. and Babarczy, A. (2003). A test of the leaf-ancestor metric for parse accuracy. *Natural Language Engineering*, vol. 9, no. 4, p. 365–380.

[91] Sampson, G. and Babarczy, A. (2016). *Grammar Without grammaticality: growth and limits of grammatical precision*. Trends in Linguistics. Studies and Monographs [TiLSM] Series. De Gruyter Mouton, New York, United States of America.

[92] Schmerling, S.F. (1975). Imperative subject deletion and some related matters. *Linguistic Inquiry*, vol. 6, no. 3, pp. 501–511.

[93] Schneider, G. (1998). *A Linguistic Comparison of constituency, dependency and link Grammar*. Master's thesis, University of Zurich. (Unpublished).

[94] Seuren, P.A.M. (2013). *From Whorf to Montague*. Oxford University Press, New York, United States of America.

Stellenbosch University https://scholar.sun.ac.za

[95] Skut, W., Brants, T., Krenn, B. and Uszkoreit, H. (1998). A linguistically interpreted corpus of German newspaper text. *Computing Research Repository (CoRR)*, vol. cmp-lg/9807008.

[96] Tanev, H. and Mitkov, R. (2002). Shallow language processing architecture for Bulgarian. In: *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002)*. Taipei, Taiwan.

[97] Taylor, A., Marcus, M. and Santorini, B. (2003). The Penn treebank: An overview. In: *Treebanks*, pp. 5–22. Springer Science + Business Media, New York, United States of America.

[98] Tesnière, L. (2015). *Elements of structural syntax*. John Benjamins Publishing Company, Amsterdam, Nederland.

[99] Tran, T., Toshniwal, S., Bansal, M., Gimpel, K., Livescu, K. and Ostendorf, M. (2017). Joint modeling of text and acoustic-prosodic cues for neural parsing. *Computing Research Repository (CoRR)*, vol. abs/1704.07287.

[100] Van der Merwe Murray, H. (1977). *A stratificational analysis of Afrikaans syntax and morphology*. Ph.D. thesis, The University of Arizona, Arizona, United States of America.

[101] Van Eynde, F. (2003). Morpho-syntactic agreement and index agreement in Dutch NPs. In: Gaustad, T. (ed.), *Computational Linguistics in the Netherlands 2002: Selected Papers from the Thirteenth CLIN Meeting*, Language and computers: studies in practical linguistics, pp. 111–127. Rodopi.

[102] Wolinski, M. and Rogozinska, D. (2013). Experiments in PCFG-like disambiguation of constituency parse forests for Polish. In: *Language and Technology Conference*.

[103] Wróblewska, A. and Woliński, M. (2012). Preliminary experiments in Polish dependency parsing. In: *Proceedings of the 2011 International Conference on Security and Intelligent Information Systems*, SIIS'11, pp. 279–292. Springer-Verlag, Warsaw, Poland.

Stellenbosch University https://scholar.sun.ac.za

[104] Xue, N., Xia, F., Chiou, F. and Palmer, M. (2005). The Penn Chinese treebank: phrase structure annotation of a large corpus. *Natural Language Engineering*, vol. 11, no. 2, pp. 207–238.

[105] Yamada, H. and Matsumoto, Y. (2003). Statistical dependency analysis with support vector machines. In: *Proceedings of the 8th International Workshop of Parsing Technologies (IWPT2003)*. Nancy, France.

[106] Zeman, D. (2008). Reusable tagset conversion using tagset drivers. In: Calzolari, N. (ed.), *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC 2008)*. European Language Resources Association (ELRA), Marrakech, Morocco.