UiO : Department of Informatics
University of Oslo

Dependency Interconversion

Norveig Anderssen Eskelund

Master's Thesis Autumn 2014

Dependency Interconversion

Norveig Anderssen Eskelund

31st July 2014

Thanks to Stephan Oepen and Lilja Øvrelid

for their dedication, patience and guidance

The Penn Treebank PoS tagset

1. CC	Coordinating conjunction	25. TO	*to*
2. CD	Cardinal number	26. UH	Interjection
3. DT	Determiner	27. VB	Verb, base form
4. EX	Existential *there*	28. VBD	Verb, past tense
5. FW	Foreign word	29. VBG	Verb, gerund/present participle
6. IN	Preposition/subordinating conjunction	30. VBN	Verb, past participle
7. JJ	Adjective	31. VBP	Verb, non-3rd ps. sing. present
8. JJR	Adjective, comparative	32. VBZ	Verb, 3rd ps. sing. present
9. JJS	Adjective, superlative	33. WDT	*wh*-determiner
10. LS	List item marker	34. WP	*wh*-pronoun
11. MD	Modal	35. WP$	Possessive *wh*-pronoun
12. NN	Noun, singular or mass	36. WRB	*wh*-adverb
13. NNS	Noun, plural	37. #	Pound sign
14. NNP	Proper noun, singular	38. $	Dollar sign
15. NNPS	Proper noun, plural	39. .	Sentence-final punctuation
16. PDT	Predeterminer	40. ,	Comma
17. POS	Possessive ending	41. :	Colon, semi-colon
18. PRP	Personal pronoun	42. (	Left bracket character
19. PP$	Possessive pronoun	43.)	Right bracket character
20. RB	Adverb	44. ¨	Straight double quote
21. RBR	Adverb, comparative	45. `	Left open single quote
20. RBS	Adverb, superlative	46. "	Left open double quote
20. RP	Particle	47. ’	Right close single quote
20. SYM	Symbol (mathematical or scientific)	48. ”	Right close double quote

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Dependency grammars have gained enormous popularity for a wide range of natural language processing (NLP) tasks in the past decade and the availability of dependency treebanks is increasing. Many of these treebanks have been created by converting constituent structures in existing constituency treebanks to dependency structures.

Linguists disagree on the syntactic analysis of several linguistic phenomena. Both the constituency treebanks that the dependency treebanks origin from, as well as the conversion procedures applied, may have been designed on the basis of different linguistic theories. As a result, the different dependency formats have diverging representations of various syntactic structures. These differences can surface in the choice of heads, inventory of labels and formal graph conditions. These differences present a challenge to several NLP tasks, e.g. for cross-lingual syntactic parsing.

1.1 Motivation and research questions

Despite the increasing interest in dependency grammars, we still have limited and only high-level knowledge about the similarities and differences between the various annotation schemes for dependency representations, even where multiple schemes exist for the same language. Although some work has been done in this area (see Section 2.7), we will aim at doing a more thorough quantitative and qualitative analysis of some selected dependency formats for English. We will perform an investigation of three annotation schemes in order to acquire and document knowledge about commonalities and differences between them. For our project, we have selected two 'classic' dependency schemes, Stanford Basic Dependencies (SB) (De Marneffe et al. 2008) and CoNLL Syntactic Dependencies (CD) (Johansson et al. 2007), and the more recent DELPH-IN Syntactic Derivation Tree (DT) (Ivanova et al. 2012). In our thesis we seek to determine:

(a) Whether any of the selected formats are more expressive than the others

(b) To what extent the different formats correspond

Ivanova et al. (2012) have conducted a contrastive study of these three formats, among others. The focus of their project was, however, on

formal and representational aspects, rather than on linguistic content. For a selected format pair, CD and DT, we will perform a more qualitative investigation and seek to acquire knowledge of any syntactic phenomena that have a different analysis in the two formats. We will investigate:

(c) What systematic differences exist between the two formats

(d) What methodology we can use to discover these differences

If we could successfully convert corpora annotated in one dependency scheme to another, this would provide new opportunities and be useful for several NLP tasks. Making texts available in more annotation schemes, quality control of parallel annotations over the same text and cross-framework parser evaluation are some potential application areas. We will investigate:

(e) How we can perform rewriting of syntactic structures and labelling

(f) What accuracy we can achieve by heuristic conversion

We choose a heuristic approach for rewriting of syntactic structures. We also use a heuristic method for labelling in this first version of our converter. We can consider the assigning of labels to dependency relations as a classification task, and thus a problem well suited for statistical classification. We will investigate:

(g) To what extent machine-learned classification improve the accuracy of the labelling

1.2 Thesis outline

Chapter 2: Background In this chapter we review some necessary background information for our project. We introduce the notion of dependency grammars and a short description of The Penn Treebank. We present a general method for converting constituent representations to dependency representations and provide a short description of the three dependency formats that we examine in this thesis. We give a summary on previous and related work on dependency interconversion.

Chapter 3: A comparison of three dependency formats This chapter presents the results of our quantitative contrastive study of the three dependency formats. We present the results of various statistical comparisons with respect to granularity of each format and correspondences between the format pairs. In addition, we briefly discuss some well-documented areas where dependency formats differ, with respect to the three formats. We give a summary of the conclusions and assumptions we draw from our investigations.

Chapter 4: How can we perform conversion? This chapter investigates how conversion of dependency structures from one format to another can be performed. We discuss subtasks involved in conversion and possible approaches to them. We propose a quantitative methodology for detection of patterns of different syntactic analyses in format pairs. A baseline converter with a heuristic labelling procedure is described.

Chapter 5: Heuristic conversion from DT to CD In this chapter we describe our work on heuristic conversion of data annotated in the DT format to the CD format. We describe the phenomena that we have found to have different syntactic analysis in the two formats and the result of our heuristic conversion.

Chapter 6: Labelling by classification and contrastive evaluation In this chapter a series of experiments to improve our result with respect to labelling, by using machine learning-based classification, is described. The final evaluation of our conversion on held-out test data is presented here, including comparison of the heuristic and machine-learning approaches.

Chapter 7: Conclusion This chapter sums up the results of our work and considers possibilities for improvement and further work.

Chapter 2

Background

In this chapter we will provide some necessary background information for our project. In Section 2.1 we will introduce the basic notion of dependency grammars. Section 2.2 contains a description of The Penn Treebank, one of the most known constituent treebanks for English, and how it was created. The general method for converting constituent representations to dependency representations is described in Section 2.3. In Section 2.4 – 2.6 we give a short description of the three dependency formats that we will examine in our project. Finally, we give a summary on previous and related work on dependency interconversion in Section 2.7.

2.1 Dependency grammars

The fundamental idea behind dependency grammars is that the syntactic structure of a sentence can be expressed by dependencies between the words of the sentence. These dependencies are binary asymmetrical relations (Nivre 2005). One of the two words linked by a dependency relation is the *head* and the other is the *dependent*. The dependent *depends* on the head and the head *governs* the dependent. A dependency structure can be represented as a labelled directed graph, where every word (node in the graph) is related to at least one other word.

Figure 2.1 below shows a dependency representation (using the Stanford scheme; see Section 2.5) of the sentence *Time flies like an arrow*.

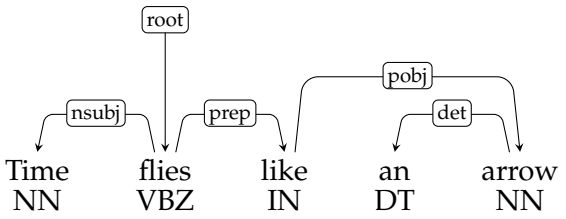


Figure 2.1: Dependency representation in Stanford format

Let us take a closer look at one of these dependency relations:

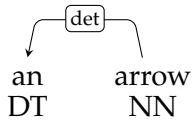


Figure 2.2: Dependency relation

In this dependency the noun NN is the head; the dependency edge originates from this word. The determiner DT is the dependent, it is attached to the head by the arc. The dependency is labelled *det*. Following is the description of this dependency label from the Stanford typed dependencies manual (De Marneffe et al. 2008):

> *det*: **determiner**
> A determiner is the relation between the head of an NP and its determiner.

How can we identify the dependencies of a sentence and, for each dependency, how can we know which is to be considered the head and which is the dependent? Several criteria for identifying relations between a head *H* and a dependent *D* in a construction *C* have been proposed. Here are some of them (Zwicky 1985; Hudson 1984; Nivre 2005):

1. *H* determines the syntactic category of *C* and can often replace *C*.
2. *H* determines the semantic category of *C*; *D* gives semantic specification.
3. *H* is obligatory; *D* may be optional.
4. *H* selects *D* and determines whether *D* is obligatory or optional.
5. The form of *D* depends on *H* (agreement or government).
6. The linear position of *D* is specified with reference to *H*.

These 6 criteria are, as we will see, by no means unambiguous.

Consider for example the following construction: *an arrow*.

Which part of this phrase is head and which is dependent according to criterion 1? If we apply the traditional NP analysis, the construction would be categorized as a noun phrase (NP), where the noun *arrow* is the head of the phrase, and the determiner *an* is the specifier. The dependency shown in Figure 2.1 between these two words is in accordance with this analysis. If we, on the other hand, take the stance that is common in many branches of generative grammar, and apply what is called a DP analysis, the result is the complete opposite. According to the DP analysis, the head of this phrase is the determiner *an*, identifying the construction as

a determiner phrase (DP). The noun *arrow* is considered to be the argument of the determiner.

If we decide to go for the DP-analysis and pin down the determiner as the head, our decision will immediately be contradicted when we apply criterion 2. When we consider semantics, the noun will undoubtedly be the natural choice of head.

According to criterion 3, the head is obligatory while the dependent may be optional. In our example both words in the construction *an arrow* are obligatory (*arrow flies, *an flies). But there are also examples of noun phrases where both the determiner and noun can be optional. The subject noun phrase in the sentence *These arrows fly high* is such an example. Here the determiner *these* or the noun *arrows* can be omitted from the sentence (*These fly high, Arrows fly high*), without making it ungrammatical. In none of these cases will criterion 3 help us identify the head.

Criterion 4 states that the head determines whether the dependent is obligatory or optional. Plural forms of nouns make the determiner optional, e.g. allowing the determiner *the* to be omitted from the phrase *the arrows fly*. This would, according to criterion 4, identify the noun *arrows* as head of the construction *the arrows*. On the other hand, several determiners can occur alone without a noun. Examples of such determiners are *two, many, those*. This makes it possible to omit the noun from phrases like *many arrows fly, those arrows fly*, and thus identify the determiners as the heads of the constructions *many arrows, those arrows*.

It is hard to find an example for English where criterion 5 cannot be used to unambiguously identify the head of a construction. For noun phrases for instance, the noun will always decide the form of the determiner. In German, however, some nouns will assume a strong or weak form depending on the article, for example *ein Beamter* (a civil servant) versus *der Beamte* (the civil servant).

Criterion 6 asserts that the position of the determiner is specified with reference to the head. This might be true, but will in most cases be useless for deciding which word is the head of a construction. For English noun phrases, it is true that the determiner always must be positioned before the noun, but it is equally correct to say that the noun must be positioned after the determiner.

In conclusion, even for a seemingly simple construction like the English nominal group, the head choice can be debated, and syntactic vs semantic criteria may pull in different directions.

A majority of proposed dependency schemes identify the noun as the head of a noun phrase. For some constructions, however, there is no general agreement about how to decide which is the head and which is the dependent(s). Such problematic constructions are constructions involving coordination, prepositional phrases or grammatical function words.

Some formal conditions are imposed on dependency structures (Nivre 2005). One of these conditions is that a dependency structure has to be *connected*. Every node in the graph has to be connected to at least one other

node in order to make the graph complete. Two other constraints that most dependency grammars adhere to, are the constraint of *acyclicity* and the constraint of *single-head*. Acyclicity implies that the structure is hierarchical, it does not contain cycles. The constraint of single-head means that one node should have at most one head.

One of the most important issues in dependency grammar is the notion of projectivity, and whether this should be a constraint on the linear realization of dependency structures or not. A dependency A –> B adheres to the constraint of projectivity, if all nodes that occur in the linear order between A and B are transitively dependents of A. A dependency graph that satisfies the constraint of projectivity will not have any crossing edges. In languages with free or flexible word order, sentences that can not be represented by a projective dependency graph, occur rather frequently. Even in English, some constructions, like relative clause extraposition, create non-projective dependencies. Figure 2.3 illustrates this. In this example non-projectivity is caused by designing *arrow* to be the head of *hit*.

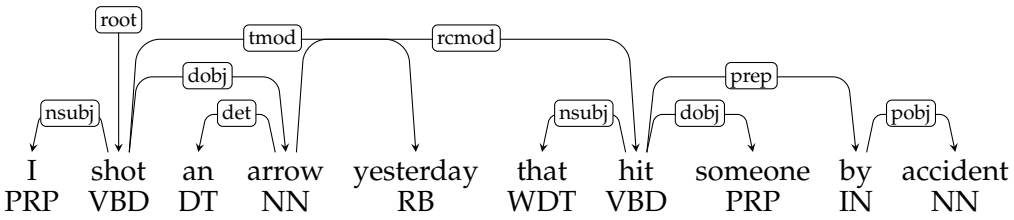


Figure 2.3: Dependency representation of an English sentence with an extra-posed relative clause

2.2 The Penn Treebank

Treebanks are collections of sentences that have been manually annotated with a correct syntactic analysis and part-of-speech (PoS) tags. Treebanks are an important resource both for linguistic research and natural language processing (NLP). In NLP treebanks are used for evaluation and comparison of parsers as well as for machine-learning of statistical models and estimation of probabilities for context-free grammars.

The Penn Treebank (PTB) is an annotated corpus consisting of over 4.5 million words of American English (M. P. Marcus et al. 1993) and thus arguably the largest existing treebank for English. In contrast to dependency grammar, the annotation used for PTB is a constituency representation. Constituents are single words or phrases; groups of words acting as units. A constituent can consist of other constituents, thus forming a hierarchy. An example of a constituency representation is shown in Figure 2.4.

The first version of PTB was collected and annotated during the period

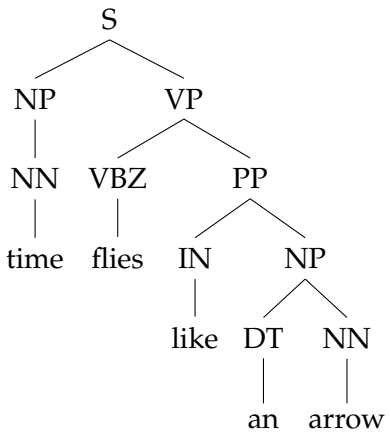


Figure 2.4: Constituency representation

1989–1993. The annotation process for PTB consisted mainly of two parts; PoS tagging and syntactic bracketing. The PoS tagging of the PTB was performed using a combination of automatic PoS assignment and manual correction. The tagset used for PoS tagging of PTB is based on the tagset of the pioneering Brown Corpus (Francis 1964; Francis et al. 1982). However, where the Brown Corpus uses 87 simple tags and also allows compound tags, PTB has through simplification of this tagset, ended up with a tagset of 36 PoS tags and 12 additional tags for punctuation marks and other symbols. A reason for this simplification was the desire to avoid tagging inconsistencies, as a reduced tagset is expected to reduce the chances of such inconsistencies.[1]

In order to create the best possible basis for the bracketing task, the PTB project strived to tag words according to their syntactic functions. For instance a verb in its gerund/present participle form can be tagged as a verb, noun or adjective depending on the syntactic context. In the Brown Corpus, however, words are often tagged without regard to their syntactic functions.

A third difference between the Brown Corpus and PTB is that multiple taggings are allowed in PTB. If a word is tagged with more than one tag, it indicates either real ambiguity or that the annotator was not able to decide which was the correct tag.

The bracketing task was also performed in two stages, an automatic and a manual one. In the automatic stage, a skeletal, bracketed representation was produced by a parser. In the manual stage, annotators corrected and completed this output by hand. The tagset used for bracketing in PTB, consists of 14 syntactic constituent tags plus 4 null elements. The choice of tagset and skeletal representation as output from the parser, was to a large extent influenced by the need for efficiency in the hand correction process.

[1]One of the strategies for simplifying the PoS tagset, was to eliminate redundancy by removing forms that define distinctions that are lexically or syntactically recoverable. For instance does the tagset not contain different tags for auxiliaries and main verbs.

The project needed, with limited time and human resources, to create a large annotated corpus.

A result of this pragmatic approach is that many syntactic details are left unannotated. Some examples of this are:

- a distinction between arguments and adjuncts is not always made
- noun phrases with a complex internal structure are often annotated as flat structures
- the head of a phrase is only rarely explicitly identified

When developing PTB version II (M. Marcus et al. 1994), it was decided to expand the representation to a richer structure. This was done in order to allow for i.a.:

- the automatic recovery of predicate-argument structure
- a distinction between verb arguments and adjuncts

In this improved representation, the tagset was extended with more null elements and, functional tags/property labels were introduced.

Here is an example of a sentence annotated in PTB I style, without functional tags (left) and the same sentence annotated in PTB II style, with functional tags (right):

```
( S ( NP an arrow )                 (S ( NP-SBJ an arrow )
   ( VP hit                            ( VP hit
        ( NP somebody )                     ( NP somebody )
        ( PP on                             ( PP-TMP on
             ( NP Monday ))))                    ( NP Monday ))))
```

Note that the subject is explicitly tagged as subject, although in this particular construction this can be recognized from the syntactic structure (the subject is the NP left of the VP in English) and is in fact redundant. Table 2.1 is reproduced from (M. Marcus et al. 1994) and shows the functional tags introduced in PTB version II.

Another version of PTB was developed in 1999, but no further extentions to the annotation scheme were made. However, we observe that Johansson 2008 report to have retained 4 labels (BNF, DTV, EXT and PUT) that are not found in this table, from PTB annotations. These labels are listed in Table 2.2. We must assume that these for some reason have been omitted from the original table, or have been added to the set of function tags without explicit communication, possibly between PTB versions 2 and 3.

Tag	Marks:
Text Categories	
-HLN	headlines and datelines
-LST	list markers
-TTL	titles
Grammatical functions	
-CLF	true clefts
-NOM	non NPs that function as NPs
-ADV	clausal and NP adverbials
-LGS	non VP predicates
-SBJ	surface subjects
-TPC	topicalized and fronted constituents
-CLR	closely related - see text
Semantic Roles	
-VOC	vocatives
-DIR	direction & trajectory
-LOC	location
-MNR	manner
-PRP	purpose and reason
-TMP	temporal phrases

Table 2.1: Functional tags used in PTB II, table reproduced from (M. Marcus et al. 1994). Note that the comment on the -CLR tag refers to the original article, not this thesis.

Tag	Meaning
BNF	Benefactorer
DTV	Dative
EXT	Extent
PUT	Various locative complements of the verb put

Table 2.2: Functional tags in CoNLL 2008 reported retained from PTB II, but missing in M. Marcus et al. 1994.

2.3 Constituent-to-dependency conversion

Dependency grammars and dependency parsing have become increasingly popular during the last decade. Still, no significant treebank for English has been built using dependency representations. Instead, methods for converting constituent representations to dependency representations have been developed and applied for converting (parts of) PTB to a dependency treebank, by several projects. The first acknowledged conversions were made by Magerman 1994; Baker et al. 1998; Yamada et al. 2003.

A crucial part of constituent-to-dependency conversion is the identi-

fication of the head in each phrase. As is obvious from the discussion in Section 1 above, identifying heads is necessary to be able to create dependencies. This is usually done using a set of head rules. The use of such rules has a long tradition. Magerman 1994 was the first to produce a set of head percolation rules. Others have used modifications of this set or produced their own sets of rules (Collins 1997; Johansson et al. 2007), inter alios. The basic procedure for conversion from a constituent representation to a dependency representation can be described by this generic algorithm:

1. Select a target constituent
2. Find its lexical head using head percolation rules
3. Find sibling constituents along head path
4. For each sibling constituent the lexical head becomes a dependent of the target constituent
5. Invoke step 1–4 recursively with each constituent as target.

We will take a look at these basic conversion rules by using them to convert the simple constituency representation shown in Figure 2.5, to a dependency representation:

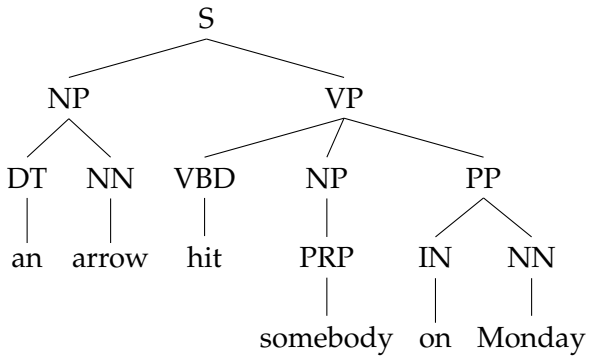
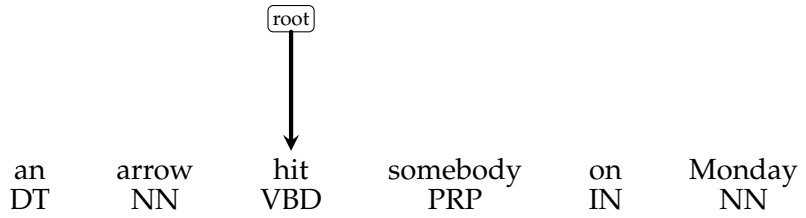


Figure 2.5: Constituency representation for conversion

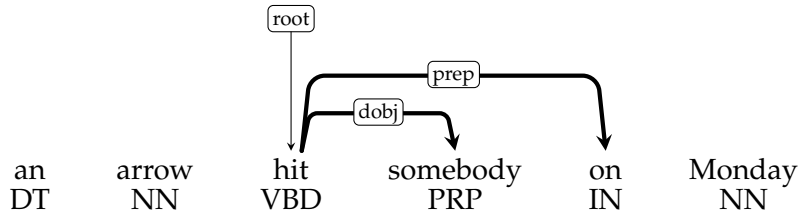
We will use the following (tiny) set of head percolation rules in our conversion ('*' denotes the head of the phrase, elements inside parenthesis '()' are optional and ',' is used to separate alternatives):

PP –> IN* NN
NP –> DT NN*, PRP*
VP –> VBD* NP (PP)
S –> NP VP*

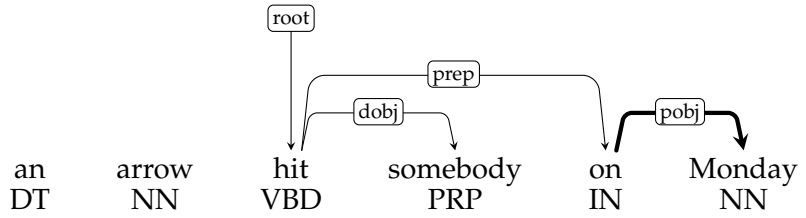
Starting at S, the rules above guide us through the VP to the VBD as the lexical head.



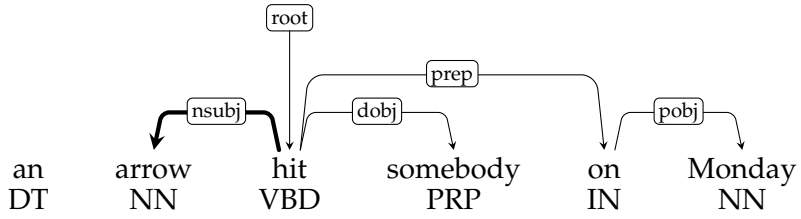
This VBD has two siblings, the NP where the PRP is the head and the PP where IN is the head. These are both dependents of the VBD.



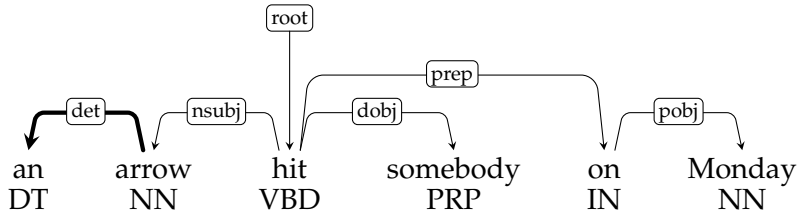
The procedure is invoked recursively with the NP as the first target constituent and the PRP identified as the lexical head. The PRP has no siblings. The next target constituent is the PP, where IN is the lexical head. The IN has one sibling, the NN, which is a dependent of the IN.



The path beneath the VP constituent is now exhausted, but it has one sibling, the NP where the NN is identified as the lexical head. This NN is also a dependent of the VBD.



Recursive invocation of the procedure, with the NP as the target constituent and the NN as the lexical head, identifies the DT sibling as a dependent of the NN.



As we have seen, we need both a constituency representation and a set of head rules as parameters to a conversion function. A third parameter is also needed; a dependency annotation scheme. In our example, we have used the Stanford format. There are some variations in these dependency labelling functions. In the next sections we will take a closer look at some of the most used annotation schemes.

2.4 The CoNLL 2008 Shared Task

The annual shared task of the Conference on Computational Natural Language Learning in 2008, was to process a unified dependency-based formalism, modelling both syntactic dependencies and semantic roles.

The annotation practice and conversion method that was developed (Surdeanu et al. 2008; Johansson et al. 2007), build on and aim to improve existing methods (Magerman 1994; Baker et al. 1998; Yamada et al. 2003; Nivre et al. 2006) for converting constituent representations from the PTB to dependency representations.

The new conversion procedure exploits the information in the PTB better than the previous methods. It makes use of extra information given in the extended structure of the later PTB versions and uses a larger set of dependency labels.

The new procedure uses heuristic rules to add internal structure to complex noun phrases. Different internal structure is apparent in the following two examples:

DT [[NN NN] NN]
the linen towel rack

DT [NN [NN NN]]
the steel towel rack

Without the distinction in the internal structure, shown by the bracketing, in these two NPs, the dependency representation of them would be the same. The rules used to identify the internal structure are heuristic in the sense that they will not capture the structure of all complex NPs.

Another change in the new procedure is that it uses more advanced head percolation rules. These rules make use of function labels. In these rules some new categories are introduced, i.e. *-PRD (any phrase with a PRD function tag) and NP-ε (NP with no function tag).

The new procedure also uses a richer set of dependency arc labels. Most of the function tags in Table 2.1 are used to label dependencies. Exceptions are HLN, TTL, NOM and TPC. The first three were left out because they were considered not to reflect any grammatical function. The latter, the TPC tag, is used in combination with a NULL element, to mark a topicalized argument:

(S (NP-TPC-1 this feeling)
(NP-SBJ everybody)
(VP has experienced
(NP *T*-1)))

This TPC label was not retained because it was considered not relevant for a dependency grammar; an object is an object, whether it is topicalized or not.

In the new procedure the heuristic rules, used to infer labels for edges that have no labels in the PTB, were extended. One of the results of these modified rules is that a distinction between direct objects (OBJ) and indirect objects (IOBJ) is made.

2.5 The Stanford typed dependencies representation

The Stanford typed dependencies representation was released in 2008. It builds on and has used parts of the GR (Carroll et al. 1999) and PARC (King et al. 2003) schemes, but is developed to provide a simple and easy-to-understand description of grammatical relationships. The aim was to make a representation that could be used and understood also by non-linguists.

All grammatical relationships are represented as binary relations between pairs of words. The relations are arranged in a hierarchy that contains 56 dependency labels.

The following 6 design principles were used when developing the representation (De Marneffe et al. 2008):

1. Everything is represented uniformly as some binary relation between two sentence words.

2. Relations should be semantically contentful and useful to applications.

3. Where possible, relations should use notions of traditional grammar for easier comprehension by users.

4. Underspecified relations should be available to deal with the complexities of real text.

5. Where possible, relations should be between content words, not indirectly mediated via function words.

6. The representation should be spartan rather than overwhelming with linguistic details.

Figure 2.6 below shows a sentence annotated in both the CoNLL and the Stanford formats.

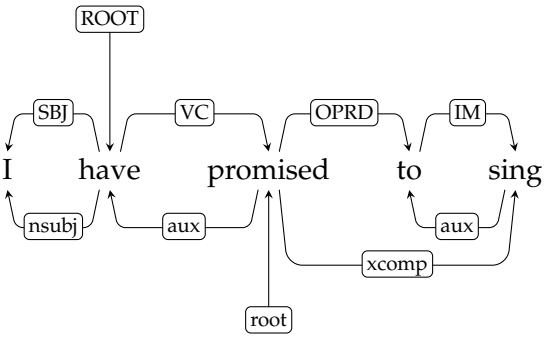


Figure 2.6: Dependency representation in CoNLL (above) and Stanford (below) format.

The differences between these two representations can be said to illustrate the use of design rule 5. In the Stanford format the content word *promised* and not the function word *have* is chosen to be the dependent of the root. This is also in accordance with rule 2, that relations should be semantically contentful.

The Stanford representation makes no distinction between arguments and adjuncts, but contains many labels representing NP-internal relations. Some of these labels are: *abbrev* (abbreviation modifier), *appos* (appositional modifier), *det* (determiner), *infmod* (infinitival modifier), *nn* (noun compound modifier). As such relations are assumed to be critical to applications, this is in accordance with rule 2.

According to rule 5, content words should be chosen as heads of dependents. This leads to the collapsing of dependencies including conjuncts. The Stanford scheme also offers an alternative, collapsed

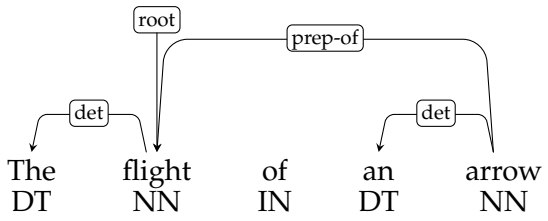


Figure 2.7: Collapsed representation of a prepositional complement

representation for other constructions, i.e. prepositional complements. Figure 2.7 shows an example of such a collapsed representation.

Here the preposition is used as a relation and does not appear as a node in the dependency structure any more. As such collapsed representations reflects the semantics of the sentence better than the basic variant, they can also be said to adhere to rule 2.

As a result of rule 4, the Stanford scheme contains some more generic labels, that can be used when a more precise dependency relation can not be determined. The most generic of these labels is the *dep* (dependent) label.

2.6 DeepBank and DELPH-IN syntactic dependencies

The English DeepBank (Flickinger et al. 2012) is a treebank that has been developed over the last 2 years. It contains the text of the first 22 of the 25 Wall Street Journal sections that are included in the PTB. The annotation of the DeepBank is grammar-based and grounded in the Head-driven Phrase Structure Grammar (HPSG) framework (Pollard et al. 1994). This annotation is expected to be richer and more fine-grained than the annotation used in PTB.

In the development of the DeepBank, no manual marking up of syntactic structure was involved. The corpus was first parsed with a parser using the English Resource Grammar (ERG) (Flickinger 2002), a broad coverage implementation of HPSG, developed over a period of 20 years. The result was then manually disambiguated by annotators using the [incr tsdb()] tool (Oepen 1999). This tool offers a discriminant-based method for treebanking, where the annotators have to make a set of binary decisions to include or exclude elements of the suggested analysis. By avoiding additional manual annotations, the tight connection to the ERG is kept intact.

Recently Ivanova et al. (2012) have conducted a contrastive study of different dependency formats. The motivation for this project was to identify differences and similarities between the different formats and to facilitate making treebanks, annotated with HPSG analyses, available to more groups of potential users. To achieve this second goal, an automated conversion procedure that converts annotations based on the HPSG framework to bi-lexical dependencies, was developed.

As the HPSG framework is founded on a head-based theory, the HPSG trees contains constructions from which heads can be identified directly. Because of this, the transformation from HPSG trees to syntactic dependencies is rather straightforward.

An HPSG tree representation of the sentence *Time flies like an arrow* will look like this (Figure 2.8):

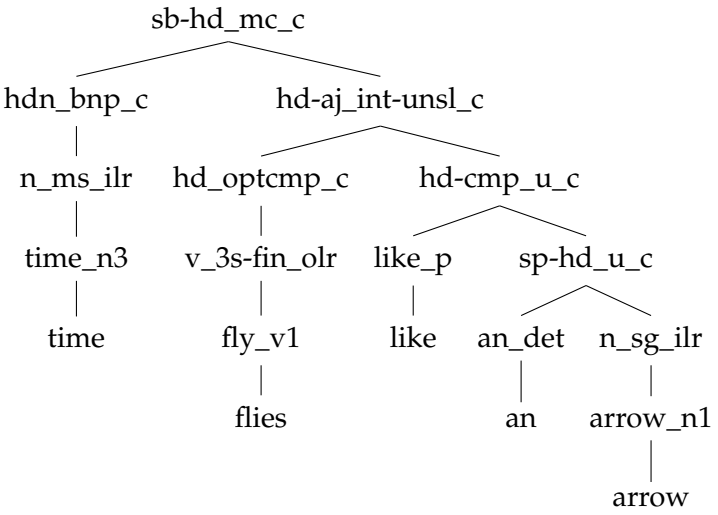


Figure 2.8: HPSG analysis tree

The labels of the nodes are used both for identifying heads and for labelling dependencies. The node labels are identifiers of HPSG constructions. Ivanova et al. 2012 have generalized the 150 ERG constructions to 52 major construction types that are used for labelling dependencies, i.e. SB-HD (subject head), HD-AJ (head adjunct), HD-CMP (head complement), SP-HD (specifier head). Figure 2.9 shows the result of converting the tree above to a dependency representation.

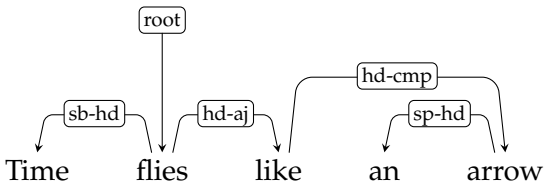


Figure 2.9: HPSG tree converted to a dependency representation

The conversion procedure was applied to the DeepBank. As a result, we now have 3 dependency structures over the same text.

2.7 Dependency interconversion: previous and related work

In this section we will present some of the earlier work that can be related to interconversion of dependency representations. Several projects have used a limited conversion of syntactic structures in dependency representations in order to improve parsing results. In these projects parsers are trained on converted data, but the output from the parsers is converted back to the original syntactic structures before evaluation. Such temporary conversion is often referred to as transformation. Other projects have focused on detecting and describing systematic differences between dependency formats. Some of these projects propose new annotation schemes and/or supply conversion tools as well. Only one project has attempted a full-scale conversion from an existing dependency format to another.

Nivre et al. (2005) propose pseudo-projective parsing for allowing the use of a data-driven parser restricted to projective dependency graphs, on non-projective structures. They projectivize (make projective) dependency structures by a minimal transformation. This projectivization is done by lifting operations, making the dependent in a former non-projective structure a dependent of the head of its former head.

Nilsson et al. (2006) use data from the Prague Dependency Treebank (Hajic et al. 2001) and transform coordinated structures 3.4 and verb groups from Prague style to Melčuk style for improved parsing. The parsed sentences are then transformed back, to the extent that this is possible, to their original Prague style annotation.

Bengoetxea et al. (2009) explore dependency transformation and inverse conversion on the Basque Dependency Treebank (Aduriz et al. 2003). In their experiments they use language-specific transformation of coordination structures and subordinated clauses as well as language-independent projectivization. They find that transformation can improve parsing results and that the order of transformations can be relevant.

Ivanova et al. (2012) conduct a contrastive study of seven different dependency formats, among them the three that are also the topic of our own work: CoNLL Syntactic Dependencies (CD), Stanford Basic Dependencies (SB) and DELPH-IN Syntactic Derivation Tree (DT). Their study is based on ten sentences from PEST (Bos et al. 2008) available in all seven formats. An analysis of these sentences indicates that CD and DT, as well as CD and SB are among the most similar format pairs, and that CD and DT is the most similar of the three pairs that we are going to investigate.

Schwartz et al. (2012) experiment with varying syntactic structures, i.e. structures that have alternative annotations within the same formalism, and their effect on parsing performance. The varying syntactic structures that they explore are: coordination structures, infinitive verbs, noun phrases, noun sequences, prepositional phrases and verb groups.

Popel et al. (2013) offers a systematizing view of the different coordin-

ation models used in dependency grammars. They propose a taxonomy for describing different representations of coordination structures, found in existing treebanks or described in literature. They also implement a tool that converts coordination structures from any taxonomy style to any other taxonomy style.

Zeman et al. (2012) have studied treebanks for 29 languages and try to identify all syntactic constructions where annotation systematically differs in at least one of the treebanks. Among these observed constructions are coordination structures, prepositional phrases, subordinated clauses, verb groups, noun phrases and punctuations. They propose a common normalized annotation format, mainly derived from the annotation style of the Prague Dependency Treebank, and provide methods and software for automatic conversion to this format. Their conversion procedure involves both structural rewriting and relabelling of dependency relations.

In their experiments on framework-independent evaluation of syntactic parsers, Miyao et al. (2007) attempt not only to convert parser output in Enju XML format (Miyao 2007) to Grammatical Relations (GR) (Carroll et al. 1998) and Stanford Typed Dependency scheme (SD) (De Marneffe et al. 2006), but also to convert parser output in the SD format to GR. A problem for the project is that no manually annotated gold standard corpus for SD is available at the time. They obtain data in the SD format by using a conversion program included in the Stanford Parser on shallow PTB-style phrase structure trees. They use heuristic rules to perform their own conversions and identify several undocumented disagreements between the formats. Their conversions result in an accuracy slightly above 80% and they conclude that undocumented differences make format conversion a non-trivial and challenging task.

As far as we know, there has been no previous attempt at full conversion between any pairs of the formats Stanford Basic Dependencies, CoNLL Syntactic Dependencies and DELPH-IN Syntactic Derivation Tree.

Chapter 3

A comparison of three dependency formats

Our task is to design, implement and evaluate an automated converter for conversion between some common dependency representation formats: Stanford Basic Dependencies (SB), CoNLL Syntactic Dependencies (CD) and DELPH-IN Syntactic Derivation Tree (DT). Before we do this we need to gain more knowledge about each of these formats and about similarities and correspondences between the different format pairs. This knowledge is required to be able to make an assessment of which pairs and in which directions conversion is most likely to be successful. Because literature on the area is rather sparse, we will use a quantitative approach in addition to studies of existing documentation.

In Section 3.1 we give an overview of which data sets we have used in our study and the preprocessing we applied to them. Results of various statistical comparisons with respect to granularity of each format and correspondences between the pairs are presented in Section 3.2 and Section 3.3. The next sections contain brief discussions of two comparatively well-documented areas where dependency formats differ: coordination in Section 3.4 and projectivity in Section 3.5. In Section 3.6 we give a summary of the conclusions and assumptions we draw from our investigations and select the format pair and the direction of conversion that we will investigate further.

3.1 Data preparation

The data we will use is a subset of the Wall Street Journal data (see Section 2.6), Sections 00–21, annotated in the three selected dependency formats[1]. The tokenization of DeepBank differs a bit from the tokenization in Penn Treebank. We limit our data to sentences that are identically tokenized in all three formats. This leaves us with 36423 sentences. We split out sections 20 and 21, 3089 sentences and 68456 dependencies, to

[1]These are the data used by Ivanova et al. (2013). We are grateful to Angelina Ivanova for preparing these data and sharing them with us.

be used as held-out test data. This leaves us with datasets containing 33334 sentences and 733618 tokens and dependencies, for training and development.

In order to perform comparison of the formats, some normalization of the data is required. To ensure that all three datasets are equally PoS-tagged, we perform a comparison of the PoS tag of each word in all formats. The results are shown in Table 3.1. As we can read from this table, there are some differences in the PoS-tagging of the datasets for CD, DT and SB. The most extensive difference, with respect to PoS tags, is between DT and CD and between DT and SB, the percentage of diverging PoS tags being 3.71 and 3.91, respectively. The difference between CD and SB is considerably smaller, diverging PoS tags are found in only 0.19% of the tokens in this format pair. The explanation for this is probably that while PTB has been tagged using a combination of automatic PoS-tagging and manual correction, DeepBank is fully automatically tagged, using the TnT tagger (Brants 2000).

	Percentage of diverging PoS tags			Percentage of sentences with diverging PoS tags		
	CD	DT	SB	CD	DT	SB
CD		3.71	0.19		52.29	2.08
DT	3.71		3.91	52.29		53.08
SB	0.19	3.91		2.08	53.08	

Table 3.1: Diverging PoS tags

Dealing with different PoS tagging in the different formats, will make the task of our project considerably harder. We therefore decide to 'patch' the DT data with PoS tags from the corresponding CD data. We do this by writing a program that leaps through the DT and CD datasets in parallel, sentence by sentence, token by token. Each time a token with different PoS tags in each format is encountered, the PoS tag in the DT dataset is replaced by the PoS tag from the CD file. In DeepBank the TnT tagger output was used only to determine the lexical categories for unknown words. Inspecting our 'patched' DT data, we find that 265 words that we have re-tagged, were originally unknown to the ERG parser. These words are distributed among 262 sentences. We suspect that these PoS tags might have had an effect on the dependency structure, and decide to remove these 262 sentences. We are now left with 33072 sentences and 726867 dependencies for training and development. We compare the now identically tagged DT and CD data with the SB data. SB uses a different notation style for bracketing tags than DT and CD, -LRB- and -RRB- versus (and). We can handle these notation variations without difficulty. The comparison reveals 11 real PoS tag differences. These are shown in Table 3.2.

These 11 words with PoS tag differences can represent a problem for us when we want to compare the different formats. They are found in

CD , DT	SB	Total
DT	CC	9
PDT	CC	2

Table 3.2: Diverging PoS tags after removal of sentences tagged with the TnT tagger in DT

11 different sentences and we decide to remove these sentences from our three datasets. We are now left with datasets containing 33061 sentences and 726556 tokens/dependencies each, for training and development.

We perform the same conversions on the held-out test data; 'patch' DT with PoS tags from CD, remove sentences with tokens tagged with the TnT tagger and remove sentences with PoS tag differences other than left and right brackets.

The test datasets now contain 3038 sentences and 67111 dependencies. We decide to split out sections 17 and 18, 3389 sentences and 74576 dependencies, from our training data and use these for development. Our final training data now contains 29672 sentences and 651980 dependencies per dataset. An overview of the data sets are given in Table 3.3.

	Sentences	Tokens
Training	29672	651980
Development	3389	74576
Test	3038	67111

Table 3.3: Overview of data sets used in our study

3.2 Estimating linguistic granularity and variability

A linguistically rich annotation scheme will have a high degree of *granularity* and *variability*. By granularity, we mean the level of detail that the annotation provides for. A fine-grained dependency scheme will have many possible combinations of PoS tags and relation types, whereas a more coarse-grained format will have fewer available combinations and thus provide less detail. Variability denotes the level of detail actually expressed in the format. A format might have a high degree of granularity and still obtain a low degree of variability, if not fully utilizing its granularity. We assume that conversion from a format that is linguistically rich to a format that is more coarse-grained, will be most likely to succeed, although such conversion will lead to loss of information. In order to investigate the granularity and variability of the annotation schemes, we will perform a quantitative study of them.

We start by calculating the possible combinations of PoS tags and relation types for each format (granularity). The result is presented in Table 3.4. The first row shows the number of PoS tags used in our data.

As our data sets are equally tokenized, this number is the same for all three formats. We see that 45 of the 48 PoS tags available in the PTB tag set are used. The second row shows the number of available dependency labels according to documentation on each format; CD (Surdeanu et al. 2008), SB (De Marneffe et al. 2008), DT (*ERG Tags*)[2]. The third row shows the number of possible combinations of dependency labels and *head PoS tags* and the number of possible *dependent PoS tag* and label combinations. The last row shows the result of calculating possible combinations of dependent PoS tags, head PoS tags and dependency labels. We observe from these numbers that CD has the highest granularity among the three formats, due to its comparatively large set of labels. DT has a slightly lower granularity than SB.

	CD	DT	SB
postag	45	45	45
label	69	52	56
hpos-label / dpos-label (posttag x label)	3105	2340	2520
dpos-hpos-label (postag x label x postag)	139725	105300	113400

Table 3.4: Counts of possible combinations for each format

To investigate the variability of our formats, we perform counts on labels and PoS tags, and combinations of these, that are actually used in our data. The results, both in absolute and relative frequencies, are presented in Table 3.5. The first row shows the total number of dependency labels used in each format. *hpos-label* shows the number of combinations of dependency labels and head PoS tags used. *dpos-label* shows the number of dependent PoS tag and label combinations.

	CD	DT	SB	CD	DT	SB
label	62	50	49	89.9	96.2	87.5
hpos-label	546	588	677	17.6	25.1	26.9
dpos-label	688	690	577	22.2	29.5	22.9
dpos-hpos-label	3503	3541	3479	2.5	3.4	3.1

Table 3.5: Counts of different combinations used in each format

We can see that only some of the possible combinations of PoS tags and labels are used. Although these proportions differ a bit between the different formats, the numbers of combinations of dependent PoS tags, labels and head PoS tags (*dpos-hpos-label*), shown in the last row, are surprisingly similar. This could indicate that the difference in variability between the schemes is trifling. It also indicates that CD, although it has

[2]According to documentation, 48 ERG labels and 4 additional technical labels (root, punct, mwe, neg) exist for the DT scheme.

a higher granularity (more available combinations of PoS tags and labels), does not actually exhibit a higher degree of variability by utilizing them.

We acknowledge that the distribution of the different PoS tag and label combinations might also be considered when estimating the variability of a format. In one format a large number of the combinations used may occur rarely, while the used combinations are more evenly distributed in another. Knowledge of such differences in distribution might provide more certain information about the actual variability of the formats. However, we will not do any further investigations of this in our study.

As a complementary basic statistics, Table 3.6 shows, for each format, the average number of dependents per head. We see that SB designates fewer nodes as heads than the other formats, because each head in this format has more dependents attached to it. This has not had the effect of diminishing the combinations of labels and head PoS tags used in this scheme, but rather the opposite. As we can se from Table 3.5, SB has the highest number of combinations of label and head PoS tags among the three formats.

	CD	DT	SB
Avg. no dependents per head	1.86	1.69	2.27
Avg. tree-depth	7.75	7.93	6.35
Max. tree-depth	24	25	20

Table 3.6: Tree-depth of formats

Table 3.6 also shows the average and maximum *tree-depth*, the number of nodes in the longest path from the root to a terminal node in a sentence, in the three formats. We can see that SB has a lower tree-depth than the other two formats. This corresponds to the higher concentration of dependents per head in this annotation scheme. The similarity in tree-depth between CD and DT might indicate that conversion between these formats is easier than conversion including SB.

The variety of PoS tags of tokens that are selected as roots might also say something about the richness of an annotation scheme. Table 3.7 shows the 10 most common PoS tags for tokens used as roots and their percentage for each format. Table 3.7 confirms some of the facts we already know about the formats. One of these facts is that in the SB scheme, content words are preferred as heads (De Marneffe et al. 2008). This explains that non-finite verbs forms (VB, VBG, VBN) occur as roots far more often in the SB format than in the other formats, while modal verbs (MD), in contrast to the other formats, hardly ever are used as roots in SB. It can also explain why, as we can see from the table, there is a higher variation among PoS tags frequently used as roots in SB than in the other formats.

Another phenomenon that Table 3.7 reveals, is that DT treats coordination different from SB and CD (Ivanova et al. 2012). In DT the coordinating conjunction (CC) often appears as root, in the other formats it practically

	CD	DT	SB
VBD	43.5	38.3	35.8
VBZ	28.3	24.2	15.7
VBP	14.3	11.6	7.4
MD	8.2	7.1	0.1
CC	0.0	13.3	0.0
VBN	0.5	0.5	13.3
VB	0.7	0.6	8.2
NN	1.1	0.8	5.6
JJ	0.1	0.1	5.0
VBG	0.1	0.0	4.0
	96.8	96.5	95.1

Table 3.7: Most common PoS tags of tokens used as roots in each format

never does. We will discuss coordination structures more thoroughly in Section 3.4.

3.3 Similarity between formats

We go on to examine pairs of formats, in an attempt to find out something about similarities and differences between them and, possibly, more about *convertability*; what format we can most successfully convert to another. For this purpose as well, we will use a quantitative methodology. We will make use of the terms *matched* and *unmatched* dependencies. Matched dependencies are dependencies that have the same head and dependent in both formats. In an unmatched dependency the dependent is assigned different heads in the two formats. Figure 3.3 illustrates a dependency structure in both CD (above) and SB (below) format that has 1 matched and 4 unmatched dependencies. '*I*' is the only word (dependent) that has the same head, '*have*', in both formats, thus being part of a matched dependency for this format pair. Matched and unmatched dependencies are more commonly referred to as *aligned* versus *unaligned* dependencies.

We will start by calculating the unlabelled attachment score and the unlabelled sentence accuracy for the format pairs. The unlabelled attachment score is the percentage of matched dependencies, dependencies that have the same head and dependent in both formats, disregarding the dependency label. The unlabelled sentence accuracy is the percentage of sentences consisting only of such matched dependencies. The result of these calculations is presented in Table 3.8. The unlabelled attachment score is quite similar for the DT/CD and the CD/SB pairs and considerably lower for the DT/SB pair. This confirms that DT/SB is the most dissimilar of our format pairs. This is in accordance with the observations made by Ivanova et al. (2012), although the similarity of the unlabelled attachment score for the DT/CD and CD/SB pair, might be a bit surprising when compared to their results. The unlabelled sentence accuracy is quite low

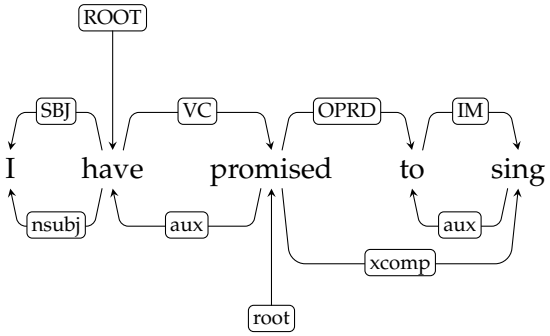


Figure 3.1: Dependency representation in CD (above) and SB (below), having 1 matched and 4 unmatched dependencies.

for all pairs, but remarkably low for the format pairs where DT is included.

	Unlabelled attachment score			Unlabelled sentence accuracy		
	CD	DT	SB	CD	DT	SB
CD		73.6	72.7		1.2	13.1
DT	73.6		55.7	1.2		1.2
SB	72.7	55.7		13.1	1.2	

Table 3.8: Percentage of matched dependencies in format pairs

In the DT format the head of a punctuation token is usually the preceding word (Ivanova et al. 2013). In SB and CD, however, these tokens used for punctuation usually have heads further up in the dependency tree. Knowing this, we want to find out if the unlabelled sentence accuracy for the CD/DT and DT/SB pairs will increase if we leave out the tokens with punctuation PoS tags from our calculations. The result is shown in Table 3.9.

	Unlabelled attachment score			Unlabelled sentence accuracy		
	CD	DT	SB	CD	DT	SB
CD		80.8	72.9		17.7	13.1
DT	80.8		60.5	17.7		5.3
SB	72.9	60.5		13.1	5.3	

Table 3.9: Percentage of matched dependencies (excluding punctuation tokens) in format pairs

As we can see from Table 3.9, leaving out punctuation tokens results in increased values for all of the similarity scores. An exception is the unlabelled sentence accuracy for the CD/SB pair, which is unchanged. The unlabelled attachment score for this pair is just slightly increased,

from 72.7 to 72.9. This implies that punctuation marks are, in most cases, identically attached in the CD/SB pair. This explains the comparatively high unlabelled sentence accuracy that we observed for this pair in Table 3.8. The most considerable increase, when punctuation tokens are excluded, is in the unlabelled sentence accuracy for DT/CD, from 1.2 to 17.7. The unlabelled sentence accuracy for DT/SB is still very low. The most similar formats, when we disregard punctuation tokens, are DT and CD, the pair having an unlabelled attachment score of 80.8.

Table 3.10 shows the percentage of the 29672 sentences included in our training data, that have the same root in both formats. We can see that there is a considerably higher correspondence between roots in the DT/CD pair, than there is in the other pairs. This is in correspondence with the distribution of PoS tags for tokens used as roots in each format, as shown in Table 3.7. These distributions are quite similar for CD and DT, with the use of tokens PoS-tagged CC (coordinating conjunction) in DT being an exception. This, as well as the high unlabelled attachment score for the DT/CB pair when punctuation marks are excluded, indicates that DT and CD are the most similar among our formats, as has also been suggested in the much more limited study of Ivanova et al. (2012).

	CD	DT	SB
CD		84.4	62.5
DT	84.4		53.7
SB	62.5	53.7	

Table 3.10: Percentage of identical roots in format pairs

In Table 3.11 the percentage of unmatched (unaligned) dependencies that are linked to the root in one of the formats is shown per format pair. As we can see, the largest percentage of unmatched dependencies attached to the root is 36.0, found in the SB format in the CD/SB pair. In other words, almost one third of the unmatched SB dependencies, when comparing to DT, are right at the root of the tree. Slightly lower is this share for the SB format in the DT/SB pair and for the CD format in the DT/CD and CD/SB pairs. From this we assume that when converting to the CD or SB formats, identifying the correct root in the target format could be crucial for successful conversion of syntactic structures. The numbers in Table 3.11 are in accordance with the 'heavy' head factor in Table 3.6, where we see that SB and CD have a higher average number of dependents per head than the DT format.

	DT/CD	DT/SB	CD/SB
CD	35.4		30.8
DT	8.0	15.1	
SB		35.6	36.0

Table 3.11: Percentage of unmatched dependencies attached to root

We perform further comparisons of the different format pairs by counting combinations of PoS tags and dependency labels in both formats and comparing them. For each token we count the combinations of source format label and target format label of the relation between the token and its head. The counts of all such different label pairs are shown as *slab-tlab* in Table 3.12. In the second row of the same table, the result of including the PoS tag of the dependent when counting the combinations used, is shown as *dpos-slab-tlab*. The highest degree of parallelism in labelling of dependency relations for tokens is between CD and SB. By this we mean that a relation type for a token in one format in the pair more often has one or few, as opposed to many or several, relation types in the other format. The lowest degree of parallelism is found in the DT/SB pair.

	DT/CD	DT/SB	CD/SB
slab-tlab	819	918	648
dpos-slab-tlab	3591	3269	2537
hpos-slab-tlab	4057	5365	6437

Table 3.12: Counts of different combinations used in each format pair

The third row *hpos-slab-tlab* in Table 3.12 shows the result of attempting to count combinations of head PoS tags and label pairs (as *dpos-slab-tlab* is the count of combinations of dependent PoS tags and label pairs). For every token, we count the combinations of PoS tag and the label of the dependency relations between the token and its dependents in both formats. These dependencies do not necessarily form pairs as was the case when we counted *dpos-slab-tlab*, a dependent always has one and only one head, regardless of the format, but a token can be the head of many dependents in one format and none in the other.

For all tokens, we count the dependency labels from the token to its dependents in one format combined with the dependency labels from the tokens in the other format and the PoS tag of the token. If a sentence has n tokens and none of these tokens are the head of more than one dependent, in any of the two formats, the maximum number of *hpos-slab-tlab* combinations is $n - 1$. If, on the other hand, in both formats, the same token is the head of all the other tokens (except the root) this number is $(n - 1)^2$. The value of *hpos-slab-tlab* tells us something about the share of 'heavy' heads (heads that have many dependents), in the two formats combined. It corresponds to the combined values from the first row of Table 3.6 for the two formats, the CD/SB pair having the highest value and DT/CD the lowest.

There also seems to be an inverse correlation between the values of *dpos-slab-tlab* and *hpos-slab-tlab*. If one is low, the other is high. The CD/SB pair has the highest total of *dpos-slab-tlab* + *hpos-slab-tlab*, for the other two pairs this number is almost identical. This might indicate that conversion between CD and SB is harder than conversion between the other formats.

We examine the same combinations as in Table 3.12, this time only

for matched (unlabelled) dependencies. Matched dependencies are dependencies that have the same dependent and head in both formats. In addition we show the number of different combinations of dependent PoS tags, dependency label in the source and target format and head PoS tag: *dpos-hpos-slab-tlab*. The results are presented in Table 3.13.

	DT/CD	DT/SB	CD/SB
slab-tlab	300	298	270
dpos-slab-tlab	1390	1049	1174
hpos-slab-tlab	1286	1226	1319
dpos-hpos-slab-tlab	3984	3069	3784

Table 3.13: Counts of different combinations used in matched dependencies in each format pair

As we can see from Table 3.13, only a small subset of the combinations represented in Table 3.12 are used in the matched dependencies. The degree of parallelism in labelling of the dependency relations of the matched dependencies is quite similar for all pairs, although best (fewest combinations of labels) for the CD/SB pair. Not surprisingly, there is a correspondence between the sum *spos-slab-tlab* combinations and *hpos-slab-tlab* combinations used and the number of *hpos-dpos-slab-tlab*.

We examine the values for *slab-tlab* and *dpos-slab-tlab* for only the unmatched dependencies as well. The result is shown in Table 3.14.

	DT/CD	DT/SB	CD/SB
slab-tlab	767	880	564
dpos-slab-tlab	3067	2956	1921

Table 3.14: Counts of different combinations used in unmatched dependencies in each format pair

We will make use of some of these combinations later, when we design a baseline labelling procedure in Chapter 4.3.

3.4 Coordination structures

From this section on, we will leave the quantitative approach and look at some well-known areas of differences between dependency formats. One such difference is coordination. In this section we will give a brief introduction to this problematic issue.

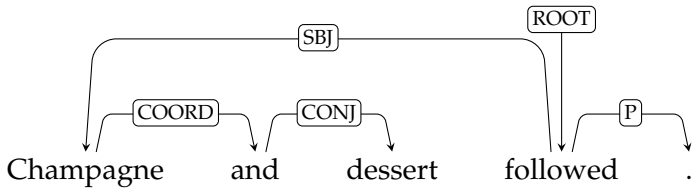
In a coordination structure words, phrases or clauses (*conjuncts*) are connected (most often by a *coordinating conjunction*) in a way that gives them equal importance. Such structures present a challenge to dependency grammars, where the fundamental idea is that all dependency relations are asymmetric. In a coordinated structure none of the conjuncts can

be claimed to be the head of another conjunct. The various dependency schemes use different approaches to this problematic issue.

Popel et al. (2013) offer a systematizing view of the different coordination models. According to them these three main approaches are most commonly used:

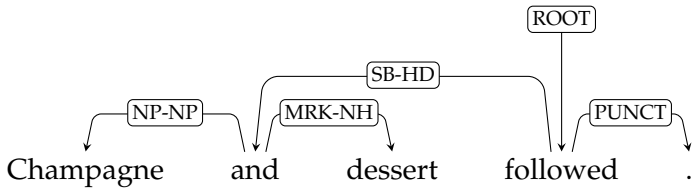
Melčuk style The first conjunct is the head of the coordination structure. The coordinating conjunction is a dependent of the second-to-last conjunct and the last conjunct is a dependent of the coordinating conjunction. If the structure involves more than two conjuncts, the second conjunct is a dependent of the first one, the third is a dependent of the second etc.

CD uses this model for coordination (Johansson 2008). A simple example is shown below.



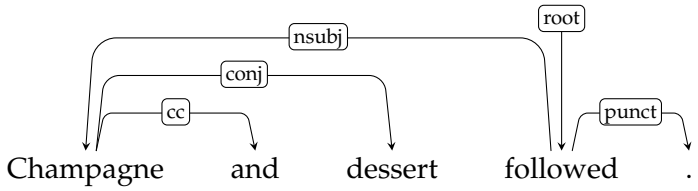
Prague Dependency Treebank style The coordinating conjunction is the head of the coordination structures and all conjuncts are dependents of it.

DT makes use of this model (Ivanova et al. 2012).



Stanford parser style The first conjunct is the head of the coordination structure, and all other conjuncts as well as the coordinating conjunction, are dependents of it.

This is the model used by the SB scheme (De Marneffe et al. 2008).

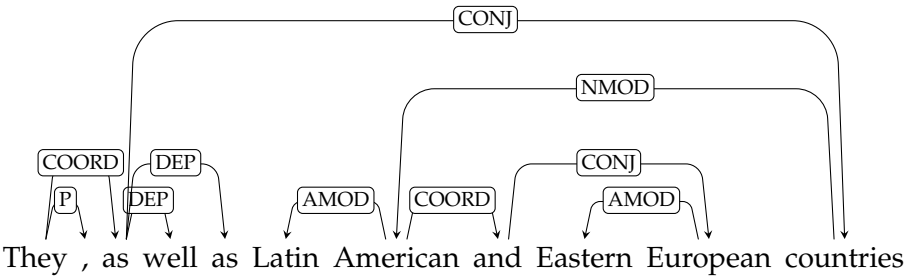


Popel et al. (2013) point at other issues, besides their symmetric nature, that may further complicate the presentation of coordination structures in dependency schemes. Following are some of them, each demonstrated by an example:

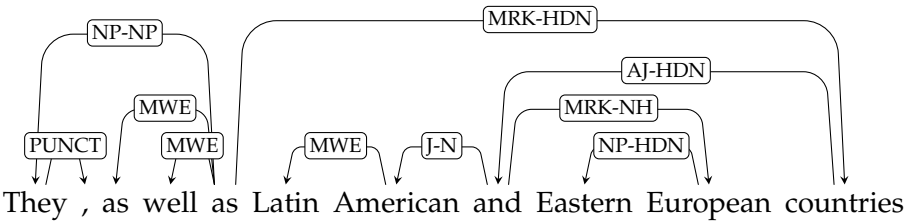
- Multiple conjuncts - *"words, phrases or clauses are connected"*
- Shared modifiers - *"the head of the coordination structure and all other conjuncts"*
- Coordinated modifiers - *"the head and the governor of the coordination structure"*
- Coordinated, shared modifiers - *"the head and the governor of the coordination structure and all other conjuncts"*
- Nested coordinations - *"We split out sections 17 and 18 and use these for development and initial testing"*
- Punctuation marks used instead of coordinating conjunctions - *"the second conjunct is a dependent of the first one, the third is a dependent of the second"*
- Multi-word expressions used as coordinating conjunctions - *"all other conjuncts as well as the coordinating conjunction"*

The construction shown in the three formats in Figure 3.2, contains a multi-word conjunction, a coordinated modifier and nested coordinations:

CD



DT



SB

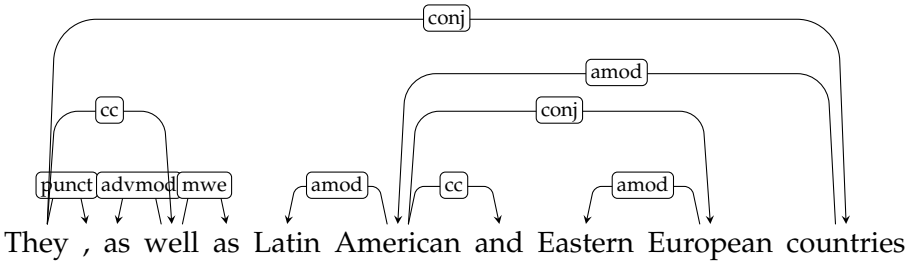
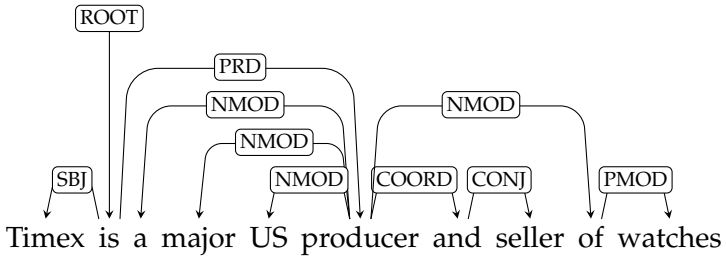


Figure 3.2: Example of a complex coordinated structure in CD, DT and SB

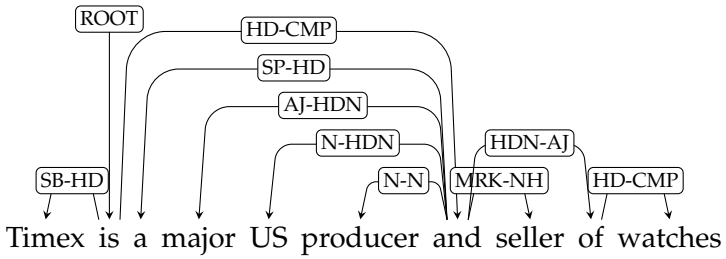
'They' are coordinated with *'numerous Latin American and East European countries'* with *'as well as'* working as a multi-word conjunction. The coordination structure *'Latin American and East European'* is nested inside it and works as a modifier of *'countries'*.

The Prague style is the only model that distinguishes between a local modifier of the first conjunct and a modifier that modifies the whole coordination structure. This is illustrated by the examples in Figure 3.3.

CD



DT



SB

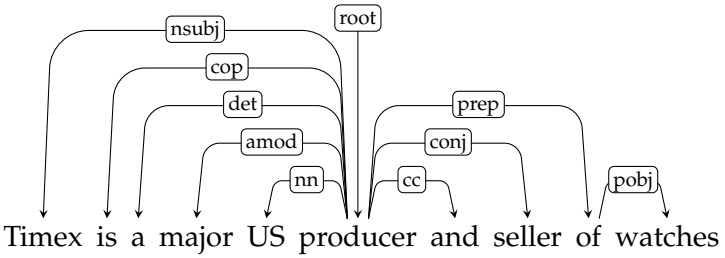


Figure 3.3: Example of a coordinated structure with shared modifiers in CD, DT and SB

In this case both *'major'* and *'US'* are meant to modify the whole coordination structure *'producer and seller'*. This is expressed in the DT format by attaching them to the conjunction, which is the head of the coordination structure. The shared modifier is attached in the same way, to the head of the coordination structure in CD and SB, but because this head is the first conjunct in this formats, it is impossible to decide whether it is a local modifier of this conjunct only, or meant to modify the complete structure. This phenomenon implies that DT, in this respect, is more fine-grained than the other two formats and that conversion of coordination structures from DT to the other formats will be lossy in some cases.

3.5 Non-projective dependencies in CD

Projectivity (see Section 2.1) is another well-documented area where dependency schemes differ. In CD non-projective dependencies have been introduced in order to handle discontinuous structures (Johansson 2008). When CD is generated, the conversion makes use of traces in the original, full PTB annotation, to achieve this. Surdeanu et al. (2008) observe that about 42% of the non-projective dependencies are caused by wh-movement, 18% by split clauses, 14% by split noun phrases and the remaining 26% by other phenomena. DT is, like CD, generated by conversion from constituents to bi-lexical dependencies, but without the use of function tags and traces, and will not contain any non-projective dependencies. For similar reasons, the SB format as well, only contains projective structures (De Marneffe et al. 2008).

Our CD training data contains 3715 non-projective dependencies in 2010 sentences. This means that 0.57 percent of the dependencies in our training data are non-projective, and that 6.77 percent of the sentences contain one or more non-projective dependencies. We extract some examples of non-projective structures. Figure 3.4 shows an example of wh-movement. In CD wh-words are attached to their semantic heads which in this case is the verb *be*. In DT, in contrast, *'how'* is attached to the modal verb *'could'*, which is also the root. In SB the copula verb *'be'* is the root of the sentence and the punctuation mark is attached to it.

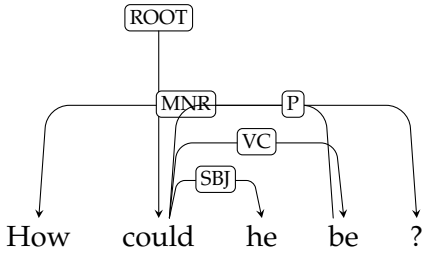


Figure 3.4: Example of non-projectivity caused by wh-movement

Figure 3.5 illustrates non-projectivity caused by a split clause. In DT projectivity is ensured by attaching *'Futures'* to the root *'says'*. In SB *'cut'* is considered the root of the sentence.

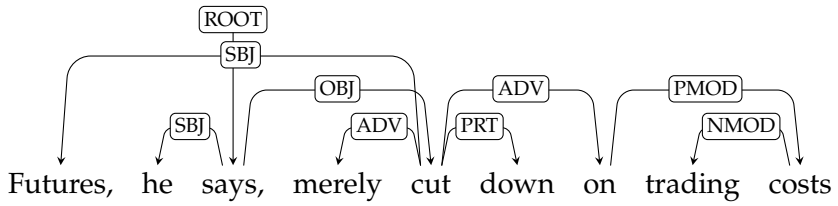


Figure 3.5: Example of non-projectivity caused by split clause

Figure 3.6 shows an example of a split noun phrase. In DT both prepositional phrases are attached to *'sales'* by a dependency relation labelled *HDN-AJ*. In SB they are attached to *'rang'* by a *prepositional modifier* relation.

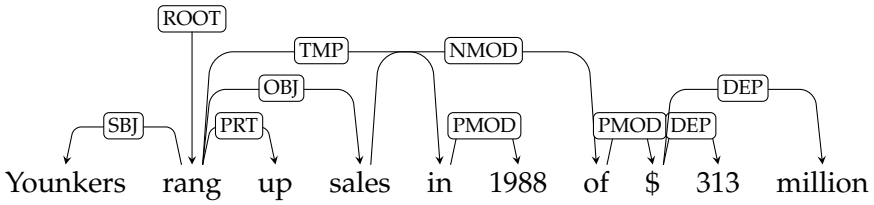


Figure 3.6: Example of non-projectivity caused by split noun phrase

3.6 Summary of format comparison

Beneath follows a list of tentative conclusions we have drawn and assumptions we have made with regard to granularity, similarity and convertibility based on the comparisons we have made of the three formats.

1. CD has more available combinations of PoS tags and dependency labels, than the other two formats, and thus, a higher degree of granularity
2. Similar numbers of combinations of dependent PoS tags, dependency labels and head PoS tags used for all formats suggest similar variability.
3. SB has a higher variation of PoS tags for tokens used as roots. This might suggest higher variability in this format.
4. The SB format has a lower tree-depth and uses fewer tokens as head (each head has more dependents attached to it), than the other formats. This indicates a lower degree of structural similarity between SB and the other formats.
5. The DT/CD and CD/SB pairs have the highest unlabelled attachment score, and thus the highest correspondence of syntactic structures.
6. The DT/CD pair has the highest unlabelled attachment score when punctuation marks are excluded.
7. The DT/CD pair has the highest share of identical roots.
8. Identification of correct root will be very important when converting to CD or SB.
9. The CD/SB pair has the highest sum of dependent PoS tag, source label, target label combinations and source label, target label, head PoS tag combinations. This might indicate a higher complexity in conversion between CD and SB.
10. DT uses the most expressive annotation for coordination structures. This indicates that successful conversion of such structures to the DT format will be challenging.
11. Non-projectivity in CD might complicate conversion including this format, but the share of such dependencies is small.

Based on these conclusions, we will make a decision on which format pair and in which direction we will attempt conversion in the rest of the study. DT/CD and CD/SB are the most similar of the format pairs. The two pairs have a similar unlabelled attachment score, but there are indications of a closer relationship between DT and CD than between CD and SB. The similarity in tree-depth, the high share of identical roots and the high unlabelled attachment score when punctuation tokens are excluded all suggest a higher degree of correspondence between DT and CD. On this background, we choose to use the DT/CD pair for our conversion experiment.

When it comes to direction (should we convert from DT to CD or from CD to DT?), the decision is less obvious. Similar variability suggests that

conversion in both directions is equally likely to be successful. A more expressive annotation for coordination structures in DT, might complicate conversion to this format. The use of non-projective structures in CD, on the other hand, could imply that CD is not the easiest target format. We decide to attempt a conversion to CD, the most well-established format. Chapters 5 and 6 will be devoted to the experiment of conversion from DT to CD. In the next chapter, we will take a closer look at how conversion between dependency formats can be performed.

Chapter 4

How can we perform conversion?

In this chapter we will take a closer look at how conversion between dependency formats can be performed. We will discuss briefly what considerations have to be made and possible approaches to this task. We will discuss subtasks involved in conversion and ways of handling these in Section 4.1. In Section 4.2 we will describe a heuristic methodology for detection of candidate patterns for rewriting. A baseline converter with a heuristic labelling procedure is described in Section 4.3.

4.1 Conversion of syntactic structures

In dependency grammars syntactic structures are expressed by dependencies, binary asymmetrical relations between the words of the sentence. A dependency consists of two words linked by a labelled, directed arc. The direction of the arc expresses the head–dependent relation. It denotes which of the words is considered the head and which is considered the dependent in the structure (see Section 2.1). The label of the arc describes the type of the relationship between the head and the dependent. Every dependency format has its own set of arc labels. These labels are syntactic in most dependency schemes, expressing grammatical functions. In some formats though, they have a more semantic nature, expressing semantic roles. According to Ivanova et al. (2012) labels are mostly syntactic in the dependency schemes that we have included in this study:

CD: labels are mostly syntactic, but also express a few semantic distinctions (like different types of adverbial modification)

DT: labels are syntactic, they are identifiers of HPSG constructions (e.g. subject-head, specifier-head)

SB: labels are largely based on syntactic functions

The labels *SBJ* (subject) in CD, *SB-HD* (head + subject) in DT and *nsubj* (nominal subject) in SB are examples of labels with a merely syntactic

content. These labels denote that the dependent has the syntactic function of subject, but they say nothing about semantic role of the dependent. The *TMP* (temporal adverbial or nominal modifier) label is an example of a label in the CD format that contains additional semantic information, as opposed to the *HD-AJ* (head + following adjunct) and *AJ-HD* (head + preceding adjunct) labels and the *prep* (prepositional modifier) and *advmod* (adverbial modifier) labels often used to name these relations in the DT and SB format, respectively.

When converting dependency structures from one format to another, we will need both to reattach dependents that are assigned 'incorrect' heads according to the target format and to relabel all dependency relations with the label set used in the target format. This can be performed as a two-step procedure:

1. For all nodes (words): identify the correct head in the target format and reattach those that are differently attached in the source format

2. For all dependency relations (arcs): label with the correct label from the set of labels used by the target format

We will take a closer look at how we can perform the first step. In dependency grammars syntactic structures are represented as labelled directed graphs, that are connected, acyclic and single-headed (see Section 2.1). This means that every node (word) is a dependent of one and only one other word, its head. The choice of head can vary in different formats. This means that the number of dependents in a sentence will be the same in all dependency formats (we assume equal tokenization), but the number of heads can vary. Rewriting a syntactic structure implies reattachment of a node to a different head. This might lead to a change in the number of heads in the sentence. In order to assure that the structure still adheres to the constraints of connectedness, acyclicity, single-headedness and, in cases where this constraint is present, projectivity, reattachment of other nodes as well is often necessary. Figures 4.1– 4.3 illustrate rewriting of a dependency structure from CD to SB annotation.

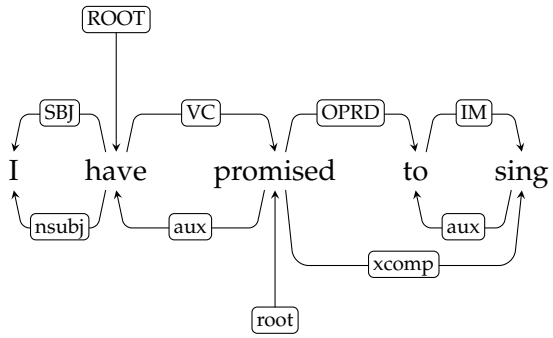


Figure 4.1: Dependency representation in CD (above) and SB (below) format.

We have discovered that in a construction like *'have promised'* SB will consider the main verb *'promised'* as head, while CD chooses the auxiliary *'have'* as head, and we want to do a rewriting according to this *pattern*. Such patterns will be the starting point for the rewriting rules that we implement later in this project.

If we only inverse the direction of the arc between *'have'* and *'promised'*, the graph will end up violating the constraints of connectedness (*'promised'* will not be attached to any other node) and single-headedness (*'have'* will have two heads). See Figure 4.2.

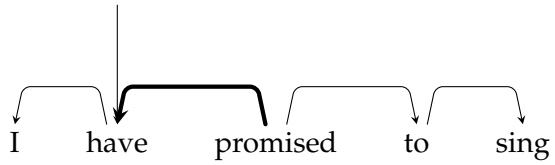


Figure 4.2: Dependency representation in CD after reattachment of a single node

We will also have to reattach *'promised'* to the head of its former head, in this case the root, in order to make the graph well-formed. See Figure 4.3.

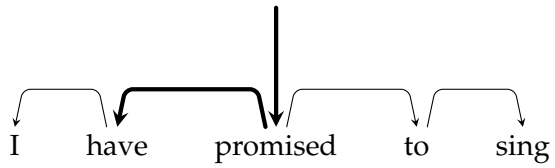


Figure 4.3: Dependency representation in CD after rewriting of a syntactic structure

Conversion of a syntactic structure will in most cases involve rewriting of a sub-tree, not just a single dependency. In this case we might want to rewrite the sub-tree for the complete verb phrase *'have promised to sing'*. This will also be the case with conversion of, for instance, coordination structures (see Section 3.4) from one annotation style to another.

The starting points or *triggers* for rewriting of structures are identified patterns of differences in the two formats. We need to identify the sub-trees in the source format that are incorrect according to the target format, and the correct structures for these sub-trees in the target format. Such patterns often correspond to a linguistic phenomenon. We envisage three different approaches to identification of such patterns:

- Use of linguistic theory and existing documentation on the different formats
- Use of quantitative analysis

- Use of machine-learning

The use of machine-learning for this purpose, is not a straightforward issue. We have no knowledge of any existing implementation of a machine-learning tool for rewriting of dependency trees from one format to another, and it is not evident how a machine-learner could be trained for this task. Some of the purpose of our project is to gain better knowledge of the differences between our selected dependency formats. In this respect as well, choosing an approach involving machine-learning would be problematic, as machine-learners rarely output any of their gained knowledge explicitly.

Some related work has, however, been performed. Graham et al. (2009) have developed a tool for automatic induction of transfer rules used for machine translation of natural languages. The rules induction procedure uses pairs of bilingual corpora, consisting of dependency structures, as input. The induced transfer rules are stored as data structures in files and used for transferring dependency structures from source language structure to target language structure. It is possible that this tool or a similar approach could have been used for rewriting dependency structures from one annotation format to another.

We have not explored this option any further, but chosen a heuristic-analytic approach for detection of patterns for rewriting. As existing documentation on differences between the selected formats is rather sparse, we will have to use a quantitative methodology. In the next section we propose such a methodology.

4.2 Methodology

Are discontinuous structures, coordination structures and punctuation marks the only phenomena treated differently in CD and DT? Are there other differences and how can we reveal them? We will make an attempt to discover systematic differences by use of quantitative analysis. We have already counted matched dependencies (dependencies that have the same head and dependent) and unmatched dependencies (dependencies with different heads in the two formats) per dependent PoS tag, source format label, target format label, head PoS tag in source format and, for unmatched dependencies, also PoS tag for head in target format (see Section 3.3).

We will compare these data to see if we can find combinations of PoS tags and labels where the number of matched and unmatched dependencies differ in a way that can imply a systematic difference between the formats. Subsequently, we will extract sample sentences that are part of these combinations and see if we can discover patterns of differences between the formats. Finally, we rewrite the sentences in the source format to the pattern identified in the target format, and evaluate the result.

The sum of matched and unmatched dependents with a certain PoS tag will always be constant. If the number of unmatched dependents

with this PoS tag is reduced, the number of matched dependents with this PoS tag will increase correspondingly. This is not the case for the sum of matched and unmatched dependents per head PoS tag. The number of matched heads with a certain PoS tag only tells us that this number of dependents are correctly attached to a node with this PoS tag. The number of unmatched heads with a certain PoS tag tells us the quantity of dependents incorrectly attached to nodes with this PoS tag. If the number of unmatched heads with a PoS tag is decreased, the number of matched heads with this PoS tag is not necessarily affected. In cases (and this will probably be most frequent) where the matched amount is unaffected, the nodes formerly incorrectly attached to nodes with this PoS tag, have been (correctly or incorrectly) reattached to nodes with other PoS tags. In cases where the number of matched heads with this PoS tag is increased, some of the formerly incorrectly attached dependents have been successfully reattached to other nodes with this PoS tag. If, on the other hand, the number of matched heads is decreased, some of the dependents already correctly attached to heads with this PoS tag, have been erroneously reattached.

Combinations of PoS tags and labels where the number of unmatched dependencies is large, relative to the total numbers of dependencies, indicate that there is a potential for discovering essential systematic differences and successfully converting a large group of dependencies. Combinations with a small number, the optimum value being zero, of matched dependencies, imply that the risk of damaging already corresponding dependency structures in the rewriting process is at a minimum.

These combinations should be worth looking into when identifying candidates for conversion:

- The number of unmatched dependencies is large and the number of matched dependencies is small. These combinations are the ones that suggest the largest potential for successful rewriting.

- The number of unmatched dependencies is large, but the number of matched dependencies is also of a considerable size, maybe even larger than the number of unmatched dependencies. There is a potential for discovering patterns involving a large quantity of dependencies, but these patterns can be hard to identify and the risk of rewriting already correct dependencies is high.

- The number of matched dependencies is low, approximating zero, but the number of unmatched dependencies is not very large. This indicates that the risk of erroneously reattaching already correctly attached dependents in our rewriting procedure is small, but the potential gain of rewriting structures involving these dependents is not necessarily very large.

Table 4.1 illustrates some examples. DT is the source format and CD the target format. The first and second row contains combinations

involving a considerable amount of unmatched dependencies and a rather low amount of matched dependencies. The first row shows that nearly all of the dependents attached by an arc labelled *NUM-N* in the DT format, are differently attached in the CD format. If we can reattach these 6682 dependencies correctly, we will increase the unlabelled attachment score by at least 1.0. The second row shows that more than 93 percent of the nodes PoS-tagged *CC (coordinating conjunction)*, are incorrectly attached according to the CD scheme. Rewriting them implies the risk of incorrect reattachment of the 1029 *CC* nodes that are already correctly attached. Still, rewriting these dependencies could lead to an increase of more than 2.0 of the unlabelled attachment score. The third row shows a combination with a very large number of unmatched dependencies, but also a considerable amount of matched dependencies. Although 20617 dependencies are erroneously attached to a *CC* head, the considerable amount of 12162 dependencies are correctly attached to a *CC* head. In this case we need to do a careful analysis of the dependencies involved, to be able to do a successful conversion. The last two rows identifiy a very specific pattern. Not only do they show that *all CD (cardinal number)* nodes attached by a *SP-HD* dependency to another *CD* node are incorrectly attached according to the CD scheme, they also show that the PoS tag of the correct head in the CD format is *$ (dollar sign)*. All we need to do is to identify the correct *$* node and reattach the *CD* node to it. We can estimate the gain of rewriting these structures to 0.5, undoubtedly worth the effort, especially when we consider that the risk of creating errors is minimal.

dpos	slabel	shpos	thpos	matched	unmatched
	NUM-N			32	6682
CC				1029	14457
		CC		12162	20617
CD	SP-HD	CD		0	3411
CD	SP-HD	CD	$	0	3411

Table 4.1: Combinations with large amount of unmatched dependencies and/or low amount of matched dependencies

4.3 Baseline system

We will construct a baseline converter for our format pairs. A baseline system is usually an implementation of the simplest possible algorithm, and is defining a performance to be improved upon by our heuristic conversion procedures. When constructing our baseline converter, we assume identical attachment in the source and target format of all nodes (although we know from our earlier investigations that this is not true). Thus, our baseline conversion does not include any rewriting of syntactic structures.

For labelling, we will derive a simple heuristic procedure, where we

label the dependency relations one by one. For this purpose, we will make use of the data we extracted for the results presented in Table 3.13 and Table 3.14. We will construct a set of labelling rules consisting of a trigger from the source format as input and a suggested label from the target format as output. The triggers that we use are comprised of different combinations of PoS tags and source format labels. We need different types of trigger patterns to be able to handle combinations of PoS tags and labels that occur after rewriting, but have not been found in the original DT dataset. The triggers are explained in Table 4.2.

Trigger	Explanation
dpos-hpos-label	PoS tag of dependent + PoS tag of head + label from source format
dpos-label	PoS tag of dependent + label from source format
hpos-label	PoS tag of head + label from source format
label	label from source format

Table 4.2: Trigger types used for labelling

The output label produced by use of these triggers, is the label most frequently used in the target format for matched dependencies (Table 3.13) where the source format dependency corresponds to the trigger value. Some examples of labelling rules are shown in Table 4.3.

Trigger values dpos+hpos+label	proposed target format label
NN + SP-HD + VB	SBJ
+ SP-HD + VBZ	SBJ
VBP + FLR-HD +	OBJ
+ FLR-HD +	SBJ

Table 4.3: Examples of labelling rules

For source labels that are not found among the matched dependencies, we will use the most frequent target label used for unmatched dependencies (Table 3.14) with this source label. We need these last rules to be able to label all unmatched dependencies as well as the matched dependencies. These rules will, however, have no effect on the performance of our converter, as the labelled attachments score obviously measures the accuracy of labelling only for the matched dependencies. The number of rules for each format pair is shown in Table 4.4.

	CD	DT	SB
CD		3490	3430
DT	2700		2571
SB	3062	2704	

Table 4.4: Number of rules used in baseline converter. Row: source format, column: target format.

We will apply these rules in an order from most to least specific. If the most specific trigger value for a dependency is not found, we attempt to use the trigger one step lower in the hierarchy etc. Although the order of specificity for the *dpos-label* and the *hpos-label* is not given, as we will observe later in this section, we will assume this order:

1. dpos-hpos-label
2. dpos-label
3. hpos-label
4. label

The evaluation results, on our training data, for this converter is represented in Table 4.5. We can see that the nnlabelled attachment score corresponds to the result in Table 3.8.

	Labelled attachment score			Unlabelled attacment score		
	CD	DT	SB	CD	DT	SB
CD		66.9	70.6		73.6	72.7
DT	66.6		54.3	73.6		55.7
SB	66.8	49.6		72.7	55.7	

Table 4.5: Evaluation results for baseline converter used on training data. Row: source format, column: target format.

We use the converter on the development data. These data have not been used earlier in this project, neither when we designed our labelling rules or when we investigated similarities between format pairs. Using our converter on these earlier unseen data, will give us an indication of its performance on any other unknown dataset as well. The evaluation results are represented in Table 4.6.

Reversing the order of the rules so that the *hpos-slabel* trigger is used before the *dpos-slabel* trigger, or omitting the use of these rules altogether results in exactly the same performance as shown in Table 4.5 for our training data. For development data, reversing the order of these rules or omitting them, has a minimal effect, up to 0.03 change in labelled attachment score for some pairs. If we omit the use of the rule using the

	Labelled attachment score			Unlabelled attacment score		
	CD	DT	SB	CD	DT	SB
CD		66.5	70.9		72.9	73.0
DT	65.8		54.2	72.9		55.6
SB	66.8	49.6		73.0	55.6	

Table 4.6: Evaluation results for baseline converter used on development data. Row: source format, column: target format.

dpos-hpos-slabel trigger, however, the order in which we use the *dpos-slabel* and the *hpos-slabel* trigger has an effect. See Table 4.7 for the result on development data.

	dpos-label first			hpos-label first		
	CD	DT	SB	CD	DT	SB
CD		64.9	70.6		54.4	52.0
DT	57.6		50.5	58.3		47.2
SB	62.9	46.2		66.2	48.3	

Table 4.7: Labelled attachment score for conversion with different rule order. Row: source format, column: target format.

We can see that when converting from SB to any pair and from DT to CD, using the *hpos-slabel* trigger first gives a better result. Using only the *label* triggers, gives a poorer performance for all format pairs and directions, with the highest score being 61.6 for SB to CD and the lowest being 41.7 for DT to SB.

When converting CD development data to SB or DT, one of the dependencies ended up not labelled. This happens because our development data contains a dependency labelled with a label that does not occur in out training data.

We believe that this, rather simple, heuristic labelling procedure, could be improved by making use of more of the information available. One such approach could be using some kind of thresholding, choosing the most common target label for a trigger only when this target label is correct for a minimum of n% of the dependencies with the trigger value in question. Another approach could be to consider word forms, PoS tags of children and siblings, direction of arcs or other available information as trigger values. In Chapter 6 we will explore the use of more information to improve the task of labelling.

Chapter 5

Heuristic conversion from DT to CD

In this chapter we will describe our work on heuristic conversion of data annotated in the DELPH-IN Syntactic Derivation Tree (DT) format to CoNLL Syntactic Dependencies (CD). This is the format pair and the direction of conversion that we decided upon in Section 3.6. As we have discussed in Section 4.1, conversion will consist of three main subtasks: identification of patterns for conversion, rewriting of syntactic structure, and labelling, i.e. rewriting of dependency types. Adaptation and extension of our baseline labelling procedure, introduced in Section 4.3, for our conversion task, is described in Section 5.1. In Section 5.2 we will describe how we will use a methodology, introduced in Section 4.2, for identification of candidate patterns for rewriting. Sections 5.3- 5.9 contains descriptions of each of the patterns, how the rewriting is performed and the results of it. In Section 5.10 reattachment of punctuation token dependents is described. In Section 5.11 we discuss unsuccessful rewritings. Finally, in Section 5.12, we give a summary of the end-to-end result of our heuristic conversion.

5.1 Extended labelling procedure

Earlier we have described syntactic rewriting and labelling as tasks performed separately and consecutively. In our conversion procedure, this will not be fully the case. Although we most frequently will perform no changes to source format labels during the rewriting process, there will be some cases where we want to deviate from this practice. Some times the correct dependency label will be identified, from the pattern that we rewrite, at the time of rewriting already. In these cases we might want to label the dependency during the rewriting procedure. In other cases, we will just remove the source format label in the rewriting process and leave the assignment of target format label to the subsequent labelling procedure. This will be the case when we want a structure to be processed only once, by procedures rewriting structures with this source label.

When converting the development data set from the CD format with

our baseline converter (see Section 4.3), one of the dependencies ended up unlabelled. Our simple labelling procedure will fail to label dependencies not caught by any of our labelling triggers, as will be the case if we encounter source format labels that are not found in our training data. We need to expand our labelling procedure to handle these instances. We add new rules containing triggers of dependent PoS tags and head PoS tags, *dpos-hpos*, to our labelling rule set. These rules will be used for labelling of dependencies where we have removed the source format label. The target format label suggested for dependencies corresponding to such a trigger, is the most frequent dependency label used in CD for matched dependencies having this dependent and head PoS tag. We also add a rule that will assign the NMOD label, the most frequent dependency type found in our CD training data, to dependencies not caught by any of our triggers. In addition, we adjust our labelling procedure so that it does not override any labelling already performed by the rewriting procedure.

We now have a total of 3456 labelling rules (see examples in Table 4.3). We will apply the new *dpos-hpos* rule after the most specific rule, *dpos-hpos-label*. The other rules will be applied in the order indicated as most appropriate, for this format pair and this direction, by our investigations in Section 4.3:

1. dpos-hpos-slabel
2. dpos-hpos
3. hpos-slabel
4. dpos-slabel
5. slabel

Applying this labelling procedure on DT data converted to CD by our baseline converter, gives the same labelled attachment score of 66.6 for the training data and 65.8 for the development data as obtained when using the baseline labelling procedure described in Section 4.3.

5.2 Identifying candidates for rewriting

The contrastive study of different dependency formats, conducted by (Ivanova et al. 2012), indicates a great extent of similarity between the CD and the DT format. Both formats adhere to the principles of connectedness, acyclicity and single-headedness. In both formats functional elements are chosen as heads. Examples of this are that auxiliaries are heads of the main verb in constructions with auxiliaries and infinitive markers are heads of the main verb in constructions with infinitive verbs. Another similarity is found in the treatment of noun phrases: in both formats the noun is considered the head of the determiner and attributive adjectives. The study documents the different treatment of coordination structures in the two formats, earlier discussed in Section 3.4. Another area where

the two formats differ is with respect to projectivity (see Section 3.5). In CD non-projective dependencies have been introduced in order to handle discontinuous structures (Johansson 2008). In DT we know that all dependencies are projective. Because CD and DT come from different sources, the source of CD is the PTB while DT origins from DeepBank, it would not be surprising if they prove to differ in the linguistic analysis of other phenomena than coordination.

We will use the methodology described in Section 4.2 to reveal patterns of systematic differences between DT and CD, that we consider to be candidates for rewriting[1]. As recommended in Section 4.2, we will try to identify dependent PoS tags, source format head PoS tags, source format labels, or combinations of these, that have a large amount of unmatched dependencies and a small number of matched dependencies, a large amount of unmatched dependencies but also a considerable amount of matched dependencies, or a very small number of matched dependencies and a considerable amount of unmatched dependencies. Initially, we will only consider patterns that do not involve punctuation mark dependents. Attachment of punctuation tokens will be treated separately in Section 5.10. In our rewriting we will concentrate on investigations of structures that constitute the unmatched dependencies with the dependent PoS tags, source format labels and/or head PoS tags shown Table 5.1.

dpos	slabel	shpos	matched	unmatched
CC			1029	14457
		CC	12125	20373
POS			43	5868
		POS	12	5997
	NUM-N		32	6682
CD			11317	12818
		CD	2010	15143
		DT	707	2398

Table 5.1: Patterns with low unlabelled attachment score (excl. punctuation marks)

The large number of unmatched dependencies involving CC (coordinating conjunction) heads and/or CC dependents, we believe are caused by the use of different coordination models in the two schemes. These structures will be thoroughly examined in Section 5.3 and 5.4. The groups of unmatched dependencies involving POS (possessive ending) heads and/or POS dependents will be discussed in Section 5.5. In Section 5.6 we examine the unmatched dependencies involving CD nodes (cardinal numbers) and currency PoS tags. In Section 5.7 unmatched dependencies labelled NUM-N (measure NP from number + noun) are examined. Some other

[1]In the sample extraction part of this procedure, we have had the advantage of access to a prototype for a search interface for the three dependency formats (Kouylekov et al. 2014). The interface is available at: http://wesearch.delph-in.net/dt/

constructions with CD heads are discussed in Section 5.8. Dependencies with DT (determiner) heads are treated in Section 5.9.

5.3 Coordination

Table 5.2 shows that we have initially 14457 CC dependents that are differently attached in DT and CD. It also shows that 20373 tokens that are assigned CC heads in DT, are attached to other nodes in CD.

dpos	slabel	shpos	matched	unmatched
CC			1029	14457
		CC	12125	20373

Table 5.2: Patterns of unmatched dependencies involving coordinating conjunctions

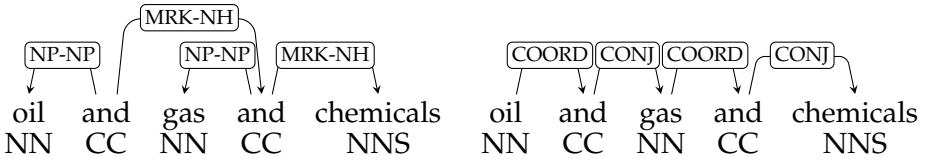
We mentioned in Section 3.4 that coordinated elements, conjuncts, can be clauses, phrases or words. We showed an example of a very simple coordination structure, where two nouns are coordinated using the conjunction *and*. This type of coordination structure could be described using the formula:

- conjunct1 conjunction1 conjunct2

We have also seen that coordination structures with multiple conjuncts are common, often with the use of commas or semicolons as conjunctions.

- conjunct1 conjunction1 conjunct2 ..... conjunction$n$–1 conjunct$n$

Example (DT left, CD right):



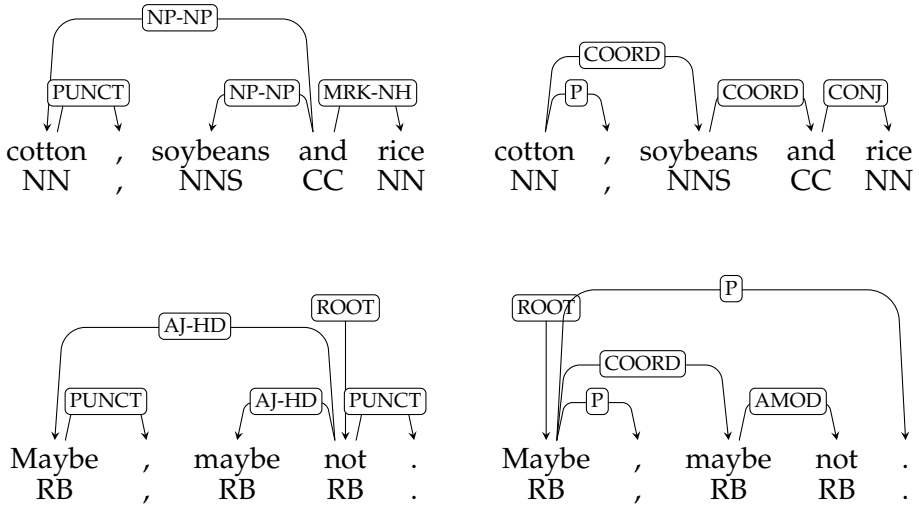
- conjunct1 , conjunct2 ..... conjunction1 conjunct$n$

Example (DT left, CD right):
Sometimes a serial comma (Oxford comma) occurs before the coordinating conjunction.

- conjunct1 , conjunct2 ..... , conjunction1 conjunct$n$

We have no occurrences of this in our training data.

Sometimes a coordinating conjunction is lacking altogether.



- conjunct1 , conjunct2

Example (DT left, CD right):

As described in Section 3.4, DT and CD use different models for annotation of coordination structures. DT makes use of the Prague Dependency Treebank style, while CD applies Mel'čuk style.

Popel et al. (2013) have identified some topological variations within and across these three main models.

- Choice of head

  In DT the rightmost conjunction is the head of the coordination structure. In CD the leftmost conjunct is chosen as head.

- Attachment of shared modifiers

  In both formats shared modifiers are attached to the head of the coordination structure. DT attaches shared modifiers to the coordinating conjunction, in CD they are attached to the leftmost conjunct.

- Attachment of punctuation tokens separating conjuncts

  In CD separating punctuations are attached to the previous conjunct. In DT they are attached to the preceding word, regardless of its function.

The example shown in Figure 5.1 illustrates these phenomena. In this example the verb phrases *'gave up ..'*, *'walked ..'* and *'did ..'* are coordinated. The rightmost conjunction *'and'* is chosen as head in the DT format. In CD the leftmost conjunct *'gave'* is assigned head of the coordinating construction. In this example *'He'* works as a shared modifier for *'gave'*, *'walked'* and *'did'*. In both formats it is attached to the head of the coordinated structure that it modifies. A coordinated, shared modifier

will be attached by its head to the modified element. The separating punctuation token ',' is attached to the preceding word *'hits'* in DT. In CD this punctuation token is attached to the head of the previous conjunct *'gave'*.

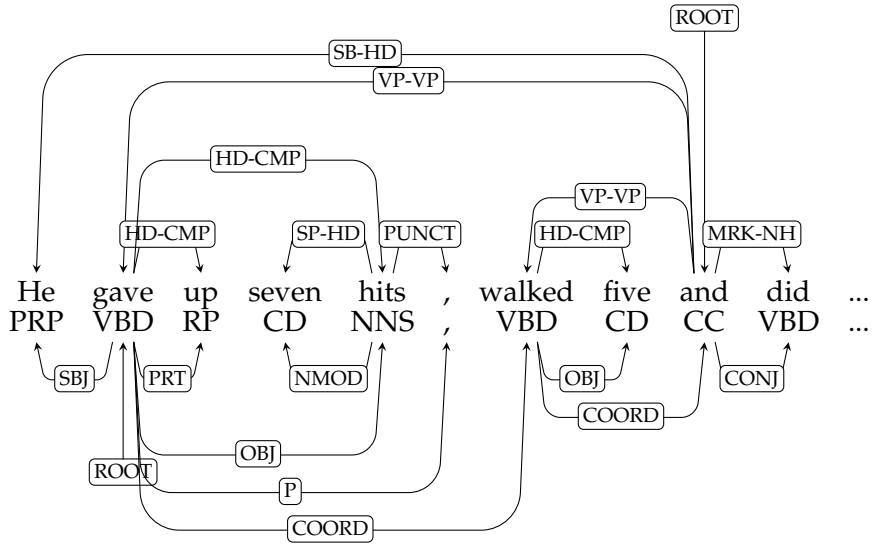


Figure 5.1: A coordination structure illustrating choice of head, attachment of shared modifiers and attachment of separating punctuation tokens (DT dependencies above, CD below)

Interestingly, we see from Table 5.2 that 1029 CC nodes are already equally attached in DT and CD. Considering the different models used for annotation of coordination structures in the two formats, it is not obvious to us what these can be. A closer examination reveals that 830 of these CC nodes are attached to their head by a dependency relation labelled N-N. An example is shown in Figure 5.2 (DT left, CD right). This is a divergence from the Prague Dependency Treebank style. We have brought this to the notice of the team working with DeepBank and DT, who detected a bug in the generation of DT. This bug has been corrected and these structures will adhere to the Prague Dependency Treebank style in the next version of DT.

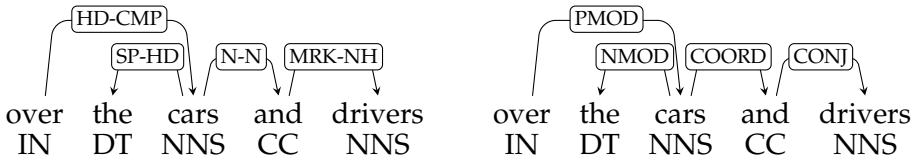


Figure 5.2: Coordination structure with deviant annotation in DT (DT left, CD right)

5.3.1 Identifying coordination structures

How can we identify coordination structures? If a one-word coordinating conjunction is present, it should be PoS-tagged with the CC PoS tag. Table 5.3 shows an overview of the CC nodes occurring in our training data and their frequencies.

Word form	Frequency	Word form	Frequency
and	10587	so	6
but	2471	whether	5
or	1649	v.	3
&	601	times	3
nor	46	less	3
either	37	versus	2
yet	26	minus	1
both	17	vs.	1
neither	14	et	1
plus	13		

Table 5.3: Words PoS-tagged CC and their frequency in WSJ Section 00–17

One of the issues that Popel et al. (2013) pointed to in their study of coordination structures was the phenomena of multi-word conjunctions. A multi-word conjunction is a multi-word expression, such as *'as well as'*, used as a coordinating conjunction. An example of coordination structure where this multi-word conjunction is shown in Figure 3.2). Investigating our data, we find that *'as well as'* is PoS-tagged as RB RB IN. The last *'as'*, tagged IN, is the head of the multi-word structure in DT. The other two words are attached to it, with dependency relations labelled MWE. In CD the head of this multi-word conjunction is the first *'as'* and the other two words are dependents of it, dependencies labelled DEP.

We envisage two different approaches to detection of multi-word conjunctions. One is to run through the DT training data set and extract all non-CC heads of dependents attached by an arc labelled MRK-NH, and their other dependents. The other approach is to do the same using the CD data and check for dependents attached by a CONJ arc to a non-CC head. We decide to use the latter method. The result is shown in Table 5.4.

Multi-word expression			Frequency
as RB	well RB	as IN	75
rather RB	than IN		38
instead RB	of IN		12
if IN	not RB		6
not RB	to TO	mention VB	2
in IN	addition NN	to TO	1
not RB	just RB		1

Table 5.4: Multi-word conjunctions in CD with PoS tags and frequency

The three most frequent multi-word conjunctions occurring in our training data, seem to have the same pattern; a IN head DT and the (first) RB functioning as head in CD. Examples are shown in Figure 5.3 (DT left, CD right). We can easily rewrite the internal structure of these expressions and reattach their head and dependents. This should be done before we convert the coordination structures. The multi-word conjunction *'if not'* are represented differently in DT, with the RB tagged *'not'* functioning as head. This already is in accordance with the CD scheme.

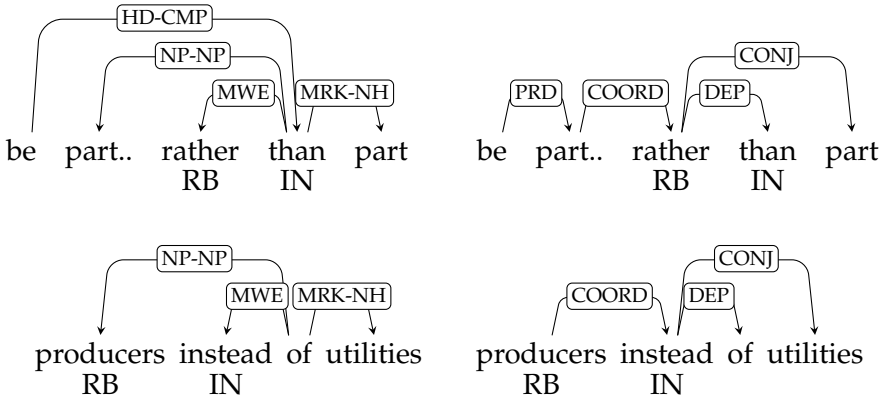


Figure 5.3: Examples of coordinated structures with multi-word conjunctions (DT left, CD right)

DT has 15 dependency labels used for dependency relations for conjuncts (other than the last conjunct) (*ERG Lexical and Syntactic Rules*). These are shown in Table 5.5. The last conjunct in coordinated structures in DT is always attached to the conjunction with a dependency

relation labelled MRK-NH. This token is already correctly attached to the conjunction. In DT other right dependents of the conjunction occur frequently. It makes sense to reattach them to the new head of the coordination structure. This may be incorrect, in cases where these should not be dependents of the coordination structure at all in CD, but we will not address this issue now.

Label	Description
vp-vp	conjoined verb phrase
v-v	conjoined verb
cl-cl	conjoined clause
pp-pp	conjoined prepositional phrase
r-r	conjoined adjective phrase
np-np	conjoined noun phrase
n-n	conjoined noun
n-j	conjoined noun + adjective
j-n	conjoined adjective + noun
j-j	conjoined adjective
jpr-jpr	
jpr-vpr	
vppr-vppr	
vpr-vpr	
ppr-ppr	

Table 5.5: Dependency labels used for conjoined elements in DT

Coordination structures without conjunctions are harder to identify. These structures are not annotated as coordinations in DT, but some of them are annotated as so-called run-on clauses. Figure 5.4 shows an example (DT dependencies above, CD below). Our DT training data set contains 3090 run-on clauses, while we identify 431 conjunction-lacking coordination structures in our CD training data. We will not perform any rewriting of these structures or a more thorough investigation of them.

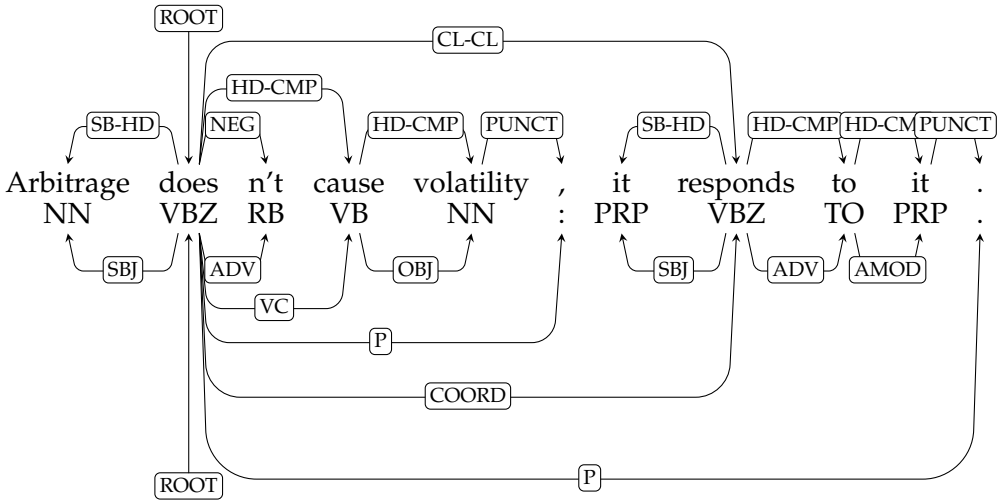


Figure 5.4: Example of coordination structure lacking coordinating conjunction (DT above, CD below)

5.3.2 Rewriting coordination structures

We rewrite multi-word conjunctions first, so that we later can rewrite coordination structures with these conjunctions and coordination structures with simple CC conjunctions using the same algorithm. We identify multi-word conjunctions by finding IN heads having RB dependent(s) labelled MWE. We rewrite them using this algorithm:

1. Find the leftmost RB dependent that is attached to the IN node by a dependency relation labelled MWE.
2. Make this RB node the head of the IN node.
3. Attach all other dependents attached to the IN node with an edge labelled MWE, to this new head. We label these dependencies DEP, in a notation that will not be overruled by the subsequent labelling procedure.
4. Attach all other dependents attached to the IN to this new head.
5. Reattach the new RB head to the head of the former IN head. Keep the MWE label.

We rewrite the coordination structures having conjunction nodes (either CC heads or RB heads attached by a MWE dependency relation), using this algorithm:

1. Find the nearest left dependent that is attached to the conjunction node by a conjoining label (se Table 5.5).
2. Perform iteratively for all left dependents (from right to left) that are attached to the conjunction node by a conjoining label (see Table 5.5):

- make this node the head of the former node
- label the dependency COORD, in a notation that will not be overruled by the subsequent labelling procedure.

3. When there are no more conjoined left dependents, make the original head of the CC node the head of the leftmost conjunct. Label this dependency with an empty label.

4. For all other left dependents of the conjunction node, make the leftmost conjunct their head (this ensures correct attachment of shared modifiers).

5. For all right dependents of the conjunction node occurring to the right of the dependent attached by a MRK-NH or MWE dependency, make the leftmost conjunct their head.

Conversion with this algorithm reattaches 31220 dependents, and would increase the unlabelled attachment score by 4.788 points if all these attachments were to match the CD gold data set. The actual increase is 3.978. The unlabelled attachment score after conversion is 77.6 and the labelled attachment score is 70.3. The share of identical roots has increased to 91.3. The number of unmatched and unmatched CC dependents and heads after this rewriting is presented in Table 5.6. Comparing with Table 5.2, we can see that the number of correctly attached CC dependents has increased from 1029 to 12026 and the number of incorrectly used CC heads has decreased from 20373 to 3032.

dpos	slabel	shpos	matched	unmatched
CC			12026	3460
		CC	12228	3032

Table 5.6: Patterns of unmatched dependencies involving coordinating conjunctions after conversion of coordination structures

5.4 Conjunctions as roots

Table 5.6 shows that there are still 3460 incorrectly attached CC nodes in our data. Further investigation of our data reveals that CC tokens or heads of multi-word conjunctions sometimes appear as roots in DT without there being any attached conjuncts to the left. Structures with this kind of CC roots will not be rewritten by the procedure described in Subsection 5.3.2. These roots will practically never have any left dependents. Figure 5.5 shows an example (DT above, CD below).

We convert structures with these roots using the algorithm described below:

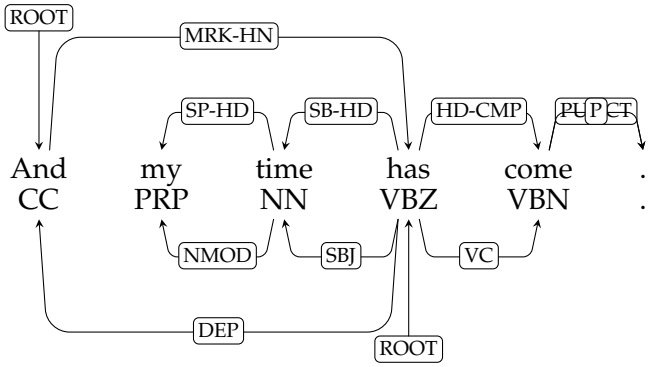


Figure 5.5: Example of a dependency structure with a coordinating conjunction as root and no left conjuncts (DT above, CD below)

1. Make the nearest right dependent the new root of the sentence. We label this dependency ROOT, in a notation that will not be overruled by the subsequent labelling procedure.
2. Make the new root the head of the former conjunction root. We label this dependency DEP, in a notation that will not be overruled by the subsequent labelling procedure.
3. Make the new root the head of all other dependents of the former conjunction root.

Applying this algorithm gives an unlabelled attachment score of 78.1 and a labelled attachment score of 70.8. The share of identical roots increases to 97.2. The number of reattached dependencies is 3636, indicating a maximum increase of 0.557 for the unlabelled attachment score. The actual increase is 0.522. We see from Table 5.7 that the number of correctly attached CC nodes has been further increased from 12026 to 13651. The number of incorrectly used CC heads has decreased to 1247.

dpos	slabel	shpos	matched	unmatched
CC			13651	1835
		CC	12224	1247

Table 5.7: Patterns of unmatched dependencies involving coordinating conjunctions after conversion of coordinating conjunction roots

5.5 Possesive endings

We examine unmatched dependencies involving POS nodes further. We see from Table 5.8 that most POS tokens are incorrectly attached and that the greater part of these are attached by a dependency relation labelled SP-HD. We can also see that of the 6031 POS tokens incorrectly used as heads,

5621 of these incorrectly attached dependents, are attached by arcs labelled SP-HD.

dpos	slabel	shpos	matched	unmatched
POS			43	5868
POS	SP-HD		13	5516
		POS	12	6031
	SP-HD	POS	12	5621

Table 5.8: Patterns of unmatched dependencies involving possessive endings

We find that the pattern illustrated in Table 5.6 is common (DT left, CD right). It seems that while CD attaches the possessive ending to its noun (the possessor), DT attaches it to the noun possessed.

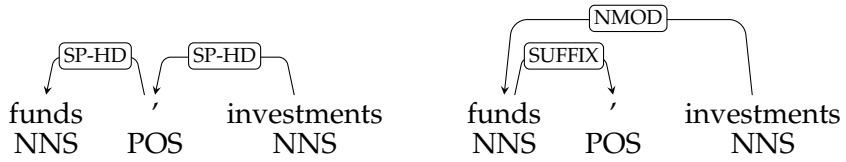


Figure 5.6: Example of dependencies involving possessive endings (DT left, CD right)

We rewrite these structures by reattaching PoS nodes attached to their head by a SP-HD dependency, to its nearest left dependent. dependents of the PoS node are reattached to the former head of the PoS node.

The result is an unlabelled attachment score of 79.8 and a labelled attachment score of 72.4. The share of identical roots is not affected. The number of reattached dependencies is 11063, indicating a maximum increase of 1.7 for the unlabelled attachment score. The actual increase is 1.65. Table 5.9 shows that the number of correctly attached PoS nodes has increased from 43 to 5486, while the number of incorrectly used PoS heads has decreased from 6031 to 512.

dpos	slabel	shpos	matched	unmatched
POS			5486	425
		POS	1	512

Table 5.9: Patterns of unmatched dependencies involving possessive endings after conversion

5.6 Cardinal numbers and currency PoS tags

We investigate the unmatched dependencies involving CD nodes further. Some counts are presented in Table 5.10.

dpos	slabel	shpos	thpos	matched	unmatched
		CD		2144	15375
CD				11416	12719
CD		CD		710	4744
	SP-HD	CD		1262	5219
CD	SP-HD	CD		706	3566
CD		CD	$	0	3411
CD	SP-HD	CD	$	0	3411

Table 5.10: Patterns of unmatched dependencies involving cardinal numbers

Table 5.10 shows us that there are 3411 CD (cardinal number) tokens that are attached to another CD token in our DT format training data, but attached to a $ (dollar sign) token in the CD scheme. All these tokens are attached by an arc labelled SP-HD. We find that the pattern illustrated in Figure 5.7 is common (DT left, CD right). These structures contain both a cardinal number and a numeral, both PoS-tagged CD. While DT attaches the numeral to the dollar sign and the cardinal number to the numeral, CD assigns the dollar sign as head of both the numeral and the cardinal number.

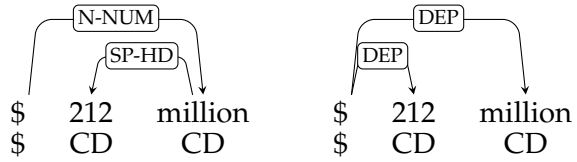


Figure 5.7: Example of dependency structures with cardinal numbers and currency PoS tags (DT scheme above, CD scheme below)

We rewrite these structures by reattaching CD nodes attached by a SP-HD dependency to another CD node, to the head of this second CD, in cases where this head is PoS-tagged $. We label this dependency DEP, in a notation that will not be overruled by the subsequent labelling procedure.

The result is an unlabelled attachment score of 80.3 and a labelled attachment score of 72.9. The share of identical roots is not affected. The number of reattached dependencies is 3417, indicating a maximum increase of 0.524 for the unlabelled attachment score. The actual increase is 0.516. Table 5.11 shows that the number of correctly attached CD nodes has increased by 3361, from 11416 to 14777. The the number of incorrectly used CD heads has decreased from 15375 to 11966.

dpos	slabel	shpos	thpos	matched	unmatched
CD				14777	9358
		CD		2136	11966
CD		CD		702	1335
CD		CD	$	0	2

Table 5.11: Patterns of unmatched dependencies involving cardinal numbers after conversion of structures with cardinal numbers and currency PoS tags

This algorithm should possibly be extended to include structures with heads PoS-tagged # (pound sign), as well. Other currency symbols are not favoured with particular PoS tags and would presumably be PoS-tagged NN or NNS, like *'dollar'* and *'dollars'* spelled out.

5.7 Measure noun phrases

We investigate the unmatched dependencies involving NUM-N labels further. The result is presented in Table 5.12.

dpos	slabel	shpos	matched	unmatched
	NUM-N		32	6682
	NUM-N	CD	30	6059
NN	NUM-N		28	3887
NNS	NUM-N		1	2766
NN			82197	9465
NNS			37659	4356

Table 5.12: Patterns of unmatched dependencies involving labels for measure NPs

We see from Table 5.12 that most of the dependents attached by an edge labelled NUM-N in DT, are attached to a different head in CD. We also see that most of these dependents are attached to a CD node in DT and that the greater part of them are PoS-tagged NN or NNS. We have a closer look at some of these sentences and find these common patterns. See Figure 5.8 (DT left, CD right). DT seems to assign the amount to be head of the measure unit, while CD takes the opposite stance and chooses the measure unit as head of the amount.

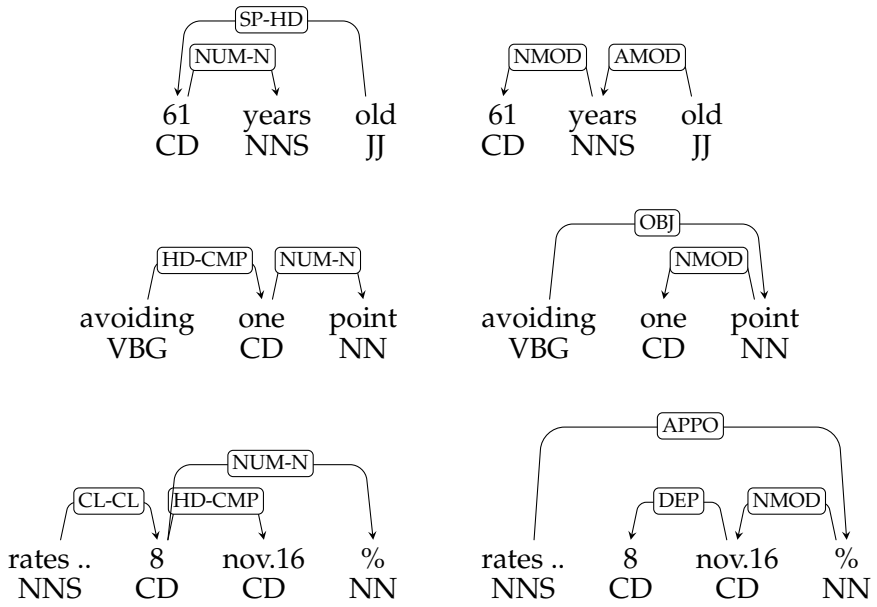


Figure 5.8: Example of dependencies labelled NUM-N in the DT scheme (DT left, CD right)

We rewrite these structures using this algorithm for all dependents attached by an arc labelled NUM-N:

1. Perform iteratively for all nodes (from dependent to head) that are attached to the dependent's head between the head and the dependent):
   - make the former node the head of this node, attach the first node found in this iteration to the dependent
2. Attach all other dependents of the head to the dependent
3. Make the head of the head the new head of the dependent, label this dependency with an empty label
4. Make the the last node found in the iteration the new head of the original head, label this dependency with an empty label

The result is an unlabelled attachment score of 82.3 and a labelled attachment score of 74.5. The share of equal roots is still 97.2. The number of reattached dependencies is 16152, indicating a maximum increase of 2.477 for the unlabelled attachment score. The actual increase is 2.0. Table 5.13 shows that the number of correctly attached NN nodes has increased by 3151 to 85348. The number of correctly attached NNS nodes has increased by 2491 to 40150. The number of incorrectly used CD heads has decreased to 3500. When comparing with Table 5.11 we see that a side effect has been that a considerable number of CD nodes have been correctly attached as this number has been increased to 20440 by this rewriting.

dpos	slabel	shpos	matched	unmatched
CD			20440	3695
		CD	2379	3500
NN			85348	6314
NNS			40150	1865

Table 5.13: Patterns of unmatched dependencies involving cardinal numbers after conversion of structures with measure noun phrases

5.8 Cardinal number heads of nouns

We still have some nouns that are incorrectly attached to CD heads. See Table 5.14.

dpos	slabel	shpos	matched	unmatched
CD			20440	3695
		CD	2379	3500
NN		CD	20	238
NNP		CD	16	906
NNS		CD	4	8
NN			85348	6314
NNP			51957	7312
NNS			40150	1865

Table 5.14: Patterns of unmatched dependencies involving CD heads of nouns

Figure 5.9 shows an example of an unmatched dependency with noun dependent and CD head labelled with other labels than NUM-N in DT (DT format left, CD format right). In CD the date and month are not directly connected.

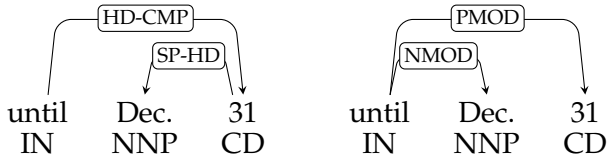


Figure 5.9: Example of a dependency with a noun dependent and cardinal number head in DT (DT left, CD right)

We rewrite these structures by making the head of the head of an NN/NNS/NNP node attached to a CD head, the new head of this dependent. The dependent is made the head of the old head. We label all these dependencies with empty labels. Other dependencies of the old head node are attached to the dependent.

The result is an unlabelled attachment score of 82.6 and a labelled attachment score of 74.8. The share of equal roots is still 97.2. The number of reattached dependencies is 3372, indicating a maximum increase of 0.517 for the unlabelled attachment score. The actual increase is 0.356.

dpos	slabel	shpos	matched	unmatched
CD			21524	2611
		CD	2175	2044
NN		CD	0	1
NNP		CD	0	0
NNS		CD	0	1
NN			85511	6151
NNP			52764	6505
NNS			40153	1862

Table 5.15: Patterns of unmatched dependencies involving cardinal numbers after conversion of structures with cardinal number heads of nouns

Table 5.15 shows that we have (almost) no nouns incorrectly attached to CD nodes after this conversion. The numbers of NN, NNS and NNP tokens correctly attached have increased by 163, 807 and 3, respectively. The number of incorrectly used CD heads has decreased to 2044. A side effect is that the number of correctly attached CD nodes have been further increased from 20440 to 21524. Unfortunately, these improvements have been at a cost of reducing the number of correctly used CD heads from 2273 to 2175.

5.9 Determiner heads of nouns

We will move on to investigate constructions with determiner heads in DT. As we can see from Table 5.16, more than half of the dependents incorrectly attached to DT heads are nouns.

dpos	slabel	shpos	matched	unmatched
		DT	748	2030
NN		DT	1	1031
NNP		DT	2	84
NNS		DT	0	49
NNPS		DT	0	3
NN			85511	6151
NNP			52764	6505
NNS			40153	1862
NNPS			1467	152

Table 5.16: Patterns of unmatched dependencies involving DT heads of nouns

Figure 5.10 shows some examples of unmatched dependencies consisting of a noun with a determiner head in the DT format (DT left, CD right). These examples indicate that in these kind of constructions, DT assigns the DT node as head of the noun, while CD treats it as an ordinary determiner and considers it a dependent of the noun.

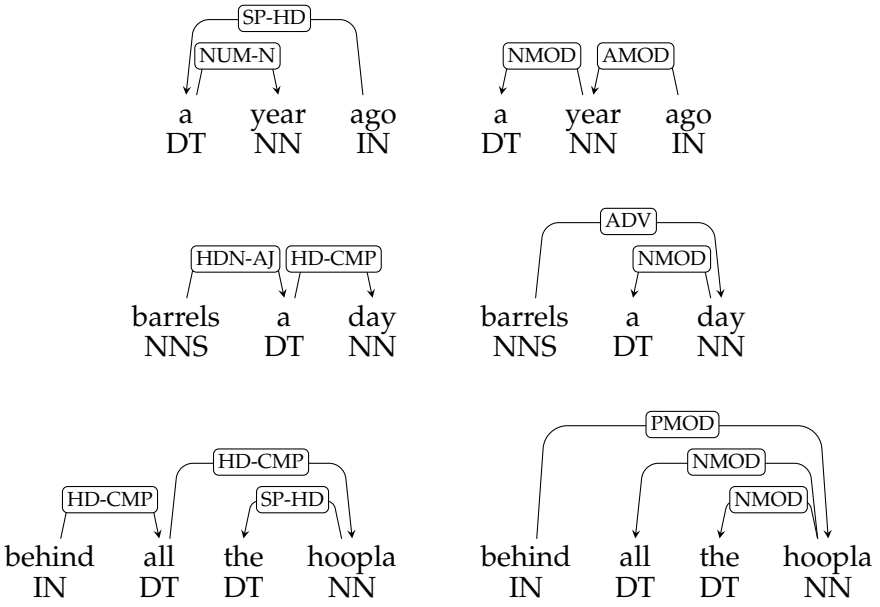


Figure 5.10: Examples of dependencies with noun dependents and determiner head in the DT data (DT left, CD right)

In the first of these examples, *'a'* is actually a cardinal number, meaning *'one'*, not a determiner. In the second graph *'a'* works as a preposition and should ideally have been PoS-tagged IN. In the last example there are two determiners. CD allows more than one specifier, DT does not.

When we compare the number of incorrectly used DT heads in Table 5.16 to the number of incorrectly used DT heads presented in Table 5.1, we observe that the number of incorrectly used DT heads has decreased. The first example graph shown in Figure 5.10 contains a dependency with an arc labelled NUM-N. This dependency, and others with the same pattern, will already have been rewritten by the procedure for rewriting structures with measure noun phrases (see Section 5.7).

We rewrite the remaining structures involving nouns with determiner heads by making the head of the head of an NN/NNS/NNP/NNPS node attached to a DT head, the new head of this dependent. The dependent is made the head of the old head. This dependency is labelled with an empty label. Other dependencies of the old head node are attached to the dependent.

The number of dependencies being reattached by this procedure is 2447. This represents a possible gain of 0.375. The actual gain is 0.356, giving an unlabelled attachment score of 83.0. The labelled attachment score is

increased to 75.0 and the share of equal roots remains 97.2.

dpos	slabel	shpos	matched	unmatched
		DT	743	759
NN		DT	0	0
NNP		DT	0	0
NNS		DT	0	1
NNPS		DT	0	0
NN			86472	5190
NNP			52811	6458
NNS			40230	1785
NNPS			1470	149

Table 5.17: Patterns of unmatched dependencies involving DT heads of nouns after conversion

Table 5.17 shows that we have (almost) no nouns incorrectly attached to DT nodes after this conversion. The number of incorrectly used DT heads has decreased from 2030 to 759. The amount of correctly attached noun dependents has increased for all noun PoS tags.

5.10 Punctuation marks

We know that in DT punctuation marks are normally attached to the neighbouring word from which it is not separated by a space (Ivanova et al. 2013). For the CD scheme there is little, if any, documentation on attachment of punctuation marks. Table 5.18 shows the amount of unmatched and matched dependencies for various punctuation mark dependents in our training data sets. Table 5.19 contains some information on how the punctuation marks are attached in our training data.

PoS tag	frequency	matched	unmatched
.	29222	1569	27653
,	31738	8330	23408
:	2517	690	1827
(	628	335	293
)	634	169	465
”	4687	837	3850
"	4779	730	4049

Table 5.18: Number of matched and unmatched dependencies with punctuation mark dependents

We rewrite the ‘:’ roots in DT by attaching them to their nearest left dependent, making this the new root. All other dependents of the ‘:’ node are reattached to this new root. The dependencies are labelled with an empty label. This increases the number of identical roots with 0.42.

	DT				CD			
	is attached to				is attached to			
PoS tag	root	prec. word	foll. word	is root	root	prec. word	foll. word	is root
.	1454	29098			28660	1560	1	15
,	676	31629	1		13515	8325	1079	
:	532	1748	58	141	1329	673	79	2
(	6		614		112	31	336	
)	6	522			111	164	24	
”	17	1101	12		3163	744	939	
"	126		4760		2417	660	724	

Table 5.19: Heads of punctuation marks in our formats

We use the rules that give the best results, to reattach punctuation marks:

- ‘.’, ‘”’ and ‘)’ are attached to the root, in cases where this does not create non-projectivity.
- ‘,’, ‘:’ and ‘"’ are attached as far up as possible without creating non-projectivity, following the dependency path
- ‘(’ is attached to the following word

The result from this conversion is an unlabelled attachment score of 89.9 and a labelled attachment score of 82.0. The share of identical roots is now 97.6. The number of reattached dependencies is 57705, indicating a maximum increase of 8.851 for the unlabelled attachment score. The actual increase is 6.957.

PoS tag	frequency	matched	unmatched
.	29222	26432	2790
,	31738	21332	10406
:	2517	1346	1171
(	628	336	292
)	634	249	385
”	4687	3782	905
"	4779	3682	1097

Table 5.20: Number of matched and unmatched dependencies with punctuation mark dependents after conversion

Table 5.20 shows the amount of matched and unmatched dependencies with punctuation mark dependents after rewriting. A considerable number of punctuation nodes are still incorrectly attached. These unmatched dependencies constitute 2.6% of the dependencies in our training data.

If we ignore punctuation nodes altogether, our conversion results in an unlabelled attachment score of 91.4.

5.11 Unsuccessful rewritings

As we observed during the rewriting process, far from all our reattachments resulted in structures identical to those in the CD data set. Table 5.21 presents an overview over proportions of erroneously reattached dependents per step in our rewriting procedure.

pattern	no deps rewritten	expected improv.	actual improv.	erroneous reattach.	error rate
CC	31220	4.788	3.978	0.81	16.9
CC roots	3636	0.557	0.522	0.035	6.3
POS	11063	1.7	1.65	0.05	2.9
CD curr.	3417	0.524	0.516	0.008	1.5
NUM-N	16152	2.477	2.0	0.477	19.3
noun + CD	3372	0.517	0.356	0.161	31.1
nount + DT	2447	0.375	0.356	0.019	5.1

Table 5.21: Erroneous reattachments per pattern

Table 5.21 shows that rewriting of structures involving cardinal numbers and currency PoS tags, had the lowest share of unsuccessful reattachments. Only 1.5% of the 3417 dependents that were reattached during this procedure, were assigned another head than in the CD data. The three procedures that had the highest share of erroneous reattachments, were rewritings of coordinated structures, structures with measure noun phrases and structures with cardinal numbers as heads of nouns, the percentage of erroneously attachments being 16.9, 19.3 and 31.1, respectively. In the following, we will look into some examples that our algorithms did not manage to convert correctly according to the CD gold data.

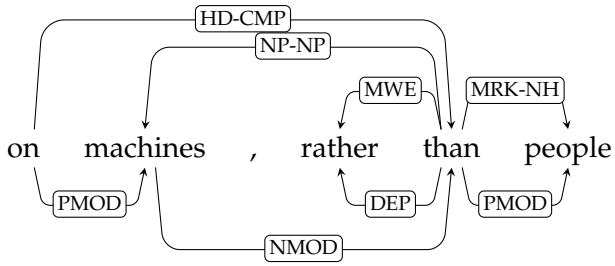


Figure 5.11: Example of coordination structure where different elements are coordinated in DT and CD (DT above, CD below)

When it comes to coordination structures, we find that some times different elements are coordinated in the two formats. In some other

cases an element is treated as a modifier in one format and as a conjunct in the other. Figure 5.11 shows an example of different coordination. In this example DT analyses *'rather than'* as a coordinating conjunction, coordinating *'machines'* with *'people'*. CD on the other hand, treats *'rather than'* as a preposition. During our conversion the multi-word expression is rewritten and *'people'* is attached to its new head *'rather'* as a conjunct.

Inspecting measure noun structures that are not rewritten according to the CD data, we find examples like the one shown in Figure 5.12 (DT above, CD below). In this example all dependencies are initially matched, all dependents are identically attached in both formats. Our conversion procedure will, however, reattach *'15'* to *'%'* and *'%'* to *'stakes'*.

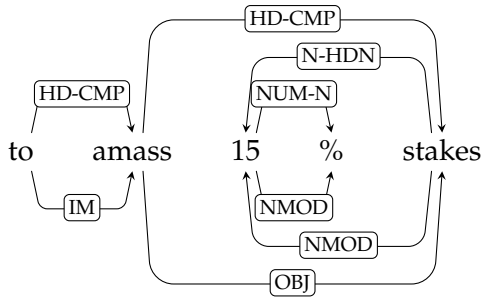


Figure 5.12: Example of a structure with measure noun that is not correctly rewritten (DT above, CD below)

Finally, we pick an example of a structure with a cardinal number head of a noun (in DT), that is not successfully rewritten. The example is shown in Figure 5.13.

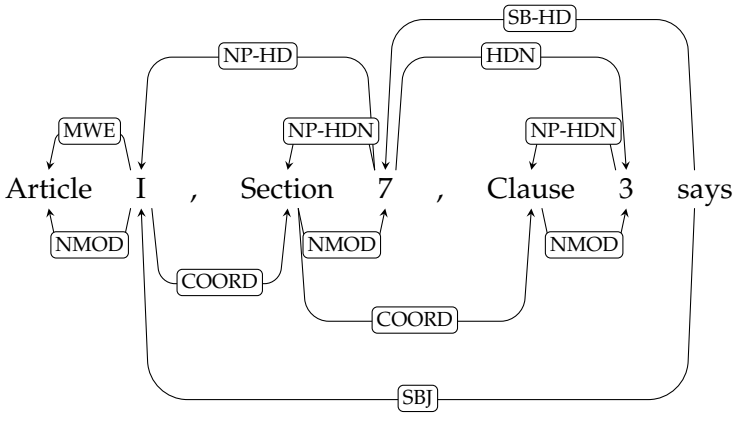


Figure 5.13: Example of a structure with a cardinal number head of a noun that is not correctly rewritten (DT above, CD below)

Our conversion procedure will erroneously attach both *'7'* and *'Clause'* to *'I'*. *'Section'*, as well, will be attached to *'I'*. This latter attachment is in accordance with the CD gold data.

5.12 Summary of results

Table 5.22 shows the results from the different steps of converting for our training data (LAS=labelled attachment score, UAS=unlabelled attachment score). The rewriting steps that have led to most improvement are the reattachment of punctuation tokens (6.9), rewriting of coordination structures (4.0) and rewriting of structures with measure noun phrases (2.0). Altogether we have achieved an improvement of 16.3 of the unlabelled attachment score. The labelled attachment score has increased by 15.4 compared to the result from the baseline conversion (Section 4.3).

	UAS	LAS	Percentage of identical roots
Baseline converter	73.6	66.6	84.4
Coordination structures	77.6	70.3	91.3
CC Roots	78.1	70.8	97.2
Possessive endings	79.8	72.4	97.2
CD and currency	80.3	72.9	97.2
CD and measure nouns	82.3	74.5	97.2
CD heads of nouns	82.6	74.8	97.2
DT heads of nouns	83.0	75.0	97.2
Punctuation marks	89.9	82.0	97.6

Table 5.22: Evaluation results for converter

Table 5.23 shows the percentage used of the 10 most common PoS tags used as roots in the converted data set and in the CD data.

	CD	Converted
VBD	43.5	43.3
VBZ	28.3	28.2
VBP	14.3	14.2
MD	8.2	8.3
CC	0.0	0.0
VBN	0.5	0.5
VB	0.7	0.8
NN	1.1	1.1
IN	0.3	0.2
RB	0.1	0.4
	97.0	97.0

Table 5.23: Most common PoS tags used as roots in CD and in DT converted

Comparing Table 5.23 table to Table 3.7, we find that the most distinctive difference in distribution of roots for the original DT format and our converted data, is that the use of CC tokens as roots is now abandoned. We also see that we have achieved an approximately similar use of finite

verbs (VBD, VBZ and VBP) as roots in our converted data as in CD. Another PoS tag that has turned up among the 10 most frequent for roots, now that we compare only two formats, is RB (adverb). 0.4 percent of all roots in our converted data is PoS-tagged RB, whereas only 0.1 percent of roots in CD has this PoS tag. This PoS tag does not appear in Table 3.7, but checking our original DT training data set, we find the same proportion of RB roots there, 0.4 percent.

Table 5.24 shows the amount of unmatched and matched dependencies with the dependent PoS tags, source format labels and/or head PoS tags that we chose to investigate in this study (see Table 5.1), after rewriting. Punctuation token dependents are not included.

dpos	hpos	matched	unmatched
CC		13860	1626
	CC	12578	909
POS		5497	414
	POS	1	512
CD		21527	2608
	CD	2175	2044
	DT	743	759

Table 5.24: Matched and unmatched dependencies after conversion

Comparing the two first rows of these tables, we observe that the number of correctly attached CC nodes has increased from 1029 to 13860. We also see that the number of correctly attached POS nodes has increased from 43 to 5497. The number of correctly attached CD nodes has increased from 11317 to 21527. The table also reveals a reduction of dependents incorrectly attached to CC, POS and CD heads. These reductions do, however, not in themselves ensure that more dependents have been attached in accordance with the CD gold data set (this is explained in Section 4.2).

Converting our development dataset using this converter gives an unlabelled attachment score of 90.1 and a labelled attachment score of 81.8. The share of identical roots after conversion is 98.0.

Earlier in this section we observed that our conversion did not increase the labelled attachment score at the same rate as the unlabelled attachment score (15.4 versus 16.3). In the next section we will describe an attempt to implement a more effective labelling procedure.

Chapter 6

Labelling by classification and contrastive evaluation

The heuristic labelling procedure that we have designed (see Sections 4.3 and 5.1) leaves about 9% of the correctly rewritten dependencies labelled with an incorrect label. In this final part of the project, we will try to implement a labelling procedure that performs better, by utilizing more of the available information about these dependencies. For this endeavour, we will use a method of statistical classification, also known as machine learning-based classification. We will give a brief introduction to the topic of machine learning and our choice of tool for this task in Section 6.1. In Section 6.2 we describe the notation we use for describing elements of information, features, that we choose to make use of. Our experiments with machine-learned classifiers are described in Section 6.3. In Section 6.4 we will perform a contrastive evaluation of our converters; we will compare the accuracy of the purely heuristic and the partly machine-learned conversion pipelines on previously unseen test data.

6.1 Choice of machine learner

Our task is to assign CoNLL Syntactic Dependencies (CD) dependency types to dependencies converted to this format from the DELPH-IN Syntactic Derivation Tree (DT) format. This can be viewed as a classification problem: we have a set of *classes* (CD labels) and a set of *objects* (dependencies) and our task is to determine which class each object belongs to. In machine-learned classification, a statistical learning algorithm will use a training set of objects encoded with the correct class, to learn a classification function. In order to train a machine-learned classifier that can assign CD labels to our converted dependencies, we will need a training set of dependencies (objects) encoded with the correct CD label (class). A possible training set could be the CD gold data. However, we decide to use the converted data for training, as these are the data that eventually will be the target for our classifier. Thus, our matched (correctly attached) dependencies after conversion along with their label from the CD gold data, will constitute our training set.

Several methods of machine learning for classification exist, each with their particular strengths and weaknesses. Issues that should be considered when seeking to find the most appropriate method are the amount of training data available, the scale of the classification problem (how many classes are there?), available time and processing power (some methods are high consumers).

In our classification task we have 62 possible classes. These are the CD labels used in our CD training data set. This implies that our task is a medium-sized classification problem. Our training data set consists of 586339 objects. This is a considerable amount of training data. On this background we choose the method of support vector machines (SVM), that can be expected to have a good performance on classification problems of this scale, when sufficient training data is available. We will use SVMlight (*SVMlight*), which is an implementation of SVM[1]. We use the SVMmulticlass instantiation of this package.

For this kind of machine learning, we need to subtract a subset of properties, that we want the learning algorithm to make use of, from the objects. Such properties are called *features*. In the next section we will describe a notation that we will use for describing such features.

6.2 Notation for describing features

Our notation for describing our feature model is an adaptation from Nivre et al. (2007). We will make use of three feature types; part-of-speech features, dependency type features and lexical features:

- *p(n)* - the part-of-speech tag of the node *n*
- *l(n,s)* - the dependency type of the node *n* in source *s*[2]
- *w(n)* - the word form of the node *n*

In our project we have two different sources for our feature values in our project, the original DT data and the converted data. The PoS tag and word form of a node are identical in both formats, we do not need the source parameter to identify these properties. We will use the following constants to refer to our source formats:

- *D* - denotes the original DT data set
- *C* - denotes our converted data set

We also need functions to address one node relative to another:

- *h(n,s)* - the head of the node *n* in source *s*

[1] We want to thank Erik Velldal for his helpfulness and invaluable advice on the use of SVM.

[2] Although we end up using only labels from the original DT data, our notation allows for the use of labels from other sources as well.

- *ls(n,s)* - the leftmost sibling of the node *n* in source *s*
- *rs(n,s)* - the rightmost sibling of the node *n* in source *s*
- *ld(n,s)* - the leftmost dependent of the node *n* in source *s*
- *rd(n,s)* - the rightmost dependent of the node *n* in source *s*

Sometimes we will want to address a node relative to another in the linear order:

- *n + x* - the node that occurs *x* positions to the right of node *n*
- *n - x* - the node that occurs *x* positions to the left of node *n*

Thus, *p(ls(t,C)))* denotes the PoS tag of leftmost sibling of the node *t* in the converted data. *l(t + 1, D)* denotes the dependency type from the DT data of the node to the immediate right of node *t*. If a property is non-existent for a dependent (all nodes does not have, for instance, a right dependent) the feature is not used, neither in training or classification. If two properties have the same value (for instance, if for a node, the PoS tag of the rightmost dependent in DT has the same value as the PoS tag of rightmost dependent in the converted format), they are still treated as two distinct features.

The SVMmulticlass part of SVM-Light does not support non-linear kernels and does not model the combination of simple features. This implies that we also need to make use of complex features that combine the simple ones. We will use *&* when describing these features. For example, *p(t)&l(t,D))* will denote the feature combining the values of PoS tag of t with the label of t from DT.

Table 6.1 shows examples of some feature values for the structure illustrated in Figure 6.1 below. In this illustration we have omitted the labels of the rewritten structure, as we do not use these as features in our experiments.

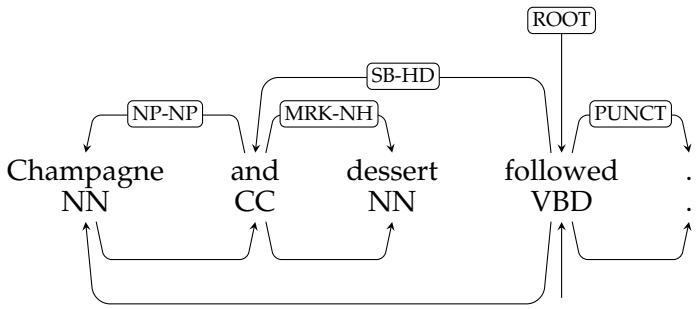


Figure 6.1: Example of a rewritten but still not labelled sentence (source DT above, converted below)

Feature	Value	Description
p(t)	CC	PoS tag of *'and'*
l(t,D)	SB-HD	Label of *'and'* in DT
w(t-1)	Champagne	Wordform of token to the immediate left of *'and'*
p(h(t,D))	VBD	PoS tag of the head of *'and'* in DT
w(rd(t,C))	dessert	Wordform of the rightmost dependent of *'and'* in the converted data
p(t+1)&l(t+1,D)	NN&MRK-NH	PoS tag & label from DT of the token immediately right of *'and'*

Table 6.1: Examples of possible feature values described in our notation. *t* refers to the token *'and'*

6.3 Experiments

In our first experiment we will attempt to replicate our heuristic labelling procedure by using feature values that correspond to the parameters (triggers) used in the heuristic procedure. These parameters are dependent PoS tag, source format label, head PoS tag and combinations of these. We will refer to this group of features as **A**.

A: p(t), l(t,D), p(h(t,D), p(t)&l(t,D), l(t,D)&p(h(t,D)), p(t)&l(t,D)&p(h(t,D)), p(t)&p(h(t,D))

Table 6.2 shows how these features correspond to the triggers described in Table 4.2. Note that the properties *dpos* and *hpos* are not used alone in the heuristic procedure, only in combination with other properties. The *dpos-hpos* trigger was introduced in the extended procedure (see Section 5.1).

Trigger	Feature
dpos-hpos-label	p(t)&l(t,D)&p(h(t,D))
dpos-label	p(t)&l(t,D)
hpos-label	l(t,D)&p(h(t,D))
dpos-hpos	l(t,D)
label	l(t,D)
dpos	p(t)
hpos	p(h(t,D)

Table 6.2: Triggers versus feature values

We train a classifier with these features and apply it for labelling of our converted development dataset. The classifier is used labelling only dependencies that have not been explicitly labelled in the rewriting process. The result is a labelled attachment score of 78.77, considerably lower that the 81.8 that we achieved by using the heuristic procedure (see Section 5.12). This is a bit puzzling, considering that the machine learner

apparently has the same information available as our heuristic procedure, in addition to the simple features *dpos* and *hpos*. A difference is that in our heuristic procedure, we limited the use of the *dpos-hpos* trigger to the dependencies not labelled during the rewriting process. This information about labels assigned by the rewriting procedure, is not made available to the machine-leaner by the features in Group **A**.

We will also train classifiers using additional features. Our feature selection is inspired by Johansson et al. (2008). We start by using the PoS tags of the leftmost and rightmost siblings of the dependent, as well as the leftmost and rigthmost dependents of it. We will use these values both from the original DT data and the converted data. We will call this group of features **B**.

B: p(ls(t,D)), p(rs(t,D)), p(ls(t,C)), p(rs(t,C)), p(ld(t,D)), p(rd(t,D)), p(ld(t,C)), p(rd(t,C)), p(ls(t,D))&p(rs(t,D)), (ls(t,C))&p(rs(t,C)), p(ld(t,D))&p(rd(t,D)), p(ld(t,C))&p(rd(t,C))

Training a classifier with these features in addition to the features in Group **A** and applying it on the converted data, results in a labelled attachment score of 80.52.

Group **C** contains features with word forms of the dependent and its head in the original DT data set. A classifier trained with these feature values in addition to the Group **A** features, achieves a labelled attachment score of 79.74.

C: w(t), w(h(t,D), w(t)&w(h(t,D)

We train a classifier using the features in all three feature groups; **A**, **B** and **C**. Using this classifier for labelling gives us a labelled attachments score of 81.32.

We have still not managed to train a classifier that performs to the level of our heuristic labelling procedure on our development data. We continue to explore additional features. Our next step is to use properties of the words linearly next to the dependents. Group **D** contains PoS tags of the node immediately to the left of the dependent, the node immediately left to this and the node to the immediate right of the dependent.

D: p(t-2), p(t-1), p(t+1), p(t-1)&p(t)&p(t+1), p(t-1)&p(t), p(t)&(p(t+1)), p(t-2)&p(t-1)&p(t)

A classifier trained with this features, as well as the feature values from Group **A** gives a labelled attachment score of 80.34. Using the feature values from Group **A**, **B**, **C** and **D** increases the labelled attachment score to 82.03, finally at the same level as our heuristic labelling procedure.

We will also explore the use of PoS tag and word form of the head in the converted data, Group **E**.

E: p(h(t,C)), w(h(t,C)), p(t)&p(h(t,C)), w(t)&w(h(t,C)

Training a classifier with the features from Group **E** in addition to the features in Group **A** and applying it on the converted data results in a labelled attachment score of 80.86. Using the features described in Group **A** – **E**, gives a labelled attachment score of 82.66.

We will end our experiments by investigating properties from further up in the dependency path, properties of the head of the head (the grandparent), as features. Feature Group **F1** contains features with PoS tags and labels from from the grandparent in the original DT data. Some features with combinations of PoS tags and labels from this grandparent are defined in Group **F2**.

F1: p(h(h(t,D))),p(t)&p(h(h(t,D))), p(h(t,D))&p(h(h(t,D))),
p(t)&p(h(t,D))&p(h(h(t,D))), l(h(t,D)), l(h(h(t,D))), l(t,D)&l(h(t,D)),
l(h(t,D))&l(h(h(t,D))), l(t,D)&l(h(t,D))&l(h(h(t,D)))

F2: p(t)&l(t,D)&p(h(t,D))&l(h(t,D)),
p(t)&l(t,D)&p(h(t,D))&l(h(t,D))&p(h(h(t,D)))&l(h(h(t,D)))

A classifier trained with the feature values from Group **A** and **F1** achieves a labelled attachment score of 79.91. If we use features from Group **F2**, as well, the score increases to 80.06. Training a classifier with the features from Group **A** – **E** in addition to the features from both Group **F1** and **F2** results in a labelled attachment score of 82.89.

Finally, we will explore the use of features from the head of the head in the converted data. Feature Group **G** contains features with PoS tags from this grandparent.

G: p(h(h(t,C))),p(t)&p(h(h(t,C))), p(h(t,C))&p(h(h(t,C))),
p(t)&p(h(t,C))&p(h(h(t,C)))

Training a classifier with the features from Group **G** in addition to the features in Group **A** and applying it on the converted data results in a labelled attachment score of 79.65. Using the features described in Group **A** – **G**, gives a labelled attachment score of 82.96.

Using all our described features for training, results in a classifier that achieves a labelled attachment score of 82.96 on our development dataset. In Table 6.3 we give a summary of the performance of classifiers trained with different sets of features.

Feature groups used								Score
A	B	C	D	E	F1	F2	G	Development
X								78.77
X	X							80.52
X		X						79.74
X	X	X						81.32
X			X					80.34
X	X	X	X					82.03
X				X				80.86
X	X	X	X	X				82.75
X					X			79.91
X					X	X		80.06
X	X	X	X	X	X	X		82.89
X							X	79.65
X	X	X	X	X	X	X	X	82.96

Table 6.3: Results from labelling with machine-learned classifiers

Table 6.4 shows the F-score per class (CD label) for both the heuristic and the machine-learned labelling procedure, for the 20 most frequent used CD labels in the training data, sorted by descending frequency. These 20 labels cover 95.8% of the edges in our training data set. The F-scores are from development data and the classifier used is trained with all features described in Table 6.3.

We observe that the machine-learned classifier achieves a higher or similar F-score than the heuristic procedure for almost all classes. The *APPO* (apposition) label is an exception. For this label the heuristic labelling procedure obtains an F-score of 0.93, while the F-score for the classifier is only 0.87. We also notice that the labels that are hard to assign correctly for the heuristic method, also are the ones that present a challenge to the classifier. The labels with the lowest F-score for both methods are the *LOC* (locative adverbial or nominal modifier), *TMP* (temporal adverbial or nominal modifier) , *PRD* (predicative complement), *ADV* (general adverbial) and *OPRD* (predicative complement of raising/control verb) labels.

We see from Table 6.3 that using all feature groups gives the best result on our converted development data. We decide to make use of this configuration when we do our final evaluation of our machine-learned labelling procedure.

CD label	Frequencies	Heuristic	Classifier
NMOD	179536	0.97	0.97
P	74135	1.00	1.00
PMOD	64569	0.98	0.98
SBJ	52163	0.97	0.98
OBJ	38686	0.87	0.89
ROOT	29672	1.00	1.00
ADV	27228	0.61	0.72
VC	20746	0.96	0.95
COORD	16598	0.98	0.97
DEP	15897	0.86	0.90
NAME	15648	0.81	0.83
TMP	14648	0.41	0.45
CONJ	13465	0.99	0.99
AMOD	10095	0.79	0.81
LOC	10049	0.02	0.31
PRD	9489	0.57	0.63
IM	9067	1.00	1.00
APPO	8793	0.93	0.87
SUB	7241	0.96	0.95
OPRD	6645	0.68	0.71

Table 6.4: F-score for CD labels for heuristic and machine-learned labelling

6.4 Evaluation

Finally, it is time to check how well our converters perform on the held-out test data. Table 6.5 shows the result of our converters on both development and test data. The first column shows the proportion of identical roots in our converted data and the CD gold data. The unlabelled attachment score is presented in the second column. In the third and fourth column, the labelled attachment scores using the heuristic procedure and the machine-learned classifier, respectively, are shown.

	Identical roots	Unlabelled att. score	Labelled att. score using	
			heuristics	classifier
Development	98.0	90.1	81.8	83.0
Test	98.0	90.0	81.6	82.9

Table 6.5: Evaluation results on development and held-out test data

Table 6.5 shows that our converter obtains a high precision on identification of the correct root, 98% on both the development and test data. The unlabelled attachments score obtained on development data is 90.1. A similar score is obtained on the test data, 90% of the nodes in the test data have been assigned the correct head.

The heuristic labelling procedure obtains a labelled attachment score of 81.8 and 81.6 on the development and test data, respectively. This means that it assigns an incorrect label to 9.3% of the dependencies that have correct attachment (and are included in the LAS metric) in the development dataset and to 9.3% of these kind of dependencies in the test dataset.

The machine-learned classifier performs better than the heuristic labelling procedure on both development and test data. It achieves a labelled attachment score of 83.0 on the development data and leaves 7.9% of the dependencies with correct attachment incorrectly labelled. The labelled attachment score for the test data is 82.9, implying that in the test data, as well, 7.9% of the possible dependencies are incorrectly labelled after conversion.

The heuristic rewriting procedure, the heuristic labelling procedure and the machine learned-based labelling procedure all perform evenly across development and test data. These are robust results on unseen data, implying that we have avoided over-tuning.

In natural language processing (NLP), statistical significance tests are often used to determine the probability that the difference in performance between two models is pure coincidence. The null hypothesis is that the difference is due to mere chance, and we can only reject this hypothesis if the computed p-value, i.e. the probability of this, is smaller than a predefined value α.

We want to check whether the difference in performance between our heuristic and machine-learned labelling procedure is statistically significant. A common value for α in NLP is 0.05 and this is also the significance level that we will use. We will use the Wilcoxon signed-rank test (Wilcoxon 1945), that looks at pairs of scores over the same samples. First, we split our test data into 10 subsets, with approximately 304 sentences in each subset. We evaluate this subset separately both for the heuristic and machine-learned procedure, and use the labelled attachment scores from both procedures as input for the Wilcoxon signed-rank test. We perform the same statistical significance test using 20 subsets (152 sentences in each) and 98 subsets (31 sentences in each). The three tests all result in a p-value smaller than 0.05, thus enabling us to state that the machine-learned classifier performs significantly better than the heuristic labelling procedure.

Finally, to further put our work into perspective, we seek to relate the accuracy levels available from our converters to the state of the art in data-driven dependency parsing. A parser generates labelled dependency structures from text that has not been given any previous syntactic analysis. One could hope that access to a gold standard syntactic analysis, although from a different linguistic theory, would make the task of dependency annotation easier. Ivanova et al. (2013) reports parsing results of both MaltParser (Nivre et al. 2007) and MST (McDonald et al. 2005) on CD. The best unlabelled attachment score obtained is 92.01 (MST). The best labelled attachment score reported is 88.74 (MaltParser). Punctuation mark tokens are excluded from the scoring. Table 6.6 shows the result of evaluation of our converters when punctuation marks are excluded. Our unlabelled

attachment score 91.7 on the test set is at approximately the same level as the state-of-the-art parsing result of 92.01 reported by Ivanova et al. (2013). The labelled attachment score of 83.7 is, however, considerably lower than the 88.74 reported in this survey.

	unlabelled att. score	labelled att. score using heuristics	classifier
Development	91.5	82.2	83.5
Test	91.7	82.2	83.7

Table 6.6: Evaluation results on development and held-out test data when punctuation marks are excluded

Chapter 7

Conclusion

In this thesis we started out performing quantitative studies of our selected dependency schemes, Stanford Basic Dependencies (SB), CoNLL Syntactic Dependencies (CD) and DELPH-IN Syntactic Derivation Tree (DT), estimating the expressiveness, with regard to granularity and variability, of each format. We compared the three formats and reported the degree of correspondences in syntactic structure, sentence roots and tree-depth for each format pair.

We investigated how conversion between dependency formats can be performed and presented a methodology for identifying patterns of structural differences in format pairs. We designed and implemented a heuristic baseline converter, taking advantage of the basic statistics obtained earlier.

Through these first parts of our study, we demonstrated how a collection of relatively simple descriptive statistics can uncover relevant structural and linguistic properties, both within a single format and when comparing parts of formats. Information that we acquired about expressiveness of formats and similarities between pairs of formats, in the first part of the study, motivated the choice of DT and CD as the format pair for which we would attempt conversion. We identified and documented several systematic differences between these two formats, using the methodology presented in the second part of the study.

In the final parts of the study, we implemented a converter, heuristic both with respect to rewriting of syntactic structures and labelling, for conversion from DT to CD. We also trained machine learned-based classifiers for the labelling task. Finally, we evaluated our converters on held-out test data.

7.1 Results

The main outcomes of our work may be summarized as follows:

- We have found that among our three selected dependency formats, CD/DT and SB/CD are the most similar format pairs, having the highest correspondence with regard to syntactic structure. With

respect to roots and tree-depth, CD/DT is the most similar pair. CD/DT is also the pair with the highest correspondence of syntactic structure, when punctuation mark dependents are ignored.

- Our quantitative study did not result in any findings indicating that any of the formats are substantially more expressive or linguistically rich than the others. The three formats use similar numbers of different dependent PoS tag, label and head PoS tag combinations.
- We have proposed a methodology for identification of patterns of systematic differences in syntactic structure between dependency formats.
- We have identified and documented several linguistic phenomena that have a different syntactic analysis in CD and DT.
- We have designed and implemented a heuristic converter for conversion of data annotated in the DT format to CD. This converter achieves an unlabelled attachment score of 90.0 (91.7 when punctuation tokens are excluded) and a labelled attachment score of 81.6 (81.6) on held-out test data.
- We trained machine-learned classifiers for improved labelling. This labelling procedure achieves a labelled attachment score of 82.9 (83.7 when punctuation tokens are excluded). This is a statistically significant improvement, but still below the level that we had hoped to obtain.

7.2 Reflections and further work

7.2.1 Estimating variability

When estimating variability for the selected dependency formats, we only counted the number of combinations of dependent PoS tag, label and head PoS tags used in each format. We did not consider the distribution of these combinations; whether in a format only a few combinations cover a comparatively high number of dependencies and a large number of combinations occur rarely. Investigating this further would probably provide more certain knowledge about the actual variability of the formats.

7.2.2 Rewriting of syntactic structures

Our heuristic rewriting procedure obtained an unlabelled attachment score of approximately 90. Among the remaining 10% of incorrect dependencies, there is bound to be some amount of irregularities, dependencies that are annotated incorrectly in one or both formats, and thus impossible to rewrite according to the gold standard in a systematic manner. Although the proportion of such irregularities is hard to estimate, we believe that there still remain some systematic differences that could be successfully rewritten.

We know that 2.6% of the incorrectly attached (according to the CD gold data) dependents in our converted training data, are punctuation mark tokens, about 60% of these being ',' (commas). These are not the simplest tokens to attach correctly in the CD format. An approach of statistical classification would probably be needed to be able to attach a reasonable amount of these tokens to the correct head.

For the remaining incorrectly attached dependents, the most frequent dependent PoS tags are IN (10380) , RB (6421), NNP (6421) and NN (5060). These dependencies comprise 4.34% of the incorrect dependencies.

We know that 1.56% of the incorrectly attached dependents have been incorrectly reattached by us during the rewriting process. We believe that the rewriting algorithms applied, or at least some of them, can be refined so that the error rate is decreased. The potential for improvement is especially large for the *Measure noun phrases* procedure, and we believe that a more thorough examination of samples, will reveal sub-patterns that we have not identified.

In our rewriting procedure, we have only converted the multi-word expressions that are occasionally used as coordinating conjunctions. An amount of 3617 dependencies labelled *MWE* in the original DT data, are still incorrect. These dependencies involve some of the incorrectly attached IN and RB dependents. We believe that these dependencies, or at least a large part of them, could be successfully rewritten. This would increase the unlabelled attachment score with about 0.5.

We also believe that the RB tokens incorrectly used as roots, according to CD, could be identified and correctly reattached.

For some phenomena the structure of one gold standard analysis may not be very helpful for identifying the correct structure in another format. PP attachments and other adverbials are notoriously difficult to annotate correctly, as sentences with these constructions are often structurally ambiguous. This could explain the large number of incorrectly attached IN and RB nodes, for which we have not identified any distinct patterns for rewriting.

We find that 2870 dependents PoS-tagged as NNP are incorrectly attached to a NNP head with a dependency relation labelled NP-HDN in the DT data. 16269 NNP dependents are correctly attached to a NNP head by such a dependency relation. Investigating this further, we find several examples of structures as those illustrated in Figure 7.1 (DT above, CD below). It would seem that CD has a different analysis of the structures like the one in the example to the left and those like the one shown to the right, possibly based on some information from the PTB that tokens like *'Journal'* (in this use) function somewhere in between a proper and a common noun. Although this seems to be a regularity, it will be hard for us to separate the two patterns, by a heuristic method, with the information available to us in the DT data.

Of the 5060 incorrectly attached NN dependents, 1554 are attached to another noun. Very recently, Oepen et al. 2014 observe that PTB and DeepBank differ in their ambitions about the bracketing internal to compound nouns. As a result, the syntactic analysis will often be different

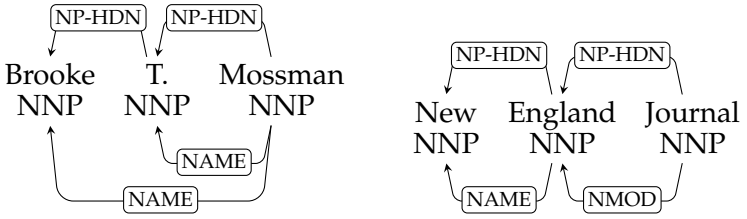


Figure 7.1: Example of structures with NNP tokens attached to another NNP token (DT above, CD below)

in DT and CD, in a way where the DT analysis will not be sufficient for identification of the correct CD structure. This could account for (at least a part of) these incorrectly attached NN nodes.

These three phenomena, as well as the commas mentioned earlier, are examples of structures where conversion from DT to CD by use of a heuristic method might not be feasible. For these structures a different approach to obtaining data in CD annotation might be more appropriate.

0.57% (3715) of the dependencies in our training data are non-projective in the CD gold data. These structures could of course overlap with some of the above mentioned incorrect dependencies. Other syntactic phenomena lurking among the incorrect attachments have yet to be discovered.

7.2.3 Labelling by classification

We have trained and tested the labelling classifier using default parameters. There might be potential for improvement in optimizing the classifier by adjusting its parameters for learning and optimization.

Bibliography

Aduriz, Itziar, Maria M. Aranzabe, Jose M Arriola, Atutxa Aitziber, Diaz de Ilarraza Arantza and Oronoz Maite (2003). *Construction of a Basque dependency treebank*.

Baker, Collin F, Charles J Fillmore and John B Lowe (1998). 'The berkeley framenet project'. In: *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*. Association for Computational Linguistics, pp. 86–90.

Bengoetxea, Kepa and Koldo Gojenola (2009). 'Exploring treebank transformations in dependency parsing'. In: *Proceedings of the International Conference on Recent Advances in Natural Language Processing, RANLP*.

Bos, Johan, Edward Briscoe, Aoife Cahill, John Carroll, Stephen Clark, Ann Copestake, Dan Flickinger, Josef van Genabith, Julia Hockenmaier, Aravind Joshi, Ronald Kaplan, Tracy Holloway King, Sandra Kuebler, Dekang Lin, Jan Tore Loenning, Christopher Manning, Yusuke Miyao, Joakim Nivre, Stephan Oepen, Kenji Sagae, Nianwen Xue and Yi Zhang, eds. (Aug. 2008). *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*. Manchester, UK: Coling 2008 Organizing Committee. URL: http://www.aclweb.org/anthology/W08-13.

Brants, Thorsten (2000). *TnT - A Statistical Part-of-Speech Tagger*.

Carroll, John, Ted Briscoe and Antonio Sanfilippo (1998). 'Parser evaluation: a survey and a new proposal'. In: *Proceedings of the 1st International Conference on Language Resources and Evaluation*, pp. 447–454.

Carroll, John, Guido Minnen and Ted Briscoe (1999). 'Corpus annotation for parser evaluation'. In: *arXiv preprint cs/9907013*.

Collins, Michael (1997). 'Three Generative, Lexicalised Models for Statistical Parsing'. In: *Proceedings of the Eighth Conference on European Chapter of the Association for Computational Linguistics*. EACL '97. Madrid, Spain: Association for Computational Linguistics, pp. 16–23. DOI: 10.3115/979617.979620. URL: http://dx.doi.org/10.3115/979617.979620.

De Marneffe, Marie-Catherine and Christopher D Manning (2008). 'Stanford typed dependencies manual'. In: *URL http://nlp. stanford. edu/software/dependencies_manual. pdf*.

De Marneffe, Marie-Catherine, Bill MacCartney, Christopher D Manning et al. (2006). 'Generating typed dependency parses from phrase structure parses'. In: *Proceedings of LREC*. Vol. 6, pp. 449–454.

ERG Lexical and Syntactic Rules. http://moin.delph-in.net/ErgRules. Accessed: 2014-01-13.

ERG Tags. http://svn.delph-in.net/erg/tags/1212/etc/rules.hds. Accessed: 2014-05-06.

Flickinger, Dan (2002). 'On Building a More Efficient Grammar by Exploiting Types'. In: *Proceedings of the Sixth Linguistic Annotation Workshop*. CSLI Publications, pp. 1–17.

Flickinger, Dan, Yi Zhang and Valia Kordoni (2012). 'DeepBank: A Dynamically Annotated Treebank of the Wall Street Journal'. In: *Proceedings of the Eleventh International Workshop on Treebanks and Linguistic Theories*. Edições Colibri, pp. 85–96.

Francis, Winthrop Nelson (1964). *A STANDARD SAMPLE OF PRESENT-DAY ENGLISH FOR USE WITH DIGITAL COMPUTERS.* ERIC.

Francis, Winthrop Nelson, Henry Kucera and Andrew W. Mackie (1982). *Frequency analysis of English usage*. Houghton Mifflin Company.

Graham, Yvette and Josef van Genabith (2009). 'An open source rule induction tool for transfer-based smt'. In: *The Prague Bulletin of Mathematical Linguistics* 91.1, pp. 37–46.

Hajic, Jan, Barbora Vidová-Hladká and Petr Pajas (2001). 'The prague dependency treebank: Annotation structure and support'. In: *Proceedings of the IRCS Workshop on Linguistic Databases*, pp. 105–114.

Hudson, Richard A (1984). *Word grammar*. Blackwell Oxford.

Ivanova, Angelina, Stephan Oepen and Lilja Øvrelid (2013). *Survey on parsing three dependency representations for English*.

Ivanova, Angelina, Stephan Oepen, Lilja Øvrelid and Dan Flickinger (2012). 'Who did what to whom?: a contrastive study of syntacto-semantic dependencies'. In: *Proceedings of the Sixth Linguistic Annotation Workshop*. Association for Computational Linguistics, pp. 2–11.

Johansson, Richard (2008). 'Dependency Syntax in the CoNLL Shared Task 2008'.

Johansson, Richard and Pierre Nugues (2007). 'Extended constituent-to-dependency conversion for English'. In: *Proc. of the 16th Nordic Conference on Computational Linguistics (NODALIDA)*, pp. 105–112.

— (2008). 'Dependency-based syntactic-semantic analysis with PropBank and NomBank'. In: *Proceedings of the Twelfth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, pp. 183–187.

King, Tracy Holloway, Richard Crouch, Stefan Riezler, Mary Dalrymple and Ronald Kaplan (2003). 'The PARC 700 dependency bank'. In: *Proceedings of the EACL03: 4th international workshop on linguistically interpreted corpora (LINC-03)*, pp. 1–8.

Kouylekov, Milen and Stephan Oepen (May 2014). 'Semantic Technologies for Querying Linguistic Annotations: An Experiment Focusing on Graph-Structured Data'. In: *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. Ed. by Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan

Odijk and Stelios Piperidis. Reykjavik, Iceland: European Language Resources Association (ELRA). ISBN: 978-2-9517408-8-4.

Magerman, David M (1994). 'Natural language parsing as statistical pattern recognition'. In: *arXiv preprint cmp-lg/9405009*.

Marcus, Mitchell P, Mary Ann Marcinkiewicz and Beatrice Santorini (1993). 'Building a large annotated corpus of English: The Penn Treebank'. In: *Computational linguistics* 19.2, pp. 313–330.

Marcus, Mitchell, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz and Britta Schasberger (1994). 'The Penn Treebank: annotating predicate argument structure'. In: *Proceedings of the workshop on Human Language Technology*. Association for Computational Linguistics, pp. 114–119.

McDonald, Ryan, Fernando Pereira, Kiril Ribarov and Jan Hajič (2005). 'Non-projective dependency parsing using spanning tree algorithms'. In: *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 523–530.

Miyao, Yusuke (2007). *Enju 2.2 Output Specifications*. Tech. rep. Technical Report TR-NLP-UT-2007-1, Tsujii Laboratory, University of Tokyo.

Miyao, Yusuke, Kenji Sagae and Junichi Tsujii (2007). 'Towards framework-independent evaluation of deep linguistic parsers'. In: *Proceedings of GEAF 2007*.

Nilsson, Jens, Joakim Nivre and Johan Hall (2006). 'Graph transformations in data-driven dependency parsing'. In: *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 257–264.

Nivre, Joakim (2005). *Dependency grammar and dependency parsing*. Tech. rep. Technical Report MSI report 05133, Växjö University: School of Mathematics and Systems Engineering.

Nivre, Joakim, Johan Hall and Jens Nilsson (2006). 'Maltparser: A data-driven parser-generator for dependency parsing'. In: *Proceedings of LREC*. Vol. 6, pp. 2216–2219.

Nivre, Joakim, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov and Erwin Marsi (2007). 'MaltParser: A language-independent system for data-driven dependency parsing'. In: *Natural Language Engineering* 13.2, pp. 95–135.

Nivre, Joakim and Jens Nilsson (2005). 'Pseudo-projective dependency parsing'. In: *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pp. 99–106.

Oepen, Stephan (1999). *[incr tsdb ()] Competence and Performance Laboratory. User and Reference Manual*. Tech. rep. Computional Linguistics, Saarland University, Saarbrücken.

Oepen, Stephan, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajič, Angelina Ivanova and Yi Zhang (2014). 'SemEval 2014 Task 8. Broad-Coverage Semantic Dependency Parsing'. In: *Pro-*

ceedings of the 8th International Workshop on Semantic Evaluation. Dublin, Ireland.

Pollard, Carl and Ivan A Sag (1994). *Head-driven phrase structure grammar*. University of Chicago Press.

Popel, Martin, David Marecek, Jan Štepanek, Daniel Zeman et al. (2013). 'Coordination Structures in Dependency Treebanks'. In: *ACL (1)*, pp. 517–527.

Schwartz, Roy, Omri Abend and Ari Rappoport (2012). 'Learnability-Based Syntactic Annotation Design.' In: *COLING*, pp. 2405–2422.

Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluis Màrquez and Joakim Nivre (2008). 'The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies'. In: *Proceedings of the Twelfth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, pp. 159–177.

SVMlight. http://svmlight.joachims.org/. Accessed: 2014-03-04.

Wilcoxon, Frank (1945). 'Individual comparisons by ranking methods'. In: *Biometrics bulletin*, pp. 80–83.

Yamada, Hiroyasu and Yuji Matsumoto (2003). 'Statistical dependency analysis with support vector machines'. In: *Proceedings of IWPT*. Vol. 3.

Zeman, Daniel, David Marecek, Martin Popel, Loganathan Ramasamy, Jan Štepanek, Zdenek Zabokrtsky and Jan Hajic (2012). 'HamleDT: To Parse or Not to Parse?' In: *LREC*, pp. 2735–2741.

Zwicky, Arnold M (1985). 'Heads'. In: *Journal of linguistics* 21.1, pp. 1–29.