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

1 Introduction 4

2 Theoretical background 6

2.1 Dependency grammars 6

2.2 Dependency length minimization 7

2.3 Possible dependency structures of a coordination 8

2.3.1 Bouquet/Stanford 9

2.3.2 Chain/Moscow 10

2.3.3 Multi-headed/London 10

2.3.4 Conjunction-headed/Prague 11

2.4 Previous studies 12

2.5 Universal Dependencies 14

2.6 Surface-syntactic Universal Dependencies 15

2.6.1 Criteria for choosing heads of dependencies 15

2.6.2 Explicit information about shared dependencies 18

2.6.3 Learnability of dependency schemes 19

3 Data processing 20

3.1 Parser training 20

3.1.1 Combined model 22

3.1.2 Spoken model 23

3.1.3 Comparison models 23

3.2 Data extraction 24

4 Statistical analysis 27

4.1 UD and SUD model comparison 27

4.2 Governor's impact on the coordination 27

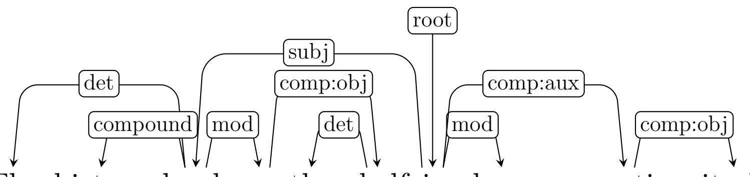
Chapter 1

Introduction

Coordinations are structures which consist of multiple elements, such that neither of the elements governs any of the others. Those elements are conjuncts and they can be connected with punctuation and with conjunctions such as *and*, *or*, *as well as* and others. An example of a coordination is provided in (1), with a conjunction *and* and conjuncts: *some apples*, *the oranges your mother gave you*.

(1) Bring [[some apples] and [the oranges your mother gave you]]

This work aims to investigate how coordinations should be interpreted in certain syntactic theories, specifically in dependency grammars. In dependency grammars the structure of a sentence is represented by a dependency tree, in which almost every word depends on another one – that word is its governor. In (2) there is an example of a sentence with a dependency annotation. In this sentence the shorter conjunct is placed as the left one, which was observed to be common in the English language.



(2) The history book on the shelf is always repeating itself

Przepiórkowski and Woźniak (2023) have conducted a study in which they provided an argument for certain approaches to annotating coordinations in dependency grammars. To do so, they checked whether coordinate structures were formed differently depending on the position of the governor of the structure. They found that in coordinations with a governor on the left (as in (3) with the governor *bring*) it is more likely that the shorter conjunct will be also placed on the left. This means that people are more likely to form sentence (3a) rather than sentence (3b). Similarly, if the coordination has no governor (as in (4)), the shorter conjunct is more likely to be on the left (as in (4a)) than on the right (as in (4b)).

(3) a. *Bring* [[some apples] and [the oranges your mother gave you]]
b. *Bring* [[the oranges your mother gave you] and [some apples]]

(4) a. [[Buy some apples] or [steal as many oranges as you can hold]]

b. [[Steal as many oranges as you can hold] or [buy some apples]]

However in sentences with a governor on the right, the shorter conjunct is more likely to appear as the right one, which means that sentence (5b) is more likely to appear in the English language than sentence (5a).

(5) a. [[An apple] and [three long orange peels]] *fell* out of the bag
b. [[Three long orange peels] and [an apple]] *fell* out of the bag

Those findings were based on the Penn Treebank, which is a corpus of texts collected from the Wall Street Journal with manually added syntactic annotation. One advantage of such a resource is high quality, reliable annotation, but since creating manual annotation requires time, the corpus is small. Additionally, the corpus contains only texts from the Wall Street Journal, therefore it is not stylistically diverse.

Another paper by Przepiórkowski et al. (2024) describes an attempt at replicating the results found in Przepiórkowski and Woźniak (2023) on a larger and more diverse corpus, namely the Corpus of Contemporary American English. Downside to using this corpus is that there is no syntactic annotation, therefore it had to be added automatically using a parser. This unfortunately lowered the quality of data, compared to the manually annotated Penn Treebank. After conducting an evaluation, Przepiórkowski et al. (2024) found only around half of the sampled coordinate structures to be correctly extracted from the text.

The dependency annotation in Przepiórkowski et al. (2024) was made according to a scheme called Universal Dependencies. Tuora et al. (2021) show that some automatic parsers can perform better when using another scheme called Surface-syntactic Universal Dependencies. Additionally, the structural differences between those two schemes allow for more precise heuristics for finding coordinations in sentences parsed according to the surface-syntactic scheme.

The main goal of the current study is to replicate the aforementioned studies – this provides the first two hypotheses that are tested here: 1) placement of the shorter conjunct within a coordination depends on the position of the governor and 2) the tendency for the shorter conjunct to appear closer to the governor is more prominent with bigger length differences between conjuncts. The third hypothesis is a result of an effort to increase the quality of the automatically annotated and extacted data. It states that the parsing accuracy is higher when text is parsed according to Surface-syntactic Universal Dependencies, rather than the original Universal Dependencies scheme.

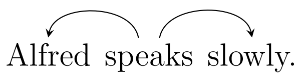
The structure of this thesis is the following: Chapter 2 explains the necessary theoretical background. Chapter 3 describes how the corpus data was prepared for analysis – from training the parser to extracting the information about coordinate structures. Chapter 4 presents the analysis of the data, which is discussed in Chapter 5. Finally Chapter 6 covers the limitations of this research.

Chapter 2

Theoretical background

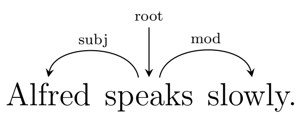
2.1 Dependency grammars

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.

(6)


Tesnière as an example used the sentence *Alfred speaks slowly*, for which a dependency tree is shown in (6). It visualises the rules described above – all of the words in a sentence are connected, those connections are directed and the verb is central 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, that complete its meaning, and optional dependencies, that 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 (6) could look similarly to (7), 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.

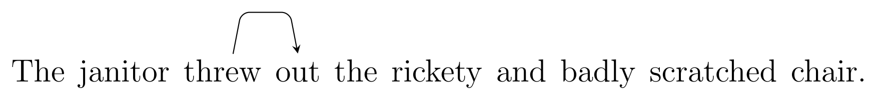
(7)


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.

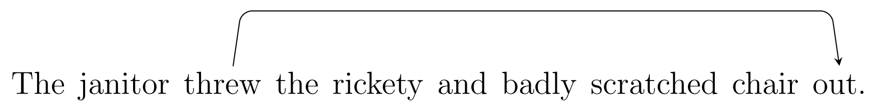
The last two sections of this chapter delve into more detail about two projects concerned with creating consistent dependency annotation schemes – Universal Dependencies in Section 2.5 and Surface-syntactic Universal Dependencies in 2.6.

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 (8), illustrates this.



(8) 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 (8a) more acceptable than the ones similar to (8b).[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 (Gibson, 1998; King and Just, 1991) and those based on corpus research (Dyer, 2023; Gildea and Temperley, 2007, 2010).

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 (8) – when there are multiple grammatical word orders available, people choose those with shortest dependencies, because it makes the sentence easier to understand. As for DLM in grammar, an example taken from Hawkins (1994) is in (9) (p. 20).

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

Both of the sentences in (9) have a constituent embedded inside of them. In (9a) that constituent is a subordinate clause *that John failed his exam*, whereas in (9b) 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 its 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 (9b) 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.

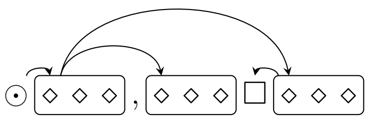
Therefore a coordination presented in (**??**) using this set of symbols would look like in (10).

(10)	in	this	and	so many things
	⊙	$\boxed{\diamond}$	□	$\boxed{\diamond \diamond \diamond}$

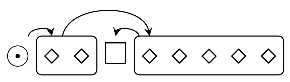
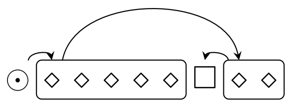
In all of the diagrams in the current section it is assumed that the head of the conjunct is its first word. This assumption is justified, because the work presented here is based solely on the English language, which is mostly head-initial, therefore the diagrams presented here are more likely to be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

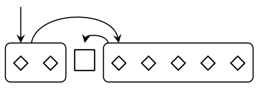
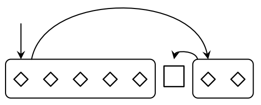
2.3.1 Bouquet/Stanford

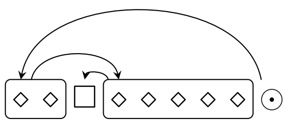
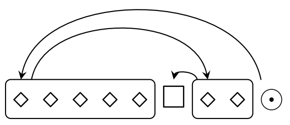
The bouquet structure comes from the Stanford parser (de Marneffe et al., 2006). Coordination with three conjuncts and a governor on the left would be annotated in the bouquet approach as shown in (11).

(11)


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.

(12) a.

b.


c.

d.


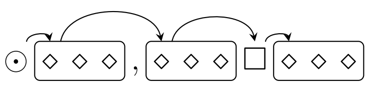
e.

f.


The diagrams in (12) show the bouquet structure with different governor positions and conjunct placements. In (12a–b) the governor is on the left with the shorter conjunct on the left in (12a) and on the right in (12b). Two of the dependencies drawn have the same lenght in both cases, but one of them is visibly longer in (12b). This means that, according to the DLM principle, the structure in (12a) should be preferred, so coordinations with conjuncts of visibly different lengths should have the shorter conjunct on the left.

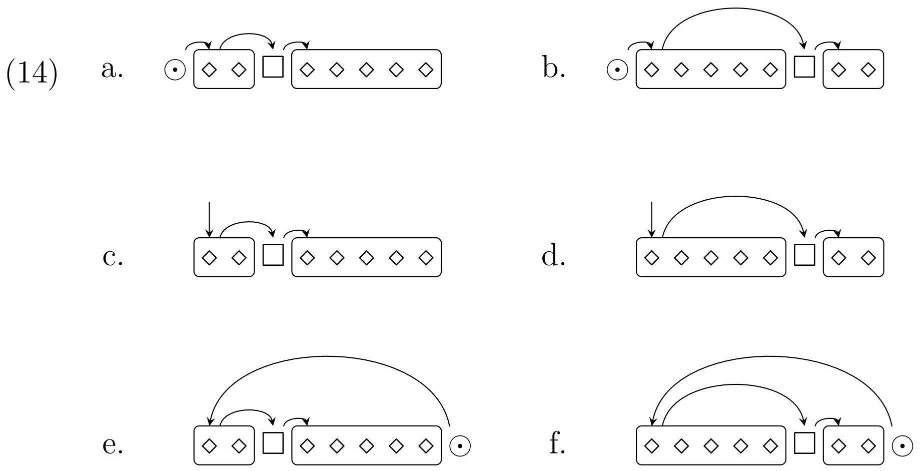
Within the bouquet approach this is the case for every governor position. Diagrams (12c–d) show the structure of coordination when the governor is absent, with dependencies shorter in (12c) than in (12d), so when the shorter conjunct is on the left. Diagrams (12e–f) show the structure when the governor is on the right, with dependencies shorter in (12e) than in (12f), 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).

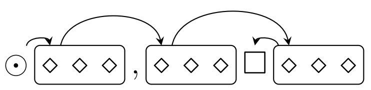
(13)


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.

(14)


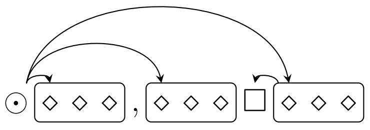
The diagrams in (14) show that, in this case, similarly to the bouquet approach, the placement of the shorter conjunct on the left should be preferred, assuming the DLM principle. The placement of the governor again does not seem to have an effect on the ordering.

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 (13) is that the conjunction is not attached to the preceding conjunct, but to the following one. This annotation is illustrated in (15).

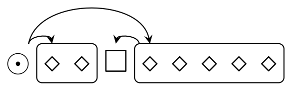
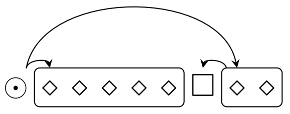
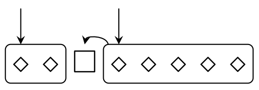
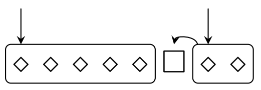
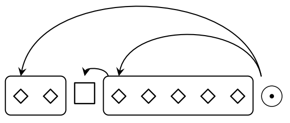
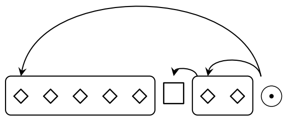
(15)


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.

(16)


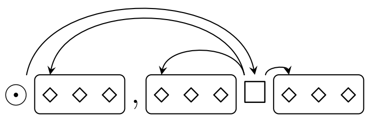
The symmetry of the approach comes from the fact that no conjunct is distinguished by being the only direct dependent of coordinations governor. Instead, all of the conjuncts have dependencies connecting them to the governor, and the conjunction is dependent on the last conjunct, similarly to the bouquet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

(17)
a.

b.

c.

d.

e.

f.


Here, the predictions for the preferred ordering of conjuncts are different to those in asymmetrical approaches. This annotation suggests that 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 (17a–b), the dependencies are shorter with the shorter conjunct placed on the left, and with the governor on the right, as in (17e–f), the shorter conjunct should be placed on the right. In the case of no word governing the coordination, as in (17c–d), there seems to be no preference for any ordering.

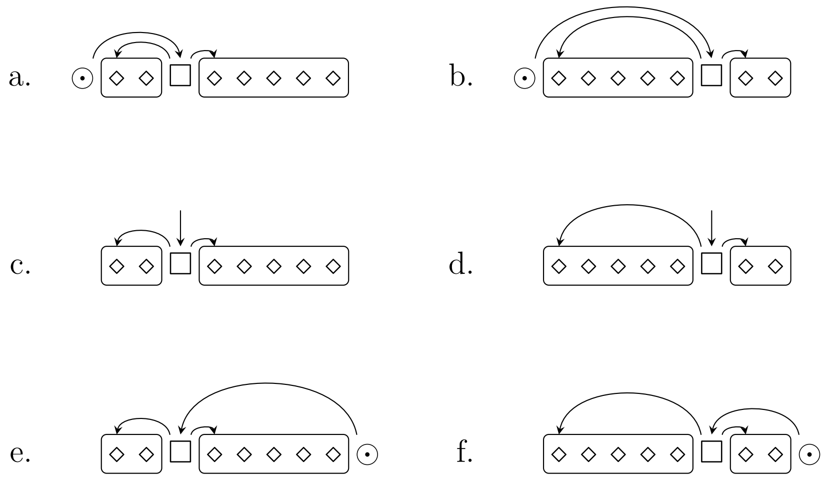
2.3.4 Conjunction-headed/Prague

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).

(18)


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.

(19)



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.

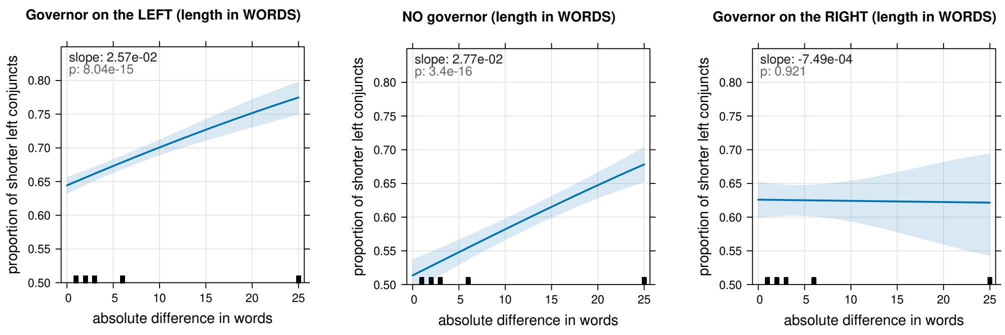


Figure 2.1: Modelled proportions of coordinations with left conjuncts shorter depending on difference in conjunct lengths from Przepiórkowski and Woźniak (2023)

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 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.

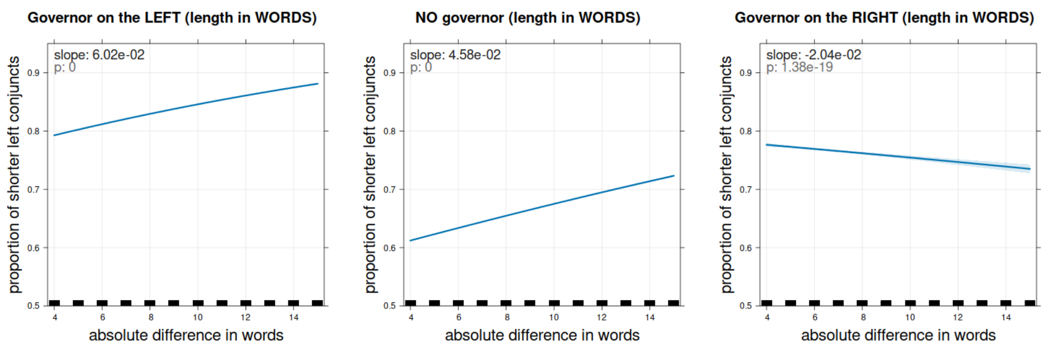


Figure 2.2: Modelled proportions of coordinations with left conjuncts shorter depending on difference in conjunct lengths from Przepiórkowski et al. (2024)

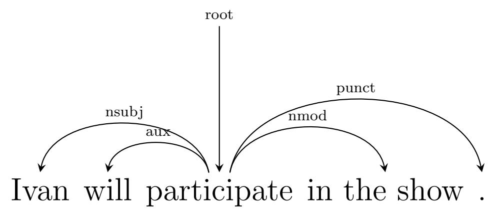
Those results point to only one of the annotation styles, namely the London one. Within this style, dependencies are minimised when the shorter conjunct is placed closer to the governor if there is one. If the coordination has no governor, neither placement should be preferred, unless DLM at the level 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.

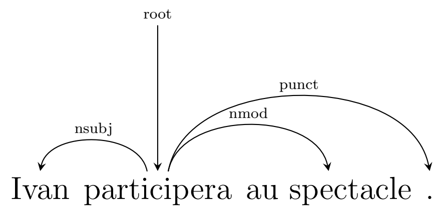
2.5 Universal Dependencies

As seen in Section 2.3, there have been many ideas on how to create dependency annotation. 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.

Rules for creating a dependency structure are the most relevant part here. In the first version of UD (Nivre et al., 2016), three of them are specified: dependency relations appear between content words, function words are attached to the content words which they describe and 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 the sentence (20), 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.

(20)


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. (21) is a tree for the French translation of the sentence in (20). 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.

(21)


2.6 Surface-syntactic Universal Dependencies

Surface-syntactic Universal Dependencies (SUD, henceforth) is another example of an attempt at creating a set of universal guidelines for dependency annotation. Gerdes et al. (2018) describe it as "near-isomorphic to UD" and offer 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 (22) and (23) show why the head of this dependency is the word *boy*. In (22a) the word *boy* can be replaced by the unit *little boy*, as shown in (22b), and the sentence is still grammatical.

(22) 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 (23a) results in an ungrammatical sentence in (23b).

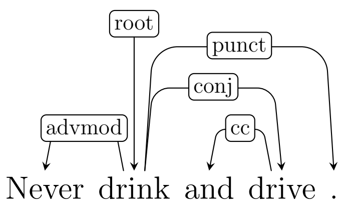
Sentences (23c–d) show that replacement in the other way is not possible either, therefore the word *little* cannot be the head of this dependency.

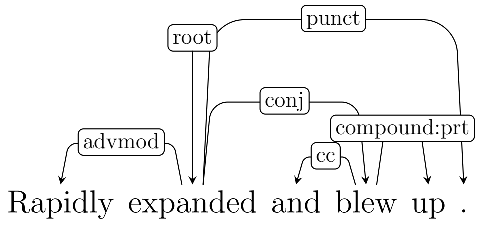
(23) a. The boy was **little**.
b. * The boy was **little boy**.
c. I found the **little boy**.
d. * I found the **little**.

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 (24) show that *Mary* does not have the same distribution as those two words together.

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

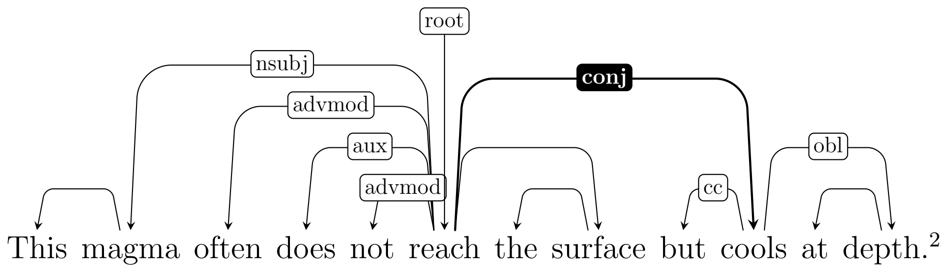
This key difference between UD and SUD has crucial consequences for extracting the exact length of conjuncts in a coordinate structure, which is an integral part of this work. As Przepiórkowski and Woźniak (2023) mention, the UD scheme is not ideal for coordination analysis, as it is not clear which dependencies are shared by the conjuncts and which are private. This can be illustrated by the sentences in (25) and (26).

(25) 

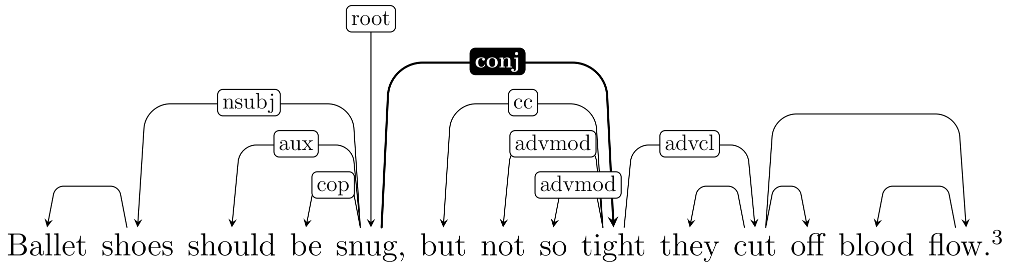
(26) 

In (25) 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 (26) 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 (25)) or private to the first conjunct (as in 26). 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 (27) 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.



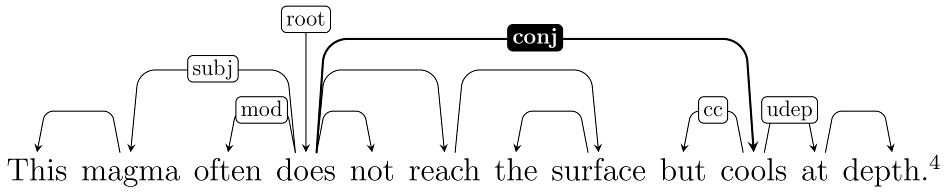
(27) This magma often does not reach the surface but cools at depth.[2]



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

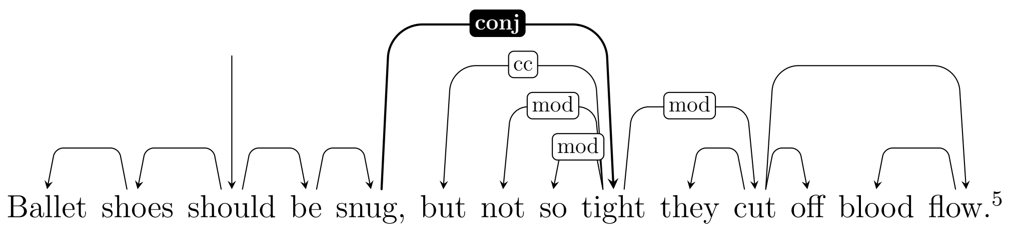
Modifying the heuristics, so that they fit this example, for instance by saying that `aux` dependencies should always be included in the left conjunct, would on the other hand mean that sentences such as in (28) would have incorrect extracted coordinations. In this example the correct coordinate structure has conjuncts *snug* and *not so tight they cut off blood flow*, but was the algorithm to include the `aux` dependency in the first conjunct (as would be required in (27)), 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 (29), 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 (28) – the word *snug* has to be the whole left conjunct, because it also does not have any dependencies in this annotation scheme.



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

[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`.



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

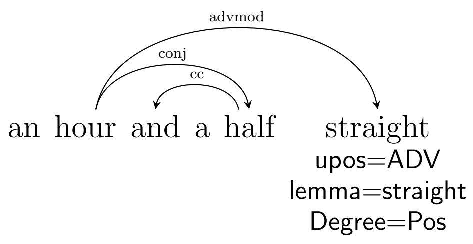
Therefore, the focus on syntax in SUD makes it a better fit for coordination analysis.

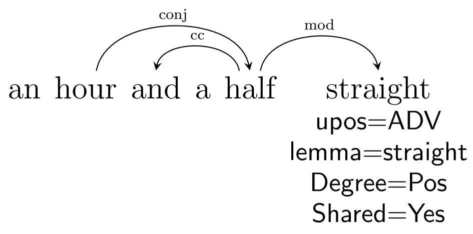
2.6.2 Explicit information about shared dependencies

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 specifially 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 UD annotation in (31) shows, the word *straight* is shared by the whole structure. This is not structurally visible in the SUD version in (32), where the `mod` dependency for the word *straight* is attached to the last conjunct.

(31)


(32)


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.

In the Stanza pipeline the processor responsible for assigning features to every token is the part-of-speech tagger. It was trained for the current study alongside the dependency parser to have the information about shared dependencies available in the data.

[5]Sentence `GUM_whow_ballet-14` from the `SUD_English-GUM` corpus.
[6]Sentence `GUM_vlog_studying-27` from the GUM corpus

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.

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 model 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 was chosen for this task. The default parsing model provided for English annotates in the Universal Dependencies scheme, but as was explained earlier, the Surface-syntactic Universal Dependencies scheme is preferred here. The parser therefore had to be trained to annotate in the SUD scheme.

The current chapter has two sections: the first one describes the process of training the parsing models that created the syntactic annotation. The second one describes the procedure of finding coordinate structures in parsed sentences and creating a table with data ready for analysis.

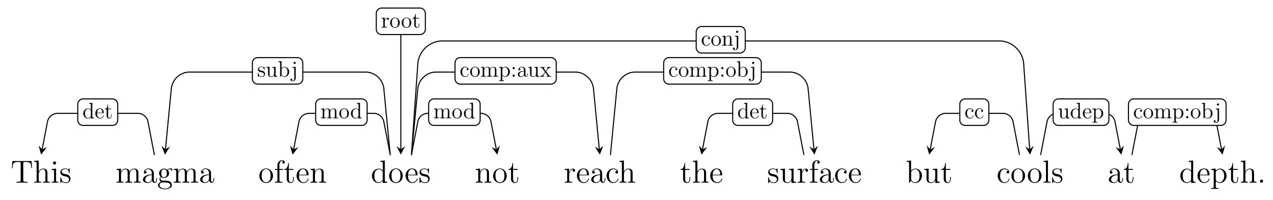
3.1 Parser training

Training was conducted using scritps made available on github by the Stanza developers.[1] For the parser to learn annotation, there needs to be already annotated data. The Surface-syntactic Universal Dependencies is a much smaller project than Universal Dependencies, therefore there are not many corpora annotated natively in this scheme. Because of that, a set of graph conversion rules was developed.[2] Using this conversion code, almost all of the UD corpora have been translated to SUD.

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. Hence Buchholz and Marsi (2006) created the CoNLL-X format. The exact purpose was to compare parser outputs in a dependency parsing shared task. Today it 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. In (33) there is an example of a SUD dependency tree presented as a tree and in the CoNLL-U format.

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

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

(33)

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
```

Corpora in the CoNLL-U format were downloaded from the SUD website. 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 parser for a certain number of learning steps (in the scripts used here this is 3000 steps).

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 5000 and the dropout rate was 0.33. This means that the whole training dataset was split into batches, each with 5000 elements – the bigger the batch the more precise the gradient function and thus the predictions. 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.

All of the corpora used in this experiment were ready for training, besides one – the GUMReddit corpus, as it is not fully available online. All of the dependencies and morphological features and so on are included in the CoNLL-U files, but the words themselves and the lemmas are missing. As the textual data is required for training, the appropriate script from the GUM corpus repository was utilised.[3]

The following subsections describe the models that were trained as a part of this study.

[3]https://github.com/amir-zeldes/gum/blob/master/get_text.py

3.1.1 Combined model

The most crucial of the models was the combined one. It was used to annotate most of the data analysed in this study. The corpora used for training were SUD_English-EWT, SUD_English-GUM, SUD_English-ParTUT.

The first of these is based on the English Web Treebank. It has 5 primary sources of data: weblogs, newsgroups, emails, reviews and question-answers. The first two – weblogs and newsgroups – were collected in the years 2003–2006. Email messages are from the Enronsent Corpus, which contains emails sent by the employees of Enron Corporation in the years 1999–2002, that were made public domain during the investigation of Enron. The emails in the corpus were processed to have as little non-human input as possible. The reviews of businesses and services included in the corpus were taken from various Google websites in 2011, but no specific dates were provided. The case was similar with question-answer data, which was collected in 2011 from the Yahoo! Answers website. In total, the corpus contains 254,820 words. The original constituency annotation was automatically converted into Stanford Dependencies and then manually corrected to UD. Using the code mentioned earlier in this chapter it was then converted to SUD. The corpus was chosen for training this model, as it is the biggest available annotated corpus for English.

The second corpus used for this model was SUD_English-GUM, which is a SUD-converted version of the Georgetown University Multilayer corpus. While the early versions of GUM were annotated in Stanford Dependencies scheme, they were later manually converted into UD and the later versions have been created natively in UD. The corpus comprises a variety of styles, for instance academic, interviews, travel guides, letters, how-to guides and forum discussions. The last of these requires special attention, since it is sometimes considered seperately from the rest of the GUM corpus. The source for this data are Reddit forum discussions, which means that the text within this part of the corpus is protected by the Reddit terms and conditions and thus not freely available. In this data, words are substituted by underscores in the form and lemma fields in CoNLL-U files. Using it for training requires first running a script that completed the corpus, as was mentioned eariler. Including the Reddit data, there are 228,399 tokens in the corpus, which makes it the second largest corpus among the ones available for English and an obvious choice for training a parser.

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 annotated in a style specific to the treebanks developed at the University of Turin, then converted to UD and from that to SUD. 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, especially considering the style difference between this corpus and the other ones.

One additional reason for choosing the corpora listed above is 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 inconsistent with the other corpora used here.

For English, the Stanza parser by default also uses a model trained on multiple corpora: EWT, GUM, PUD and Pronouns. The two last ones were not used to train the model here, as they were too small to have been split into the training, development and testing sets.

3.1.2 Spoken model

Spoken and written language differ significantly, which can lead to poor automatic parsing performance. To avoid that, this experiment involves training a model specialising in spoken data. The corpora used to this end were SUD_English-Atis and parts of the SUD_English-GUM corpus.

Parts of the GUM corpus utilised here were those with interviews, conversations and vlogs. The relevant sentences were extracted from the original CoNLL-U files by looking for documents labelled `id = GUM_interview`, `id = GUM_vlog` or `id = GUM_conversation`.

As for the other corpus, Atis comprises sentences from the Airline Travel Informations dataset. Those are transcriptions of people asking automated inquiry systems for flight information. The corpus has 61,879 tokens and was natively annotated in UD.

3.1.3 Comparison models

The main purpose of the combined model is to create annotation for analysis, though one of the intentions of this study was also to compare the learnability of the two dependency annotation schemes: the "semantic" UD and the "syntactic" SUD. However a direct comparison between the two combined models is not be possible for a lack of information about the UD combined model. The datasets included in the training are described in the Stanza documentation,[4] however there is no information on how the datasets were split into training and testing sets. The splits are known for big corpora, such as GUM and EWT, but the PUD and Pronouns are too small to be split into smaller sets. Another issue is that the evaluation for the UD combined model was not made available and running an evaluation script on this model is not possible without the training, development and testing splits.

Considering this, the learnability of the annotation schemes described here is compared based on the models that are trained on one corpus each. All of those models have been trained on corpora big enough to have official splits into training, development and testing sets. Evaluation of those models trained on UD is reported in the Stanza documentation and was conducted on the testing sets. The comparison SUD models were trained using the official splits for every corpus. Results of the evaluation of this training along with the evaluation results for the corresponding UD models are shown in Section (there will be a ref here).

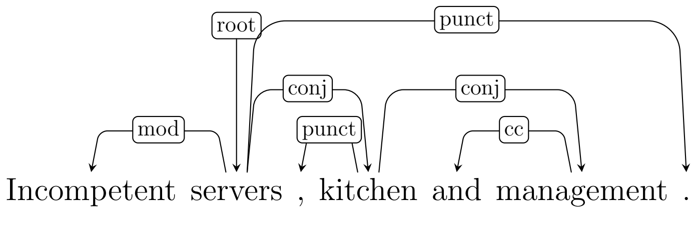
[4]https://stanfordnlp.github.io/stanza/combined_models.html

3.2 Data extraction

The input data from the COCA corpus was in the form of text files. Those were split into sentences using another parser, Trankit (Nguyen et al., 2021), as the sentence segmentation outputs from Trankit and Stanza suggested that this would provide better quality data for further analysis. Split texts were then put into .tsv files, so that along with the sentence text, also the information about text identifier and the index of the sentence in a document could be included. Those sentences were then put into batches, but separated with `\n\n`. This way the parsing process was faster than it would be when parsing sentence by sentence, but the sentences were clearly already split. The next step was parsing the texts, which was done using a parsing pipeline. Creating a parsing pipeline requires specifying the input language and the processors that make up the pipeline – in this case those were the tokeniser, part-of-speech-tagger, lemmatiser and dependency parser. Here the tokeniser and lemmatiser were those of the default English model, as those two processes are the same in UD and SUD. The other two processors, POS-tagger and dependency parser were trained on SUD and used in the pipeline. Additionally, it was specified that the text was already split into sentences by setting the `tokenize_no_split` parameter to `True`. The pipeline configuration is shown in (34).

(34)
```
config = {
    'processors': 'tokenize,pos,lemma,depparse',
    'lang': 'en',
    'use_gpu': True,
    'pos_model_path':
'./saved_models/en_combined-sud_charlm_tagger.pt',
    'depparse_model_path':
'./saved_models/en_combined-sud_charlm_parser.pt',
    'tokenize_pretokenized': False,
    'tokenize_no_ssplit': True,
    'download_method': stanza.DownloadMethod.REUSE_RESOURCES
    }
nlp = stanza.Pipeline(**config)
```

The process of extracting coordinations will be illustrated by the sentence shown in (35) with a SUD tree.[5]

(35)


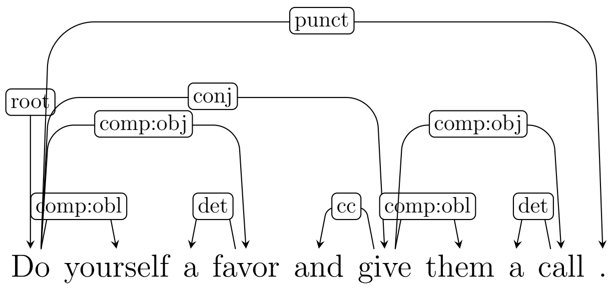
The text is then given to the `nlp` pipeline and parsed. In the output there is a list of Sentence objects, which have a list of dependencies as one of their attributes.

Coordinations are found by looking for `conj` dependencies within a sentence. The SUD approach assumes the chain representation of a coordination,

[5]Sentence `reviews-357217-0002` from the EWT corpus (Silveira et al., 2014).

therefore once there a `conj` dependency is found, the script looks for more links in the chain. In the sentence (35) first the `conj` dependency will be found between the words *servers* and *kitchen*. Indices of those two words are saved in a list and the script looks through the dependencies attached to the word *kitchen*: there is `punct` and `conj`. The `conj` one is attached to the word *management*, therefore the same is then done for that word: its index is saved to the list and the script checks whether it has a `conj` dependency. Since it does not have such a dependency, the whole coordination has been found – it has three conjuncts: *servers*, *kitchen* and *management*.

Once all lists of indices representing coordinations in a given sentence are ready, for each one of them a dictionary is created. An example of such a dictionary will be presented for the coordination in the sentence in (36)[6].



(36)

The first and last indices are popped from the list and added to the dictionary as the left and right conjunct heads respectively with the keys `'L'` and `'R'`. The rest of the conjuncts are added with the key `'other_conjuncts'` – for the coordination in sentence (36) that list is empty, as there are only two conjuncts. This coordination does not have a governor, therefore the key `'gov'` is not added to the dictionary. The next step is looking for the conjunction of the coordination. If there is one, it is always attached to the right conjunct and it is labelled `cc`. Once it is found, the whole word is added to the dictionary with the `'conj'` key.

Finally, the information about the conjunct lengths has to be extracted. All of the dependencies appearing between the left and right conjunct heads are considered private to the conjuncts they are attached to. This means that in sentence (36) the case of the left conjunct is simple – both of the dependencies of the word *Do* (`comp:obj` and `comp:obl`) are directed to the right and are therefore part of the left conjunct. As for the head of the right conjunct, the word *give*, its dependencies could be shared by the whole coordination. A heuristic applied here is that if any of the other conjuncts have a dependency with the same label as some external dependency of the leftmost or rightmost conjunct, that external dependency is private to this conjunct. Here, the word *give* has a dependency `comp:obj` directed to the right. The head of the other conjunct, *Do*, also has a dependency with this label, therefore this dependency is private to and part of the right conjunct. If the word *Do* did not have such a dependency, the word *call* would have to be excluded from the right conjunct and shared by the whole coordination.

[6]Sentence `answers-20111024111513AAAQhAO_ans-0006` from the EWT corpus (Silveira et al., 2014).

With conjunct texts found, they can be measured. This is done in characters by taking the length of the conjunct text, in words and tokens by counting how many of those parser found in the conjunct and in syllables using the `cmudict` from the `nltk` package. In (37) there are contents of the dictionary corresponding to the coordination found in sentence (36).

(37)
```
’L’: {"id": 1,                                ’R’: {"id": 6,
    "text": "Do",                                 "text": "give",
    "lemma": "do",                                "lemma": "give",
    "upos": "VERB",                               "upos": "VERB",
    "xpos": "VB",                                 "xpos": "VB",
    "feats": "Mood=Imp|VerbForm=Fin",             "feats": "Mood=Imp|VerbForm=Fin",
    "head": 0,                                    "head": 1,
    "deprel": "root"}                             "deprel": "conj"}
’Lconj’: ’Do yourself a favor’                ’Rconj’: ’give them a call’
’Lwords’: 4                                   ’Rwords’: 4
’Ltokens’: 4                                  ’Rtokens’: 4
’Lsyl’: 6                                     ’Rsyl’: 4

’conj’: {"id": 5,                             ’other_conjuncts’: []
    "text": "and",                            ’conj_lengths’: [(4, 19), (4, 16)]
    "lemma": "and",                           ’sentence’: ’Do yourself a favor and
    "upos": "CCONJ",                          give them a call.’
    "xpos": "CC",                             ’sent_id’:
    "head": 6,                                ’answers-20111024111513AAAQhAO_ans-0006’
    "deprel": "cc"}
```

A list of dictionaries like the one in (37) is created for every processed corpus file. Then every dictionary becomes an observation in an output .csv table.

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 in the SUD scheme will perform better than the one annotating in the UD scheme. The models were evaluated using the modified version of the `conll18_ud_eval.py` script, same as the one used by Tuora et al. (2021). Scripts used to evaluate their results are available in the github repository related to their study[1] and were adapted here to fit the current experiment.

The difference between the official `conll18_ud_eval.py` script and the one used by Tuora et al. (2021) is the treatment of colons in dependency labels. In UD labels only contain colons if there is language-specific informaion added to the basic label (e.g. `advmod:arg`) – this information is discarded by the original evaluation script. Because of that, the script is not appropriate for SUD data, since in this scheme colons are included in some of the basic dependency labels (e.g. comp:obj). Discarding the part of the label after the colon leads to false evaluation results.

Unlabelled and labelled attachment scores (UAS and LAS) are presented in Table 4.1, along with the absolute difference between them. Both of the differences are significant and in both cases it is in favour of the UD model.

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

4.2 Governor's impact on the coordination

The second hypothesis stated here is that the position of the governor of the coordination influences the placement of the shorter conjunct. Specifically, if the governor is on the left of the coordinate structure, or missing altogether,

[1]https://github.com/ryszardtuora/ud_vs_sud

the left conjunct will be shorter than the right one. The oppostie will be true when the governor is on the right.

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.

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 Žabokrtský, 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.

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.

Silveira, N., Dozat, T., de Marneffe, M.-C., Bowman, S., Connor, M., Bauer, J., and Manning, C. D. (2014). A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*.

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 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.