University of Warsaw
Faculty of Philosophy

Magdalena Borysiak
Record book number: 446267

Dependency structure of English coordination: a surface-syntactic approach

Bachelor's thesis
in the field of Cognitive Science

The thesis was written under the supervision of
prof. dr hab. Adam Przepiórkowski
Faculty of Philosophy, University of Warsaw
Institute of Computer Science, Polish Academy of Sciences

Warsaw 2024

Summary

Key words

Streszczenie

Słowa kluczowe

Contents

Chapter 1

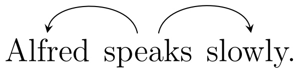
Introduction

Chapter 2

Theoretical background

2.1 Dependency grammars

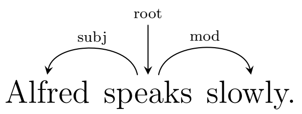
The first full-fledged theory of dependency grammar was proposed by Tesnière (1959, 2015). One of the key ideas included in his work was that a sentence is not comprised solely of its words, but also of the connections between them – dependencies. The connections he proposed were directed, therefore one of the two connected words is always a governor (head) and the other is a dependent. Another crucial element of Tesnière's approach was verb centrality – the verb is the root of every sentence structure in dependency grammar.



(1) Alfred speaks slowly.

Tesnière as an example used the sentence *Alfred speaks slowly*, for which a dependency tree is shown in (1). It visualises the rules described above – all of the words in a sentence are connected, those connections are directed and the verb is central to the whole structure.

Dependency grammars have changed significantly since Tesnière's ideas were published. One example of such changes is the widespread usage of dependency labels, which describe the grammatical function that a word serves – Tesnière differentiated only between actants and circumstants, which can be understood as obligatory dependencies of a verb, which complete its meaning, and optional dependencies, which are not necessary to complete the meaning of the verb. Different corpora have their own ideas for sets of dependency labels, for instance full annotation for (1) could look similarly to (2), with the label `root` marking the central element of the sentence, `subj` marking the subject of the sentence and `mod` marking a modifier of the verb.



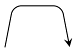
(2) Alfred speaks slowly.

Section 2.2 describes some specific phenomena that can be explained using dependency grammars. Section 2.3 presents some of the ideas for dependency annotation of coordinate structures that have been used in different corpora and Section 2.4 describes the studies that were the basis for the present one.

The last two sections of this chapter delve into more detail about two projects concerned with creating consistent dependency annotation schemes – Universal Dependencies in Section 2.5 and Surface-syntactic Universal Dependencies in Section 2.6.

2.2 Dependency length minimization

Familiarity with dependency grammars helps understand the principle of Dependency Length Minimization (henceforth DLM). It states that natural languages prefer shorter dependencies in their sentences. An example from Hunter and Prideaux (1983), shown in (3), illustrates this.



(3) a. The janitor threw out the rickety and badly scratched chair.

b. The janitor threw the rickety and badly scratched chair out.

The study has shown that speakers deem sentences similar to (3a) more acceptable than the ones similar to (3b).[1] Proposed explanations for this preference are based on language-processing constraints, which are usually said to be caused by working memory limitations. With longer dependencies, while reading or hearing a sentence, a person has to keep certain words in their working memory for a longer time. The longer the dependency, the harder the retrieval of the needed word from the working memory. Similar effects have been found in other studies, both psycholinguistic ones (King and Just, 1991; Gibson, 1998) and those based on corpus research (Gildea and Temperley, 2007, 2010; Dyer, 2023).

The DLM effect has been observed both at the level of usage and at the level of grammar. The level of usage is visible in (3) – when there are multiple grammatical word orders available, people tend to choose those with shortest dependencies, because it makes the sentence easier to understand. As for DLM in grammar, an example taken from (Hawkins, 1994, p. 20) is in (4).

(4) a. * Did $_S$[that John failed his exam] surprise Mary?
b. Did $_{NP}$[that fact] surprise Mary?

Both of the sentences in (4) have a constituent embedded inside of them. In (4a) that constituent is a subordinate clause *that John failed his exam*, whereas in (4b) the constituent is a noun phrase *that fact*. Subordinate clauses are usually longer than noun phrases, therefore dependencies in sentences with embedded clauses can be much longer. According to the DLM hypothesis, this makes the sentence more difficult to process and Hawkins argues that

[1] In the study, it is actually found that it is not the distance between the verb and the particle that affects the acceptability of sentences like those, but the syntactic complexity of the phrases within that distance. However, according to Wasow (2002), syntactic complexity as a measure of dependency length correlates with many others proposed, intervening words included.

due to this processing difficulty sentences with clauses embedded this way are ungrammatical in English. Since noun phrases are usually shorter, sentences similar to (4b) are allowed.

One of the earlier formulations of rules similar to DLM was made by Behaghel (1930). He proposed two laws of word order:

1. That which belongs together mentally is placed close together.
2. Of two sentence components, the shorter goes before the longer, when possible.

The first of these could be understood as DLM, while the other is a consequence of DLM in head-initial languages. Looking at how syntactic trees are usually shaped in a language allows for a judgement on the headedness or directionality of said language – it can be head-initial if most dependencies are directed to right, or head-final if they are mostly directed to the left. English is an example of a head-initial language, therefore the second law proposed by Behaghel holds for English sentences.

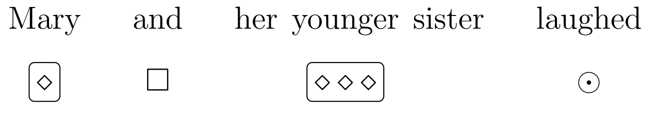
2.3 Possible dependency structures of a coordination

Different corpora choose different approaches to annotating the dependency structure of coordination. Popel et al. (2013) proposed a taxonomy of those, which consists of three families of annotation styles: Prague, Stanford and Moscow. Przepiórkowski and Woźniak (2023) add to those three a London family. All four of those families are described in more detail in the following subsections. Diagrams are used to better illustrate them, where:

- ⊙ is the governor of the coordination;
- each ◇ symbolises a token, grouped together with a few others in a rectangle, forming a conjunct;
- □ is the conjunction of the coordination.

Therefore the sentence *Mary and her younger sister laughed* using this set of symbols would look like in (5).

(5)

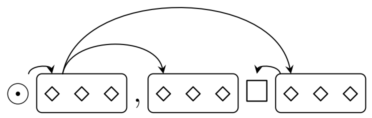


In all of the diagrams in the current section it is assumed that the head of the conjunct is its first word. This assumption is justified, because the work presented here is based solely on the English language, which is mostly head-initial, therefore the diagrams presented here are more likely to be shaped as they are here than in any other way.

2.3.1 Bouquet/Stanford

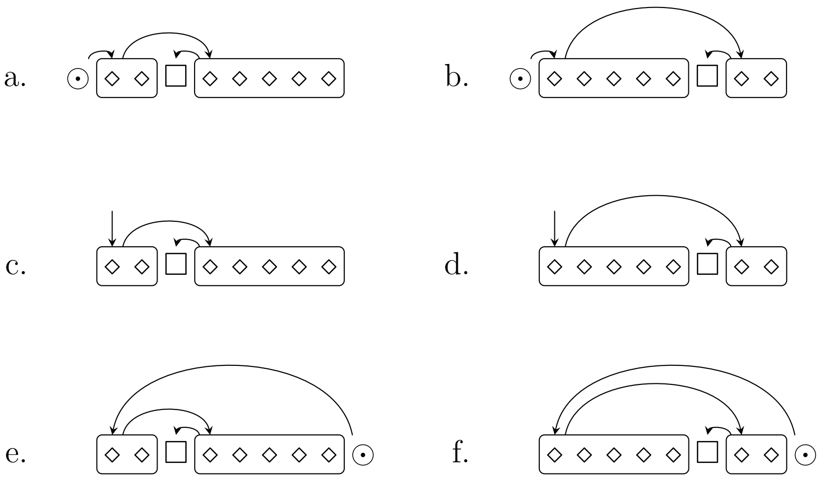
The bouquet structure comes from the Stanford parser (de Marneffe et al., 2006). Coordination with three conjuncts and a governor on the left would be annotated in the bouquet approach as shown in (6).

(6)



In this approach there is a dependency connecting the governor of the coordination to the first conjunct, which is then connected to the heads of each conjunct, thus forming a bouquet. The conjunction is attached to the last conjunct in the structure. This style of annotating is one of the asymmetrical ones, since it does not treat all conjuncts of the coordination equally – it places emphasis on the first conjunct of a coordination by making it the head of every other conjunct.

(7)



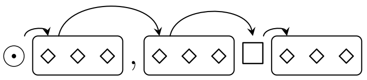
The diagrams in (7) show the bouquet structure with different governor positions and conjunct placements. In (7a–b) the governor is on the left with the shorter conjunct on the left in (7a) and on the right in (7b). Two of the dependencies drawn have the same lenght in both cases, but one of them is visibly longer in (7b). This means that, according to the DLM principle, the structure in (7a) should be preferred, so coordinations with conjuncts of visibly different lengths should have the shorter conjunct on the left.

Within the bouquet approach this is the case for every governor position. Diagrams (7c–d) show the structure of coordination when the governor is absent, with dependencies shorter in (7c) than in (7d), so when the shorter conjunct is on the left. Diagrams (7e–f) show the structure when the governor is on the right, with dependencies shorter in (7e) than in (7f), also when the shorter conjunct is on the left. Therefore within this approach the position of the governor does not influence the order of the conjuncts and the shorter conjunct should always be placed on the left.

2.3.2 Chain/Moscow

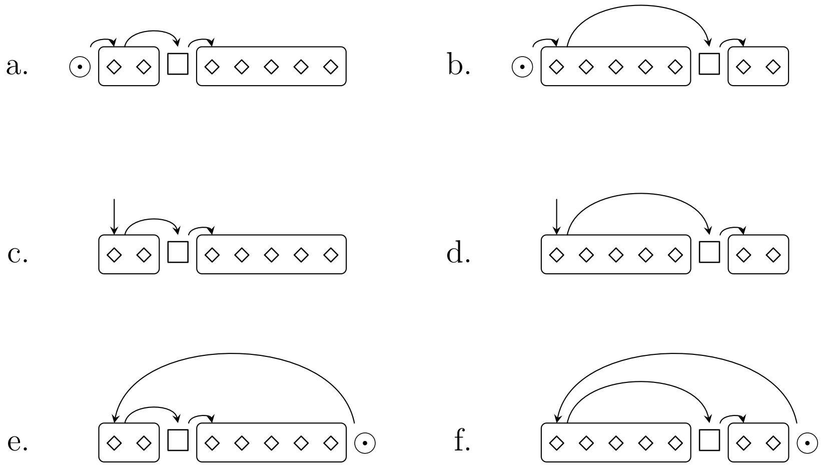
Another asymmetrical approach is the chain, or Moscow one. It is utilised in the Meaning-Text Theory proposed by Mel'čuk (1988).

(8)



The structure is created by connecting the governor to the first conjunct of the coordination, then every element of the coordination (including the conjunction) to the next one. As the dependency between the governor and the coordination is specifically between the governor and the first conjunct, the chain approach is another example of an asymmetrical annotation style.

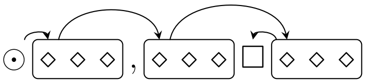
(9)



The diagrams in (9) show that, in this case, similarly to the bouquet approach, the placement of the shorter conjunct on the left should be preferred, assuming the DLM principle. The placement of the governor again does not seem to have an effect on the ordering.

A variation of this annotation style is now used in the Surface-syntactic Universal Dependencies scheme (described in Section 2.6), which is based on the Universal Dependencies. The authors chose the Chain style instead of the Bouquet style, as it minimizes the dependency lengths, which the authors wanted to be reflected in the syntactic structure. The difference between their style and the one shown in (8) is that the conjunction is not attached to the preceding conjunct, but to the following one. This annotation is illustrated in (10).

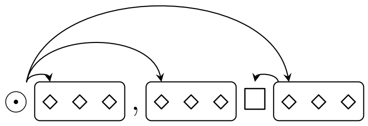
(10)



2.3.3 Multi-headed/London

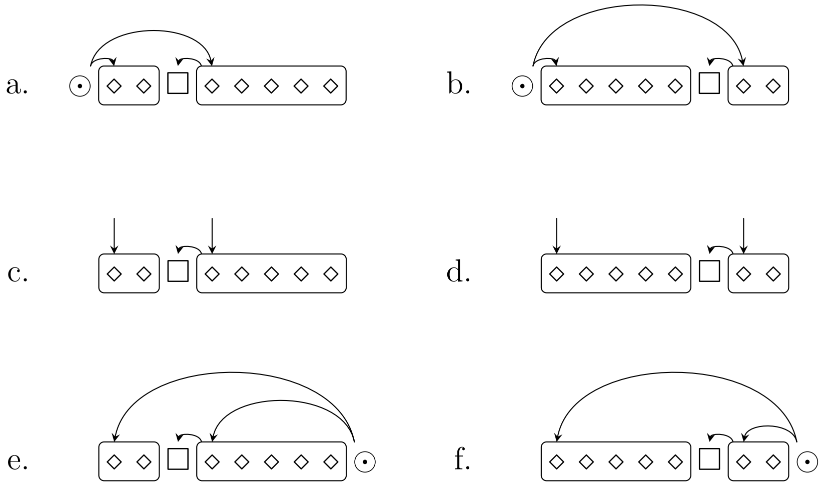
The symmetrical approaches, as the name suggests, treat every conjunct in the coordination the same way. One of them is the multi-headed, or London approach, for which Przepiórkowski and Woźniak (2023) propose the name based on its appearance in Word Grammar developed by Hudson (1984, 2010) at University College London.

(11)



The symmetry of the approach comes from the fact that no conjunct is distinguished by being the only direct dependent of coordinations governor. Instead, all of the conjuncts have dependencies connecting them to the governor, and the conjunction is dependent on the last conjunct, similarly to the bouquet approach.

(12)

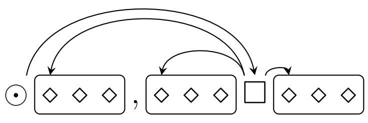


Here, the predictions for the preferred ordering of conjuncts are different from those in asymmetrical approaches. According to this approach the placement of the governor influences the conjunct ordering, specifically that the shorter conjunct will tend to be placed near the governor: with the governor on the left, as in (12a–b), the dependencies are shorter with the shorter conjunct placed on the left, and with the governor on the right, as in (12e–f), the shorter conjunct should be placed on the right. In the case of no word governing the coordination, as in (12c–d), there seems to be no preference for any ordering.

2.3.4 Conjunction-headed/Prague

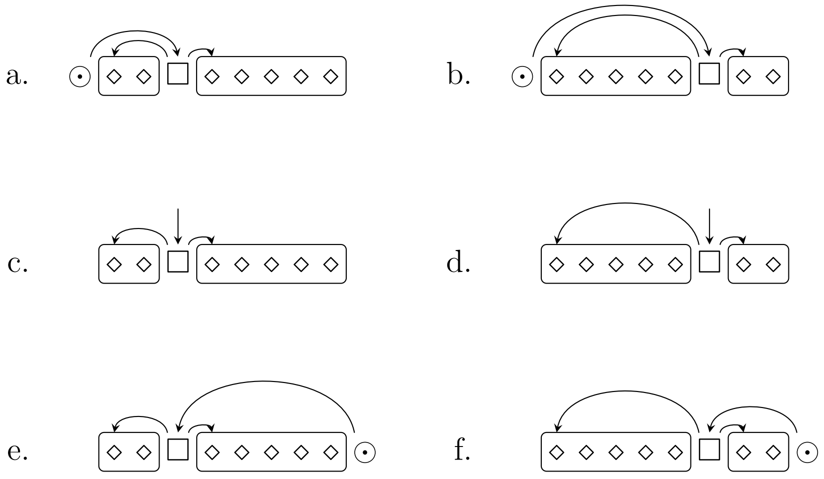
The last approach discussed here is the one associated with the Prague Dependency Treebank, called the Prague approach by Popel et al. (2013) or the conjunction-headed by Przepiórkowski and Woźniak (2023).

(13)



This is another example of the symmetrical styles, as here again the governor treats all of the conjuncts the same way – in this case, it does not connect to any of them. Instead, there is a dependency connecting the governor and the conjunction, which then has the conjuncts of the coordination as its dependents. In the case of a coordination without a conjunction, the governor would connect to a punctuation mark.

(14)



This annotation style again generates new predictions about the preferred ordering of conjuncts. With the governor placed on the left and without any governor present, this approach should prefer to have the shorter conjunct on the left side of the coordination. With the governor on the right, this approach predicts that ordering does not matter – in both cases presented here the sum of dependency lengths is the same.

2.4 Previous studies

The current study is a replication of Przepiórkowski and Woźniak (2023), which researched coordinate structures to find out whether ordering of conjuncts in English is as simple as placing shorter conjuncts on the left or whether the placement of the governor of the coordination has some influence on the ordering. They used the Penn Treebank, which is an annotated corpus of texts from the Wall Street Journal. This relatively small, but high quality dataset allowed them to make an argument for the symmetric styles of annotating coordination. Figure 2.1 shows how modelled proportions of coordinations with the shorter conjunct placed on the left changed with growing differences in conjunct length, here measured in words. When the governor is on the left or it is absent altogether, the proportions grow with length differences. This means that it is more likely that the shorter conjunct will be on the left when coordinating *some apples* and *the oranges your mother gave you* than when coordinating *apples* and *oranges*. No such tendency was found when the governor is on the right.

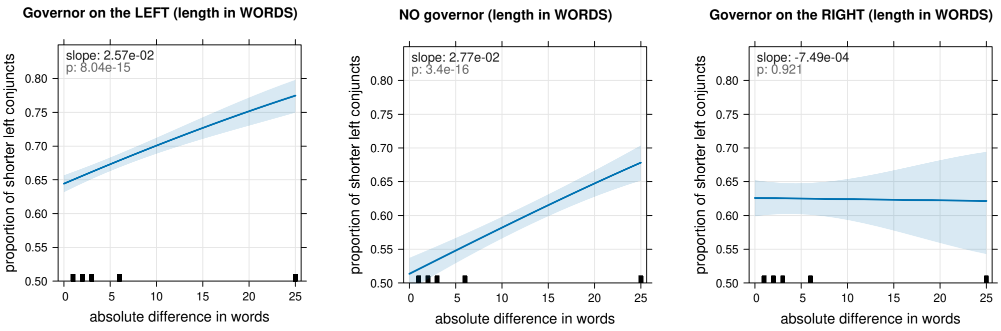


Figure 2.1: Modelled proportions of coordinations with left conjuncts shorter depending on difference in conjunct lengths from Przepiórkowski and Woźniak (2023)

Those results are compatible with the predictions of the symmetric annotation styles: the Prague one, assuming DLM working at the level of usage and the London one, assuming also DLM at the level of grammar. An example of DLM in grammar was given in Section 2.2, but it was not described how it could present itself in coordinations. As was mentioned previously, English is a mostly head-initial language, which in coordinations means that the governor is usually on the left. When the governor is in fact on the left, the dependencies are shortest when the shorter conjunct is also on the left. There is therefore a grammatical pressure to always put shorter conjuncts on the left, because it should usually lead to shorter dependencies in total. This means that when the coordination has no governor and there is no immediate pressure to order the conjuncts in any way, the shorter conjunct still may be placed on the left, because there is a grammatical pressure to do so. However as Przepiórkowski and Woźniak (2023) point out, this pressure may be reduced when the length differences between conjuncts are noticably bigger, because then the DLM effect at the level of usage is stronger.

Przepiórkowski et al. (2024) have already conducted a replication study of this research. The aim was to see whether the conclusions drawn in the original study hold up when the data come from a bigger and more diverse corpus. The results are presented in Figure 2.2 – slightly different from what was found in the original study, but they sharpened the original conlusions. In the bigger dataset the coordinations with the governor on the left and without a governor behave the same – with growing length differences between conjuncts grows also the proportion of shorter left conjuncts. The difference is that, with the governor on the right, proportion of coordinations with the shorter conjunct on the left decreases with the growing length difference.

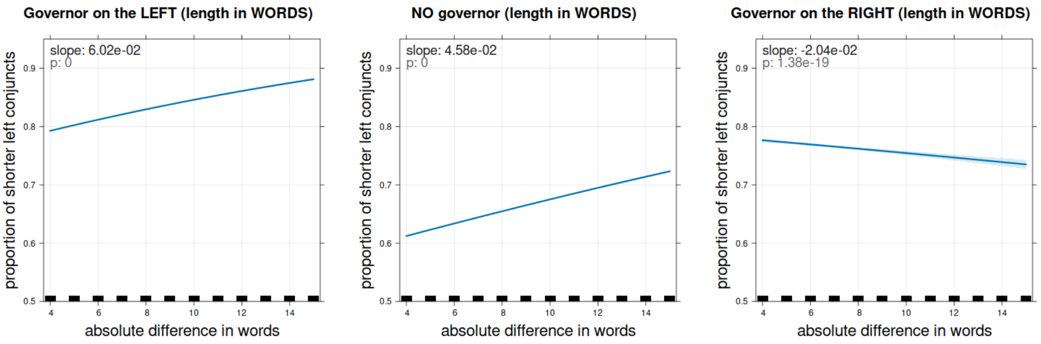


Figure 2.2: Modelled proportions of coordinations with left conjuncts shorter depending on difference in conjunct lengths from Przepiórkowski et al. (2024)

Those results point to only one of the annotation styles, namely the London one. Within this style, dependencies are minimised when the shorter conjunct is placed closer to the governor if there is one. If the coordination has no governor, neither placement should be preferred, unless DLM at the level of grammar is taken into account – then the shorter conjunct should be preferred on the left, since that usually helps minimise the dependencies.

Przepiórkowski and Woźniak (2023) conducted their study on a relatively small, but manually annotated corpus. Przepiórkowski et al. (2024) replicated that study on a larger, but automatically annotated corpus. This automatic annotation resulted in a poor quality of data – after evaluating the coordinations extracted for analysis, they found only 50.1% of their sample to be correctly extracted. This study attempts the replication again, with the same larger corpus but aiming to improve the quality of the automatic annotation.

2.5 Universal Dependencies

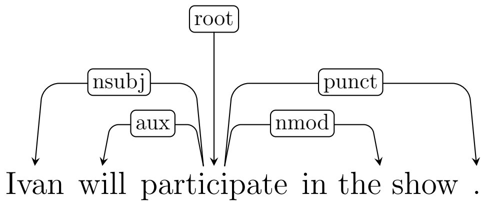
As seen in Section 2.3, there are many ideas on the structure of coordination. Universal Dependencies (UD, henceforth) is a project focused on formulating guidelines for creating dependency annotation, that would suit as many languages as possible, while maintaining the possibility to represent phenomena specific to any given language. The guidelines outline how one should deal with word segmentation, part-of-speech tagging, assigning morphological features and creating an appropriate dependency structure for a sentence.

Here, the most relevant part of the project are the rules for creating a dependency structure. In the first version of UD (Nivre et al., 2016), three of them were specified:

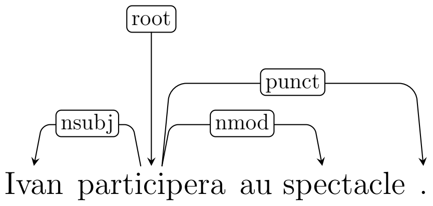
1. dependency relations appear between content words,
2. function words are attached to the content words which they describe,
3. punctuation marks are attached to the head of the phrase or clause in which they appear.

Content words chosen here for dependency heads can otherwise be called “lexical” or “semantic” centres, whereas function words serve mostly a syntactic purpose in a sentence. In sentence (15), the word *participate* carries the meaning, therefore it is the content word and the root of the sentence. The word *will* is a function word and is attached to the content word.

(15)



(16)



The reasoning behind setting those criteria is that it increases the chance of finding similar tree structures in different languages, for example when comparing sentences between English and French, which is morphologically richer. (16) is a tree for the French translation of the sentence in (15). Even though the French sentence does not have an auxiliary word to mark the future tense, the structures of those sentences are almost identical, which would not be possible if UD chose to make functional words governors of content words.

2.6 Surface-syntactic Universal Dependencies

Surface-syntactic Universal Dependencies (SUD, henceforth) is another example of a project aiming to create a set of universal guidelines for dependency annotation. Gerdes et al. (2018) describe it as "near-isomorphic to UD" and propose a set of conversion rules between the schemes. Most of the SUD annotation guidelines are the same as in UD, the most prominent difference is the change in choosing dependency heads, from content words to function words. This section covers the relevant differences between the schemes and how those are beneficial for research described in this work.

2.6.1 Criteria for choosing heads of dependencies

Instead of favouring content words, SUD uses the distributional criteria for choosing dependency heads, which means that "the surface syntactic head determines the distribution of the unit" (Gerdes et al., 2018). It can be tested by checking which of the words within a dependency behaves in sentences similarly to the way the whole unit does, so which of the words can be replaced by the unit and *vice versa*. The example sentence the authors use to explain their criteria is *The little boy talked to Mary*. There is a dependency between words *little* and *boy*, and sentences in (17) and (18) show why the head of this dependency is the word *boy*. In (17a) the word *boy* can be replaced by the unit *little boy*, as shown in (17b), and the sentence is still grammatical.

(17) a. I saw a **boy**.
b. I saw a **little boy**.

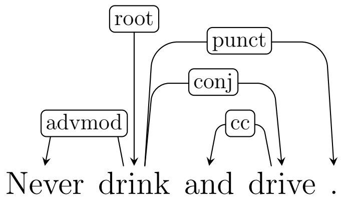
The same is not the case for the word *little*. Trying to replace that word with the whole unit in (18a) results in an ungrammatical sentence in (18b). Sentences (18c–d) show that replacement in the other way is not possible either, therefore the word *little* cannot be the head of this dependency.

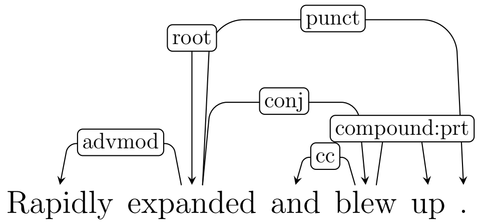
(18) a. The boy was **little**.
b. * The boy was **little boy**.
c. I found the **little boy**.
d. * I found the **little**.

It is not always possible to test both of the words within a dependency, but in such cases showing that one of the words does not commute with the whole unit is enough to decide it is not the head, therefore the other one must be. As shown in Gerdes et al. (2018), that is exactly the case with the words *to Mary* – it is impossible to see how the word *to* behaves on its own, as it needs a noun or a verb, but the sentences in (19) show that *Mary* does not have the same distribution as those two words together.

(19) a. I saw **Mary**.
b. * I saw **to Mary**.
c. I talked **to Mary**.
d. * I talked **Mary**.

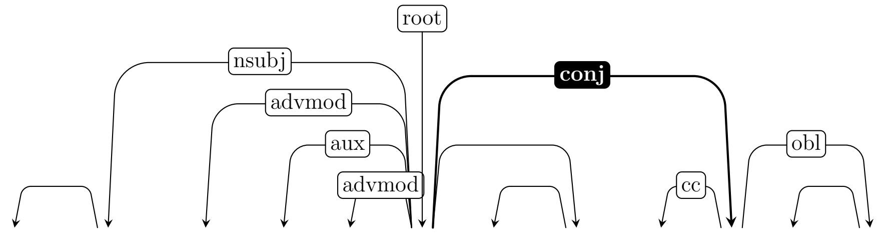
This key difference between UD and SUD has crucial consequences for extracting the exact length of conjuncts in a coordinate structure, which is an integral part of this work. As Przepiórkowski and Woźniak (2023) mention, the UD scheme is not ideal for coordination analysis, as it is not clear which dependencies are shared by the conjuncts and which are private. This is illustrated by the sentences in (20) and (21).

(20)


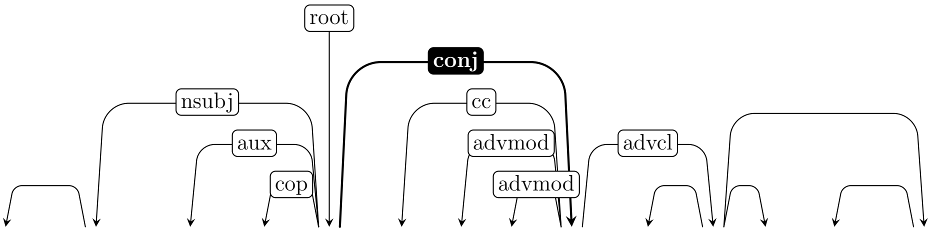
(21)


In (20) the coordinated words are *drink* and *drive*, the word *never* is attached to the first conjunct and even though English speakers reading this sentence know that it applies to the whole coordination, it is not obvious from the structure of the sentence. The structure in (21) is similar, but this time the word *rapidly* applies only to the first conjunct. In both of those sentences knowledge of the real world is required to accurately determine whether the dependency is shared by the whole coordination (as in (20)) or private to the first conjunct (as in (21)). Because of this ambiguity, it is difficult to construct accurate heuristics for determining the extent of each conjunct in a coordination. This issue appears in both UD and SUD, however the following examples show that some of the ambiguities present in UD can be resolved in SUD.

Based on heuristics used by Przepiórkowski et al. (2024), the tree in (22) would give a coordination with conjuncts *reach the surface* and *cools at depth*. This is, because the head of the left conjunct, *reach*, has on its left side dependencies labeled `nsubj`, `advmod` and `aux`, but the head of the right conjunct, *cools*, does not have any of those dependencies. Therefore all of those dependencies are predicted to be shared by both conjuncts. The correct parse of this sentence gives a coordination with conjuncts *does not reach the surface* and *cools at depth*, so the dependencies *This magma often* should not be shared, but private to the first conjunct.



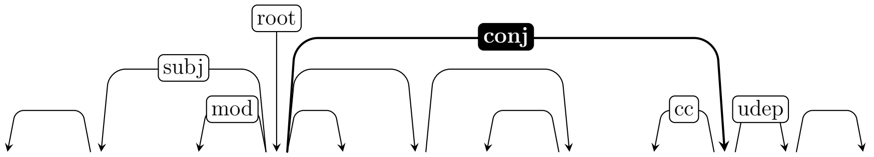
(22) This magma often does not reach the surface but cools at depth.[2]



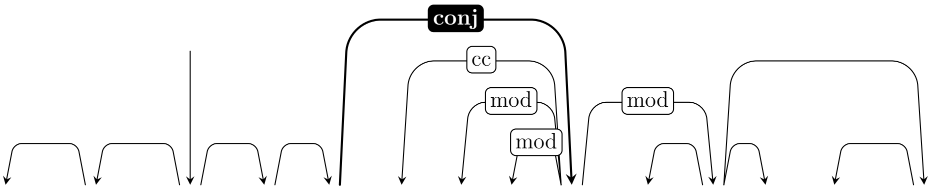
(23) Ballet shoes should be snug, but not so tight they cut off blood flow.[3]

Modifying the heuristics, so that they fit this example, for instance by saying that `aux` dependencies should always be included in the left conjunct, would on the other hand mean that sentences such as in (23) would have incorrect extracted coordinations. In this example the correct coordinate structure has conjuncts *snug* and *not so tight they cut off blood flow*, but if the algorithm included the `aux` dependency in the first conjunct (as would be required in (22)), the result would be conjuncts *should be snug* and *not so tight they cut off blood flow*.

This however is not an issue when using the SUD scheme. As shown in (24), the words *does not* cannot be dependencies of the whole coordination, because the word *does* is the head of the left conjunct and the word *not* is one of its dependencies on the right and thus is always included in the conjunct. Changing the annotation scheme to SUD does not affect the sentence in (23) – the word *snug* has to be the whole left conjunct, because it also does not have any dependencies in this annotation scheme.



(24) This magma often does not reach the surface but cools at depth.[4]



(25) Ballet shoes should be snug, but not so tight they cut off blood flow.[5]

[2] Sentence `w01031015` from the `UD_English-PUD` corpus (Zeman et al., 2017).

[3] Sentence `GUM_whow_ballet-14` from the `UD_English-GUM` corpus (Zeldes, 2017).

[4] Sentence `w01031015` from the `SUD_English-PUD`.

[5] Sentence `GUM_whow_ballet-14` from the `SUD_English-GUM` corpus.

Therefore, the focus on syntax in SUD makes it a better fit for coordination analysis.

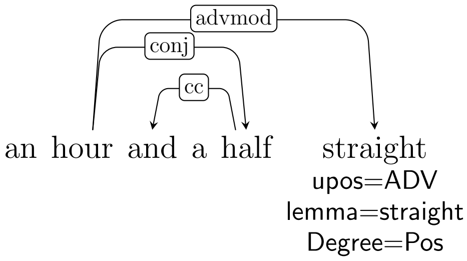
2.6.2 Explicit information about shared dependencies

Besides the structural advantages that SUD has over UD when it comes to analysing coordination, there is one additional feature that is added in SUD treebanks that helps find the extent of conjuncts.

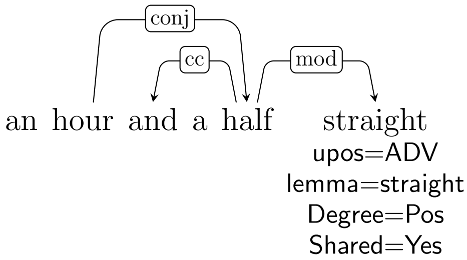
While UD corpora are often created specifically for the purpose of participating in the UD project or converted with manual corrections from different dependency annotations, the SUD corpora are mostly converted automatically from UD. There are a few French treebanks, as well as treebanks for Beja, Zaar, Chinese and Naija, that are natively made for SUD, but all others are converted from UD using rule-based graph transformation grammars, which are described in more detail in Chapter 3.

As was mentioned in Section 2.3, UD uses the Bouquet approach to annotating coordination, while SUD uses the Chain one. This means that in the conversion process, some information about the privacy status of a dependency of the coordination can be lost. This is visible in the coordination presented in the sentence *I just sat in there for like an hour and a half straight and studied.*[6] As the UD annotation in (26) shows, the word *straight* is shared by the whole structure. This is not structurally visible in the SUD version in (27), where the `mod` dependency for the word *straight* is attached to the last conjunct.

(26)



(27)



So as not to lose this information while converting the annotation scheme, feature Shared=Yes is added. Similarly, in coordinations where a dependent is attached to the right conjunct in the UD scheme (therefore private to the right conjuct), during the conversion to SUD the feature Shared=No is added.

2.6.3 Learnability of dependency schemes

The current study is another replication of Przepiórkowski and Woźniak (2023). As was pointed out in Chapter 1, Przepiórkowski et al. (2024) have conducted a similar analysis on the COCA corpus annotated automatically in the UD scheme. After evaluating the automatically annotated data they found only 50.1% of the coordinations in the evaluation sample to be correctly extracted from the corpus. Reasons for such an outcome can be twofold: the issues lie either within the parsing accuracy or within the script for extracting coordinations from dependency trees.

[6] Sentence `GUM_vlog_studying-27` from the GUM corpus (Zeldes, 2017).

Issues within the script have been addressed to some extent in Section 2.6.1 – heuristics for finding conjunct extents are easier to develop within a function-word focused annotation scheme. As for the parsing performance, there are studies showing that the UD scheme is harder to parse. Rehbein et al. (2017) show that choosing content words rather than function words for dependency heads increases arc direction entropy (a measure describing how consistent dependency directions in a given treebank are), which then lowers parsing accuracy. In another study, Kohita et al. (2017) converted UD trees into ones with function heads, rather than content heads. They then used the converted trees for training parsers and parsed 19 treebanks using both UD and converted models. After parsing, the results from the converted models were converted back to UD and for most of the languages (11 out of 19) those results had better scores.

The criterion for choosing dependency heads may not be the only structural advantage that SUD has over UD in terms of parsing. As Gerdes et al. (2018) demonstrate, the Chain approach to annotating coordination, that is used in SUD, minimises the dependency lengths compared to the Bouquet approach used in UD. This may be beneficial for parsing accuracy, as parsers tend to perform better when working with shorter dependencies (Nilsson et al., 2006; Eisner and Smith, 2005).

In the studies cited above the comparison was between UD and a scheme that differed from UD only in some particular aspect, not a new, comprehensive scheme. Tuora et al. (2021), however, compared UD to SUD, which matters because, as they say, "any realistic annotation schema which employs a more 'syntactic' approach to headedness than UD will also differ from UD in the repertoire and distribution of dependency labels, and will also take into account the intrinsic linguistic interaction between various constructions". They trained five parsers, two of which were transition-based and three graph-based, using 21 corpora representing 18 languages. While transition-based parsers seemed to perform similarly on both annotation schemes, the graph-based ones preferred SUD. As for attachment scores for the English corpus tested in this experiment (GUM), all of the parsers scored higher with the SUD annotation. The parser utilised in the current study, Stanza, is graph-based and the language of the texts it annotates is English, therefore SUD might be the better choice for the annotation scheme for this data.

Chapter 3

Data processing

The data used in this work is based on the Corpus of Contemporary American English. The corpus consists of raw texts collected in a span of 30 years (1990 – 2019) representing 8 styles: academic, fiction, newspapers, magazines, TV/movies, websites, blogs and spoken data. For the analysis of coordinations to be possible, first the texts have to be annotated syntactically – here the Stanza parser (Qi et al., 2020) was chosen for this task. The first subsection of this chapter describes how the parser works and how it was trained for annotation. The second subsection describes the procedure of finding coordinate structures in parsed sentences and creating tables with data ready for analysis.

3.1 Parser training

Stanza is a Python package intended for natural language analysis, which contains multiple processors responsible for different steps of said analysis, e.g. tokenisation, lemmatisation, part-of-speech tagging, dependency parsing, sentiment analysis. All of the processors are neural networks, which together are put into a pipeline that takes raw text as input and returns documents with parsed sentences as output. The default parsing model provided by Stanza for English annotates according to the UD scheme, therefore two processors – part-of-speech tagger and dependency parser – had to be trained to use the SUD scheme.

The dependency parser creates the dependency trees that can later be searched for coordinations. The part-of-speech tagger assigns the part of speech as well as the features appropriate for each word in the input. The parts of speech used in SUD are the same as in UD, but the features may include additional information about shared dependencies, as was explained in Section 2.6.2. If the neural network is trained on data containing this information, it can than be able to determine which of the dependencies are shared and which are private to specific conjuncts.

Dependency trees as shown in previous chapters, though possibly comprehensible for people, are not written in a way that is easily understandable by computer programs, including parsers. Buchholz and Marsi (2006) created the CoNLL-X format, which was first intended for the comparison of parser outputs in a dependency parsing shared task. Today this format is widely used for representing dependency trees in a plain-text form. The UD project adapted the format to their needs by replacing some of the information included in CoNLL-X and thus creating the CoNLL-U format, now also used for SUD data. Appendix A shows an example of an SUD dependency tree presented as a tree and in the CoNLL-U format.

Training was conducted using the scripts made available by the Stanza developers.[1] The models were trained on the English SUD corpora, which were created by converting the UD corpora into SUD using a set of graph conversion rules developed by the authors of SUD (Gerdes et al., 2018).[2] If a corpus is large enough, it is split into three parts: training, development and testing. The training set is used to expose the parser to the correct dependency trees and based on that a prediction model is created. The model is tuned using the development set – the model tries to predict what is the correct dependency tree for a sentence and the prediction is then confronted with the data in the set. Adjustments are made until there is no gain in the scores acheived by the model.

Training a model requires word vector data (which is provided with the default model for English) and a prepared treebank – this means that all of the possible annotations from a treebank have to be listed for the prediction model to choose from. After all the needed files were provided, the training script was run. The batch size was set to 1000 and the dropout rate was 0.33. This means that the whole training dataset was split into batches, each with 1000 elements, which were then given to the model to assign weigths to different possible parses of a sentence. Dropout rate is the proportion of nodes in the neural network that are dropped during training. Without any dropout, a model might become overfitted for the training data. This means that it will be very good at predicting annotations for the sentences it has already seen, but not so much with any other data. The dropout rate chosen for training here was recommended in the training documentation, the batch size was dictated by the hardware limitations.

The following subsections describe the corpora used to train the models for this study.

3.1.1 Combined model

The combined model was used to annotate most of the data analysed in this study. It was trained on the combined training sets from the EWT, GUM and ParTUT corpora, all available converted to the SUD annotation scheme.[3] A corresponding model was also trained for the UD scheme to compare the performance on those two schemes.

EWT is the English Web Treebank. The data was collected between the years 1999 and 2011 and comes from 5 primary sources: weblogs, newsgroups, emails, reviews and question-answers. In total, the corpus contains 254,820 words, which makes it the biggest available SUD corpus for English. The

[1] https://github.com/stanfordnlp/stanza-train

[2] https://github.com/surfacesyntacticud/tools/blob/v2.12/converter/grs/UD_to_SUD.grs

[3] https://surfacesyntacticud.github.io/data/

texts originally had constituency annotation, which was then automatically converted into Stanford Dependencies and then manually corrected to UD.

The second corpus used for this model was GUM – the Georgetown University Multilayer corpus. The early versions of GUM were annotated according to the Stanford Dependencies scheme, later they were manually converted into UD and the subsequent additions to the corpus have been annotated natively using UD. The corpus contains 228,399 tokens and is made up of a variety of styles, for instance academic, interviews, travel guides, letters, how-to guides and forum discussions.[4]

The third corpus used for training the combined model was one based on the English part of ParTUT, the multilingual parallel treebank from the University of Turin. It consists of legal texts, Wikipedia articles and transcriptions of TED Talks. It was originally manually annotated in a style specific to the treebanks developed at the University of Turin, then converted to UD. The corpus has 49,602 tokens, which is a lot less compared to the corpora described earlier, but is still a significant contribution to the model.

Corpora listed above were chosen because of their size and the consistency of annotation. Some corpora, despite the style diversity they could provide for the parsing model, had to be excluded from training, because some information was missing or annotated inconsistently with the other corpora used here.

3.1.2 Spoken model

Spoken and written language differ significantly, therefore a model trained mainly on one type of data can perform poorly when presented with the other type. Most of the corpus data is from written text, as it is easier to obtain. This experiment involves training a model specialising in spoken data to avoid the poor quality resulting from an ill-fitted model. The corpora used for this model were parts of the GUM corpus, specifically those with interviews, conversations and vlogs (51,451 tokens) and the Atis corpus. Atis comprises sentences from the Airline Travel Informations dataset, which come from transcriptions of people asking automated inquiry systems for flight information. The corpus has 61,879 tokens and was natively annotated in UD.

[4] The GUMReddit corpus, which contains the forum discussions, was here included in the whole GUM corpus. Before training any models it is required to run a script that recovers the textual data that is by default not included. The script is available in the GUM corpus repository: https://github.com/amir-zeldes/gum/blob/master/get_text.py.

name	no. of tokens	source texts	annotation style
combined model			
EWT	251 492	weblogs, newsgroups, emails, reviews and question-answers	constituency, then converted to Stanford Dependencies, then to UD
GUM	228 399	academic, Wikipedia articles, vlogs, conversations, courtroom transcripts, essays, fiction, forum, how-to guides, interviews, letters, news stories, podcasts, political speeches, textbooks, travel guides	Stanford Dependencies, then converted to UD
ParTUT	49 602	legal texts, Wikipedia articles, public talk transcripts	own annotation style, then converted to UD
total	529 493		
spoken model			
Atis	61 879	transcriptions of questions about flight information	natively UD
GUM (parts)	51 451	interviews, conversations, vlogs	Stanford Dependencies, then converted to UD
total	113 330		

Table 3.1: Summary of the information about corpora used to train models

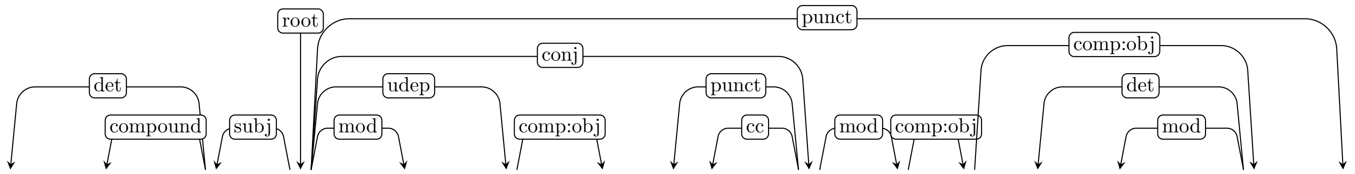
3.2 Data extraction

The scripts used for parsing and extracting data are available in a github repository.[5] The extraction process will be illustrated by the sentence *The Bernoulli family came originally from Antwerp, but emigrated to escape the Spanish persecution.*[6]

Texts from the COCA corpus are first split into sentences using the Trankit parser (Nguyen et al., 2021), which deals with the task more accurately than Stanza. Those sentences are then put into the Stanza's parsing pipeline: first the sentences are tokenised, then lemmas of all of the words in the sentence are found, parts of speech and morphological features are assigned and finally the dependency trees for all of the sentences are created. After running through the pipeline, the example sentence has a dependency tree shown in (28).

[5] https://github.com/bmagdab/sud-coords

[6] Modified version of the sentence GUM_bio_bernoulli-9 from the GUM corpus (Zeldes, 2017)



(28) The Bernoulli family came originally from Antwerp , but emigrated to escape the Spanish persecution .

Every dependency tree is then searched for coordinations, which are marked by the dependency label `conj`. In (28) there is one `conj` dependency that connects the words *came* and *emigrated*. If such a dependency is found, the algorithm looks for every conjunct within that coordination and checks whether there are any other coordinations embedded inside of the one already found – if there are any, they are separated and analysed later. After all coordinations in a document are found, the algorithm searches for all information necessary for later analysis: the conjunction, heads of conjuncts, the exact text and length of the left and right conjunct, the governor position and additional information about parts of speech and morphological features of all of the elements of the coordination. In the example (28) the heads of the left and right conjuncts are the words *came* and *emigrated* respectively. The word *do* has no head in this sentence, therefore there is no governor of the coordination. If the head of the right conjunct has a `cc` dependency, that dependency is the conjunction of the coordination – in (28) this is the word *but*.

Then the algorithm looks for the text of the right conjunct – it does not add to the conjunct the dependency labelled `cc`, because this is a seperate element of the coordination. It does not add the `punct` dependency either, if it appears at the beginning of the conjunct, because a conjunct has to start with a word. The only other dependency that the word *emigrated* has is `mod`. Since it appears after the head of the right conjunct, it can be either private to the conjunct or shared by the whole coordination. The heuristic applied here after Przepiórkowski et al. (2024) is that if any of the other conjuncts have a dependency like this, this dependency is private. Otherwise it is shared by the whole coordination. In this case the only other conjunct is headed by the word *came* and it does have a `mod` dependency, therefore each head has a private `mod` dependency that is included in the appropriate conjunct. After all direct dependencies of the head are covered, the whole branches starting with those direct dependencies are added to the conjunct. This means that since the word *to* from the `mod` dependency has been added to the conjunct, this words dependencies are also added – here this is the word *escape*. This continues until there are no more nodes in the branch of the tree to add. The text of the right conjunct is found this way and it is *emigrated to escape the Spanish persecution*. The process is then repeated for the left conjunct. In (28) the word *came* has three dependencies: `subj`, `mod` and `udep`. The rule here is mirroring the one from the right conjunct – dependencies appearing on the left side of the left conjunct are private to the conjunct if any of the other conjuncts have the same dependency, otherwise they are shared by the whole coordination. The `mod` and `udep` dependencies appear after the head of the conjunct and are therefore automatically added to the conjunct. The `subj` dependency has to be checked – this time the algorithm finds that no other conjunct has the same dependency, therefore this dependency has to be shared by the whole coordination and is not included in the left conjunct. The text

	governor.position	governor.word	conjunction.word	no.conjuncts
	0		but	2

L.conjunct	L.dep.label	L.words	L.syllables	L.chars
came originally from Antwerp	root	4	9	28

R.conjunct	R.dep.label	R.words	R.syllables	R.chars
emigrated to escape the Spanish persecution	conj	6	14	43

sentence
The Bernoulli family came originally from Antwerp, but emigrated to escape the Spanish persecution.

Table 3.2: An example of a table with the extracted information about coordinations. Some columns are excluded for simplicity.

of the left conjunct is found to be *came originally from Antwerp*.

The last step is measuring the lengths of both conjuncts in characters, syllables and words. All of the information found during this process is put in a table similar to the one in (3.2).

Chapter 4

Statistical analysis

4.1 UD and SUD model comparison

The first hypothesis this work aims to verify is that the parsing model trained to annotate according to the SUD scheme will perform better than the one annotating according to the UD scheme.

Corpora chosen for training are all split into three parts each: training, validation and testing. The training and validation splits were used during the training process. For evaluation, the models predicted the correct annotation for every sentence in the testing split. Those predictions were then compared to the original annotations for those sentences and two metrics were calculated: unlabelled and labelled attachment score. Unlabelled attachment score, or UAS, is the percentage of words in a sentence that have a correctly predicted governor. Labelled attachment score, or LAS, is the percentage of words that have a correctly predicted governor as well as the dependency label that connects that word to its governor. Those scores for both the UD and SUD model are presented in Table 4.1 along with the differences between those accuracies. Both of the differences are significant and in both cases it is in favour of the UD model.[1]

UAS			LAS		
UD	SUD	Δ	UD	SUD	Δ
89.75	88.95	**0.8**	87.29	86.80	**0.49**

Table 4.1: Attachment scores for the models trained on combined UD and SUD corpora and the difference between them

Thus the first hypothesis was not confirmed, but the differences between the models trained on UD and SUD were small enought to continue the study.

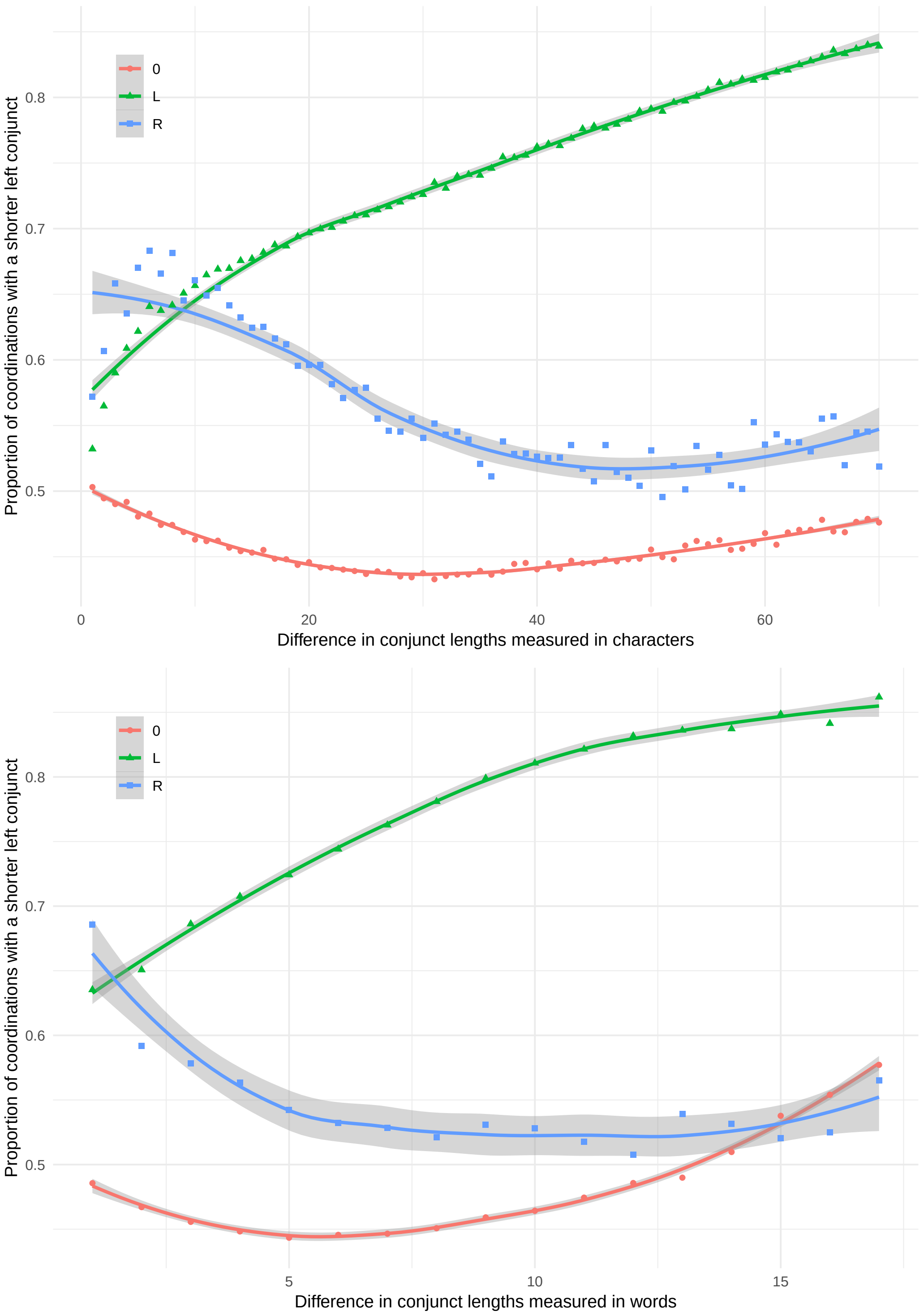


Figure 4.1: Observed and loess-smoothed proportions of coordinations with shorter left conjuncts depending on the length difference between the conjuncts and on the position of the governor

4.2 Governor's impact on the coordination

The second hypothesis tested here is that placement of the shorter conjunct in a coordination is affected by the governor position and the length difference between the conjuncts. Figure 4.1 shows the observed proportions of coordinations with shorter left conjuncts depending on the length difference between the conjuncts, grouped by the possible positions of the governor. Regardless of the measure, the proportion of coordinations with shorter left conjuncts increases steadily when the governor is on the left. When there is no governor the proportion decrases at first and starts to grow around length differences equal to 30 characters or 7 words. Similarly with the governor on the right, the proportions decrease initially and rise slightly with bigger length differences, but the changes happen at different rates between the two presentes measures.

Figure 4.2 shows the results of fitting logistic regression models to the observations presented in Figure 4.1. Here the slopes are more consistent between the utilised measures. When the governor is on the left, the slopes are positive, when there is no governor they are slightly negative and when the governor is on the right they are also negative, but this time much steeper.

The Hosmer-Lemeshow test (Hosmer et al., 2013) was used to test the goodness of fit of the models presented in Figure 4.2. The null hypothesis of the test is that the observed and predicted proportions are the same. The test was performed using R's `ResourceSelection::hoslem.test`. In the models fitted to data gathered here, the proportion of coordinations with shorter left conjuncts is predicted based on the length difference between the two conjuncts. Those proportions are arranged in an ascending order and grouped into 10 percentile groups. Then the Hosmer-Lemeshow statistic is calculated using the formula in (4.1).

$$H = \sum_{g=1}^{G} \left(\frac{(O_{1g} - E_{1g})^2}{E_{1g}} + \frac{(O_{0g} - E_{0g})^2}{E_{0g}} \right) \tag{4.1}$$

The results of the tests run on all of the models are presented in (4.2).

model	X-squared	degrees of freedom	p-value
L-char	9762.6	8	< 2.2e-16
R-char	2781.5	8	< 2.2e-16
0-char	4926.6	8	< 2.2e-16
L-words	460.9	4	< 2.2e-16
R-words	3490.8	2	< 2.2e-16
0-words	4142.6	5	< 2.2e-16

Table 4.2: Results of the Hosmer-Lemeshow test conducted on the logistic regression models.

[1] Evaluation was performed using the `conll18_ud_eval.py` script from the study conducted by Tuora et al. (2021), found in the related repository: https://github.com/ryszardtuora/ud_vs_sud.

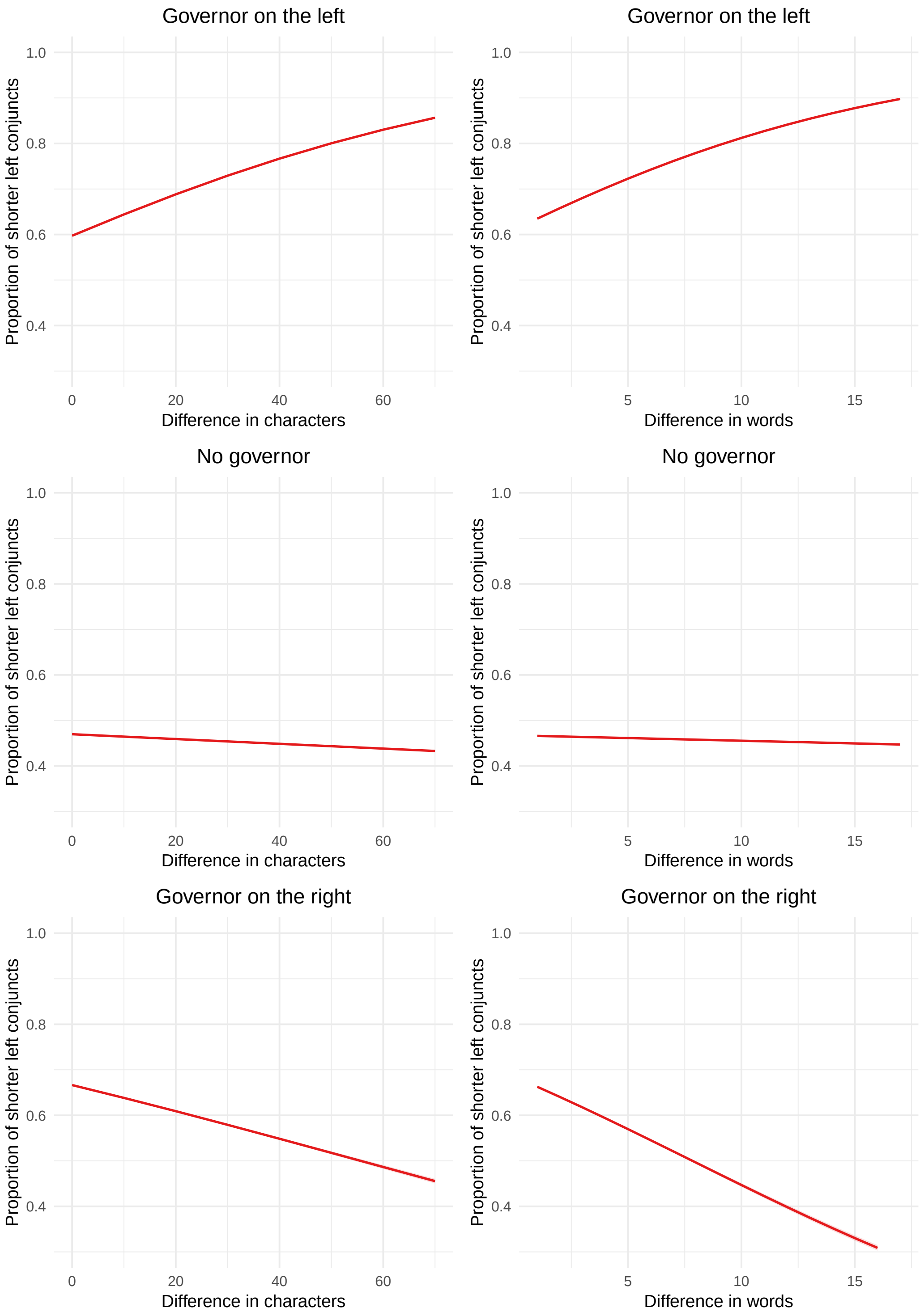
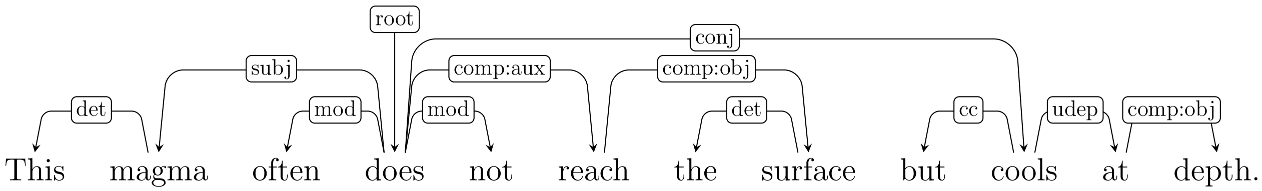


Figure 4.2: Modelled proportions of coordinations with shorter left conjunct depending on the length difference between the conjuncts

Appendix A

Example of a CoNLL-U representation


root
conj
subj
comp:aux
comp:obj
det
mod
mod
det
cc
udep
comp:obj
This magma often does not reach the surface but cools at depth.

```
# text = This magma often does not reach the surface but cools at depth.
 1   This     this     DET    DT   Number=Sing|PronType=Dem                                   2   det       _   start_char=0|end_char=4
 2   magma    magma    NOUN   NN   Number=Sing|Shared=No                                      4   subj      _   start_char=5|end_char=10
 3   often    often    ADV    RB   _                                                          4   mod       _   start_char=11|end_char=16
 4   does     do       AUX    VBZ  Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin      0   root      _   start_char=17|end_char=21
 5   not      not      PART   RB   _                                                          4   mod       _   start_char=22|end_char=25
 6   reach    reach    VERB   VB   VerbForm=Inf                                               4   comp:aux  _   start_char=26|end_char=31
 7   the      the      DET    DT   Definite=Def|PronType=Art                                  8   det       _   start_char=32|end_char=35
 8   surface  surface  NOUN   NN   Number=Sing                                                6   comp:obj  _   start_char=36|end_char=43
 9   but      but      CCONJ  CC   _                                                          10  cc        _   start_char=44|end_char=47
 10  cools    cool     VERB   VBZ  Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin      4   conj      _   start_char=48|end_char=53
 11  at       at       ADP    IN   Shared=No                                                  10  udep      _   start_char=54|end_char=56
 12  depth    depth    NOUN   NN   Number=Sing                                                11  comp:obj  _   start_char=57|end_char=62
 13  .        .        PUNCT  .    _                                                          4   punct     _   start_char=62|end_char=63
```

Figure A.1: conllu

Bibliography

Behaghel, O. (1930). Zur Wortstellung des Deutschen. *Language*, 6(4):29–33.

Buchholz, S. and Marsi, E. (2006). CoNLL-X shared task on multilingual dependency parsing. In Màrquez, L. and Klein, D., editors, *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164, New York City. Association for Computational Linguistics.

de Marneffe, M.-C., MacCartney, B., and Manning, C. D. (2006). Generating typed dependency parses from phrase structure parses. In Calzolari, N., Choukri, K., Gangemi, A., Maegaard, B., Mariani, J., Odijk, J., and Tapias, D., editors, *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy. European Language Resources Association (ELRA).

Dyer, A. (2023). Revisiting dependency length and intervener complexity minimisation on a parallel corpus in 35 languages. In *Proceedings of the 5th Workshop on Research in Computational Linguistic Typology and Multilingual NLP*, pages 110–119.

Eisner, J. and Smith, N. A. (2005). Parsing with soft and hard constraints on dependency length. In Bunt, H. and Malouf, R., editors, *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 30–41, Vancouver, British Columbia. Association for Computational Linguistics.

Gerdes, K., Guillaume, B., Kahane, S., and Perrier, G. (2018). SUD or Surface-Syntactic Universal Dependencies: An annotation scheme near-isomorphic to UD. In de Marneffe, M.-C., Lynn, T., and Schuster, S., editors, *Proceedings of the Second Workshop on Universal Dependencies (UDW 2018)*, pages 66–74, Brussels, Belgium. Association for Computational Linguistics.

Gibson, E. (1998). Linguistic complexity: locality of syntactic dependencies. *Cognition*, 68(1):1–76.

Gildea, D. and Temperley, D. (2007). Optimizing grammars for minimum dependency length. In Zaenen, A. and van den Bosch, A., editors, *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 184–191, Prague, Czech Republic. Association for Computational Linguistics.

Gildea, D. and Temperley, D. (2010). Do grammars minimize dependency length? *Cognitive Science*, 34(2):286–310.

Hawkins, J. A. (1994). *A Performance Theory of Order and Constituency*, volume 73 of *Cambridge Studies in Linguistics*. Cambridge University Press, Cambridge.

Hosmer, D. W. J., Lemeshow, S., and Sturdivant, R. X. (2013). Assessing the fit of the model. In *Applied Logistic Regression*, chapter 5, pages 153–225. John Wiley & Sons, Ltd.

Hudson, R. (1984). *Word Grammar*. Blackwell, Oxford.

Hudson, R. (2010). *An Introduction to Word Grammar*. Cambridge Textbooks in Linguistics. Cambridge University Press.

Hunter, P. J. and Prideaux, G. D. (1983). Empirical constraints on the verb-particle construction in English. *Journal of the Atlantic Provinces Linguistic Association*.

King, J. and Just, M. A. (1991). Individual differences in syntactic processing: The role of working memory. *Journal of Memory and Language*, 30(5):580–602.

Kohita, R., Noji, H., and Matsumoto, Y. (2017). Multilingual back-and-forth conversion between content and function head for easy dependency parsing. In Lapata, M., Blunsom, P., and Koller, A., editors, *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 1–7, Valencia, Spain. Association for Computational Linguistics.

Mel'čuk, I. (1988). *Dependency Syntax: Theory and Practice*. The SUNY Press, Albany, NY.

Nguyen, M. V., Lai, V., Veyseh, A. P. B., and Nguyen, T. H. (2021). Trankit: A light-weight transformer-based toolkit for multilingual natural language processing. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*.

Nilsson, J., Nivre, J., and Hall, J. (2006). Graph transformations in data-driven dependency parsing. In Calzolari, N., Cardie, C., and Isabelle, P., editors, *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 257–264, Sydney, Australia. Association for Computational Linguistics.

Nivre, J., de Marneffe, M.-C., Ginter, F., Goldberg, Y., Hajič, 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., Choukri, K., Declerck, T., Goggi, S., Grobelnik, M., Maegaard, B., Mariani, J., Mazo, H., Moreno, A., Odijk, J., and Piperidis, S., editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666, Portorož, Slovenia. European Language Resources Association (ELRA).

Popel, M., Mareček, D., Štěpánek, J., Zeman, D., and Žabokrtskỳ, Z. (2013). Coordination structures in dependency treebanks. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 517–527.

Przepiórkowski, A., Borysiak, M., and Głowacki, A. (2024). An argument for symmetric coordination from Dependency Length Minimization: A replication study. To appear in the proceedings of *LREC-COLING 2024*.

Przepiórkowski, A. and Woźniak, M. (2023). Conjunct lengths in English, Dependency Length Minimization, and dependency structure of coordination. In Rogers, A., Boyd-Graber, J., and Okazaki, N., editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15494–15512, Toronto, Canada. Association for Computational Linguistics.

Qi, P., Zhang, Y., Zhang, Y., Bolton, J., and Manning, C. D. (2020). Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Rehbein, I., Steen, J., Do, B.-N., and Frank, A. (2017). Universal Dependencies are hard to parse – or are they? In Montemagni, S. and Nivre, J., editors, *Proceedings of the Fourth International Conference on Dependency Linguistics (Depling 2017)*, pages 218–228, Pisa, Italy. Linköping University Electronic Press.

Tesnière, L. (1959). *Elements of Structural Syntax*. Klincksieck, Paris.

Tesnière, L. (2015). *Elements of Structural Syntax*. John Benjamins, Amsterdam.

Tuora, R., Przepiórkowski, A., and Leczkowski, A. (2021). Comparing learnability of two dependency schemes: 'semantic' (UD) and 'syntactic' (SUD). In *Findings of the Association for Computational Linguistics: EMNLP 2021*. Association for Computational Linguistics.

Wasow, T. (2002). *Postverbal Behavior*. CSLI Publications, Stanford.

Zeldes, A. (2017). The GUM corpus: Creating multilayer resources in the classroom. *Language Resources and Evaluation*, 51(3):581–612.

Zeman, D., Popel, M., Straka, M., Hajic, J., Nivre, J., Ginter, F., Luotolahti, J., Pyysalo, S., Petrov, S., Potthast, M., Tyers, F., Badmaeva, E., Gokirmak, M., Nedoluzhko, A., Cinkova, S., Hajic jr., J., Hlavacova, J., Kettnerová, V., Uresova, Z., Kanerva, J., Ojala, S., Missilä, A., Manning, C. D., Schuster, S., Reddy, S., Taji, D., Habash, N., Leung, H., de Marneffe, M.-C., Sanguinetti, M., Simi, M., Kanayama, H., dePaiva, V., Droganova, K., Martínez Alonso, H., Çöltekin, c., Sulubacak, U., Uszkoreit, H., Macketanz, V., Burchardt, A., Harris, K., Marheinecke, K., Rehm, G., Kayadelen, T., Attia, M., Elkahky, A., Yu, Z., Pitler, E., Lertpradit, S., Mandl, M., Kirchner, J., Alcalde, H. F., Strnadová, J., Banerjee, E., Manurung, R., Stella, A., Shimada, A., Kwak, S., Mendonca, G., Lando, T., Nitisaroj, R., and Li, J. (2017). Conll 2017 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19, Vancouver, Canada. Association for Computational Linguistics.