UiO : Department of Informatics
University of Oslo

Linguistic Features in Data-driven Dependency Parsing of Norwegian

Miriam Næss Jørstad

Master's Thesis Autumn 2014

Acknowledgements

First, I would like to express my deepest gratitude to my supervisors Lilja Øvrelid and Arne Skjærholt for their patience and the guidance and encouragement which made the completion of this thesis possible. I am honored for having the opportunity to work with them.

My gratitude goes also to my fellow students, the program committee for Informatics: Language and Communication (P:ISK) and the staff of the Language Technology Group at the University of Oslo for creating a great and including learning environment.

I would also like to thank the developers of MaltParser and the Norwegian Dependency Treebank. I would like to give a special thank to Per Erik Solberg for answering all my questions about the treebank.

I am immensely grateful to my family, who have been great support.

Last, but not least, my gratitude goes to Ivar, for keeping me sane and for having a positive attitude through out the process.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

There are currently 7,106 living languages in the world (Lewis et al. 2014). Any speaker of a language knows tens of thousand of words and can create and understand an infinite number of sentences. At the same time anyone can learn thousands of words in a language without knowing or understanding the language. Words are an important part of linguistic knowledge and of our mental grammar. Each word we know includes information about its sound, meaning and grammatical category. Unless we have this kind of knowledge of a word, we would not have the ability to form grammatical sentences or recognize ungrammatical ones. This knowledge is what we know as grammar. Grammar can be described as the set of rules on the composition of clauses, phrases, and words in a natural language. Grammar is also the study of these rules and consists of several structural subfields such as morphology, syntax, phonology, phonetics, semantics, and pragmatics.

For most of us, the word grammar might be something we associate with primary school. We all know the properties of the different lexical categories. In our mental lexicon we know that nouns describe things, either abstract or concrete, such as people, places or ideas. We know that words describing things you can do are verbs. We know that adjectives are words that describe the noun and that adverbs describe verbs. That all of these words and their categories have different sets of properties and features, and an internal, rule-governed structure is a basic part of language learning. Words are combined with other words to form sentences and the relations between the words creates the meaning of a sentence.

In our daily life we use our knowledge of grammar unconsciously when we speak a language fluently. In the field of Natural Language Processing (NLP), where the focus is on the interactions between computers and human languages, the knowledge of grammar is an essential tool. In order to gain a greater understanding of the interaction between humans and computers and to optimize this interaction, one often try to mimic parts of human cognition through different processing tasks. Grammar, such as morphology and syntax helps us create programs we use in order to get a better understanding of "who did what to whom". This again enables us to come closer to an understanding of the patterns we can expect to find in the grammatical sentences of a language.

In January 2014 the National Library in Norway released a treebank for Norwegian, the Norwegian Dependency Treebank (NDT), where the syntactical analysis in the texts is based on dependency grammar. Dependency grammar is based on the idea that the syntactic structure of a sentence can be described in terms of the words in the sentence and binary asymmetric relations between these words. Treebanks like this exist for many other languages and has provided the resources necessary to gain a better understanding of the methods in NLP, however it is the first of its kind for Norwegian. This treebank provide the opportunity to create dependency parsers for Norwegian.

1.1 Thesis

In this thesis we have studied data-driven dependency parsing for Norwegian and the effect of linguistic features on parsing accuracy. Data-driven dependency parsing is increasingly popular and this has led to the development of a range of parsing systems, where we have used the MaltParser system. Out focus has been directed towards gaining a better understanding how different linguistic features, such as person, gender and tense, influence the parsing performance. This kind of knowledge exists for most Scandinavian languages, but not for Norwegian, and we used the existing knowledge as a guide. To understand how to best use these features during parsing of Norwegian we conducted several experiments using lexicalization, morphological features and other linguistic features available in the NDT.

1.2 Thesis outline

Chapter 2 describes the theoretical background, concepts and systems used in this thesis. We will give an introduction to the theory of dependency grammar and explain the concepts of data-driven dependency parsing. We will also present the MaltParser system, MaltOptimizer, the system for MaltParser optimization, and the Norwegian Dependency Treebank (NDT).

Chapter 3 begins with a precise description of the experimental protocol for the experiments conducted in this thesis. Then we introduce our initial experiments where we use the MaltParser system with its default settings and present an error analysis performed. Finally, we present the results from running MaltOptimizer with the NDT.

Chapter 4 describes the experiments performed while working with this thesis and the results we obtained. We will first cover our experiments with extended part-of-speech tags before we present the experiments with morphological features. After that we present experiments performed with various linguistic features. Finally, we will present a final evaluation performed on a held-out test set and and an experiment where we investigate domain sensitivity.

Chapter 5 sums up the outcomes of our experiments and we present our main findings. We briefly discuss the work we have done and consider the possibilities for future work.

Chapter 2

Background

In this chapter we describe the theoretical background and concepts of this thesis and give an introduction to the systems and the resources used for the experiments presented in the following chapters. We begin by introducing the theory of dependency grammar and the Norwegian Dependency Treebank. After that we introduce data-driven dependency parsing and explain this approach in more detail by presenting MaltParser, the system used for data-driven dependency parsing in this thesis. We will then give a brief introduction to the MaltOptimizer system for MaltParser optimization.

2.1 Dependency grammar

There are several well-known theories of dependency grammar and its roots can be traced back to Panini's grammar of Sanskrit from the 6th century B.C. For a long time the tradition was been given little attention in both linguistics and NLP, but during the last decades there has been an increased interest in the tradition. The modern theoretical traditions of dependency grammar are primarily based on the dependency grammar developed by Lucien Tesnière. Nivre (2005) translates Tesnière's description of the idea behind dependency grammar:

> "The sentence is an organized whole, the constituent elements of which are words. Every word that belongs to a sentence ceases by itself to be isolated as in the dictionary. Between the word and its neighbors, the mind perceives connections, the totality of which forms the structure of the sen-

> tence. The structural connections establish dependency relations between the words. Each connection in principle unites a superior term and an inferior term. The superior term receives the name governor. The inferior term receives the name subordinate. Thus, in the sentence *Alfred parle* [. . .], *parle* is the governor and *Alfred* the subordinate."

This grammar representation differs a lot from the most influential grammar formalism in the field, phrase structure grammar by Chomsky (1957). In this formalism the idea is that a set of phrase structure rules can represent the hierarchical and linear order of a sentence, which is presented by a syntactic tree. In the syntactic tree the words in the sentence are the terminal nodes which are grouped together forming phrases.

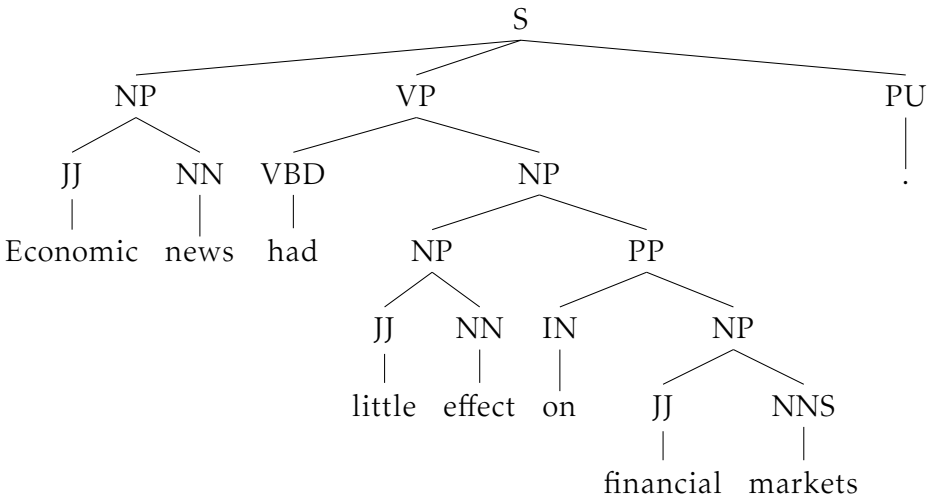

Figure 2.1: An example of a phrase structure tree for an English sentence taken from the Penn Treebank (Nivre 2005).

An example of a phrase structure can bee seen in figure 2.1. Here we see that the words "Economic new had little effect on financial markets." are the terminal nodes combined into non-terminal nodes representing the phrase structure.

Dependency grammar on the other hand represents syntactic structure as words linked by binary, asymmetrical relations called dependency relations or just dependencies. A dependency relation holds between a su-

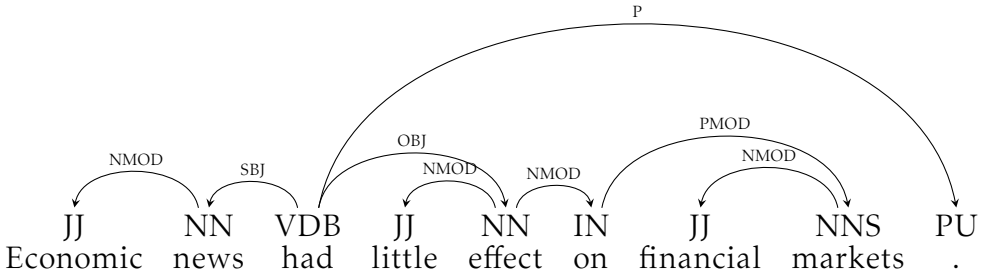

Figure 2.2: An example of a dependency structure for an English sentence taken from the Penn Treebank (Nivre 2005) .

perior word called the head (the governor) and its subordinates, named dependents (or governed) (Nivre 2005).

Figure 2.2 is an example of a dependency structure, and we can see that the dependencies are between words and not between phrases as in figure 2.1. The relations are represented by arrows pointing from the head to its dependents. Each arrow is given a label indicating the syntactic category of the dependency relation.

Even though the syntactic relations in dependency grammar are between the words in a sentence, and not between phrases such as in phrase structure grammar, the difference between these two formalisms only concern what is encoded in the different representations. Phrases can be distinguished in a dependency structure and a we can identify functional relations such as subjects and objects in phrase structures. However, (Kübler et al. 2009) stress that parts of the task of converting from one type of representation to the other is non-trivial. Some theories use a combination of dependency structure and phrase structures so even though they might seem very different they are not mutually exclusive approaches to natural language syntax.

To determine which is the head and which is the dependent in a relation, several criteria have been suggested, both syntactic and semantic. Based on the work of (Zwicky 1985) and (Hudson 1990), Nivre (2005) suggest the following criteria for identifying the syntactic relation between the head H and the dependent D in a construction C:

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*.

For example in figure 2.2, we find that the verb *had* decides the position and the semantic category of both *news* and *effect* because it requires something to be had and something to have. By relying on the property of valency we find it to be the head of both *news* and *effect*. Another example of the criteria in use is the direction in the relation between *financial* and *markets*. By using the first criteria in the list above, we can remove *Financial* without disrupting the syntactic structure, but is not the case for the relation between *news* and *had*.

A dependency structure can be represented as a labeled directed graph, due to it consisting of lexical elements linked by binary asymmetrical relations. A labeled directed graph is a graph where the nodes are connected by labeled arcs that have a direction associated with them. The set of lexical elements can be represented as the nodes, and the labeled arcs can represent the dependency relations from the heads to their dependents. In figure 2.2 we can see that the arc between the words *had* and *news* carries the label SBJ, indicating that *news* is the subject of the sentence. We represent the dependency relation between two nodes *i* and *j* with the arc (*i*, *j*), where *i* is the *head* and *j* is the dependent of the arch (*i*, *j*). $i \rightarrow j$ is used to represent that there is an arc connecting *i* and j. The notation $i \rightarrow^* j$ is used to represent the reflexive and transitive closure of an arc relation, meaning $i \rightarrow^* j$ iff $i = j$ or there is a path of arcs connecting *i* to *j* (Nivre et al. 2007).

In most formalisms of dependency grammar we find some basic constraints supporting this notion of representing the dependency graph as a rooted tree. These are constraints regarding:

1. Connectedness.

2. Acyclicity.

3. The number of heads.

The constraints regarding *connectedness* comes from the need of every node being related to at least one other node in order to create a complete syntactic analysis of a sentence.

There is also a common assumption that the graph should not contain any cycles. This is captured by the *acyclic* constraint, meaning that a word can not be both the head of and dependent on the same word. Formally, if we let i be the head in a relation and j is the dependent, then if $i \rightarrow j$ then not $j \rightarrow^* i$ (Nivre et al. 2007).

Most theories also assume that each node has only one head, this is called the *single-head constraint*. Meaning, that each node has at most one head. Usually, the only node without a head is the root node. Formally, every node in the graph G, except the root node, has *one head*, meaning if $i \rightarrow j$ then there is no node k such that $k \neq i$ and $k \rightarrow j$. Looking at figure 2.2 we can see that each node in the tree is connected to at least one other node. It is also easy to see that there are no cycles in the representation and each node has at most one head with the exception of the root *had*. These constraints are assumed by most theories of dependency grammar, but some theories, such as the word grammar theory by Hudson allow multiple heads and cyclic graphs (Nivre 2005).

The formal representation of dependency grammar is one of the most debated issues in the field, namely how to represent the relation between dependency structure and word order. Nivre (2005) explains that while dependency relations have a structural order, strings of words have a linear order. Most theories in dependency grammar assumes what Tesnière proposed, that "the nodes of a dependency structure are not linearly ordered in themselves but only in relation to a particular surface realization of this structure"(Nivre 2005).

This takes us to the *projectivity constraint*. A dependency graph is projective if, when we put the words in their linear order, "preceded by the root, the edges can be drawn above the words without crossing, or equivalently, a word and its descendants form a contiguous substring of the sentence" (McDonald et al. 2005). Formally, Nivre et al. (2007) explains

that a graph is projective "if $i \rightarrow j$ then $i \rightarrow^* k$, for every node k such that $i < k < j$ or $j < k < i$".

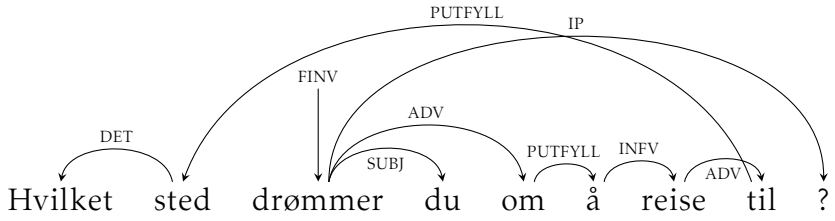

Figure 2.3: A non-projective dependency tree from the NDT that assumes non-projectivity (Kinn et al. 2013).

In figure 2.3, the word *sted* is separated from the head *til*. A dependency tree for this sentence can, as we can see from figure 2.3, only be drawn with edges crossing each other. In Norwegian grammar projectivity is an issue when topicalization occur and with sentences that contain multiple clauses (Kinn et al. 2013). The distinction between projective and non-projective dependency grammar comes down to whether the constraint of projectivity is assumed or not. Most theoretical formulations of dependency grammar consider projectivity as the norm, but also recognize the need for non-projective representations. Most also consider the constraint of projectivity to be too rigid for the description of languages with free word order (Nivre 2005).

2.2 The Norwegian Dependency Treebank

The Norwegian Dependency Treebank (NDT) is a so-called gold standard corpus. It is the result of a project by Språkbanken at the Norwegian National Library in collaboration with the Text Laboratory and the Department of Informatics at the University of Oslo. The Treebank is a syntactic treebank divided into two parts since there are two written standards of Norwegian — bokmål and nynorsk. The version in bokmål consists of 311 000 tokens while the version in nynorsk consist of 303 000 tokens (Solberg et al. 2014). Like comparable treebanks for other languages such as the Prague Dependency Treebank and the TIGER Corpus, the NDT consist mainly of newspaper text. In addition to this source, it also contains text from government reports and transcripts from parliament debates, in addition to selected posts from individual bloggers.

Bokmål	Nynorsk
Newspapers	
Bergens Tidende	Firda
Dagbladet	Vest-Telemark Blad
Klassekampen	Klassekampen
Sunnmørsposten	
Verdens Gang	
Blogs	
Pias Verden	Interessert? (Hallvards blogg)
Frøken Makeløs	Alt godt
Hc Svnt Dracones	Pur Glede
Kongen av Briskeby	
Breddefotballfrue	
Her på sandaker	
Government reports	
NOU 2006:2, chapter 1	St. meld. nr. 35 (2007-2008), chapter 1
NOU 2006:2,,chapter2	St. meld. nr. 35 (2007-2008),chapter 2
NOU 1999:2, chapter 3	St. meld. nr. 35 (2007-2008), chapter 3
NOU 1999:2, chapter 11	
Parliament debates	
Møte onsdag den 12. januar kl. 10	Møte mandag den 6. oktober 2008 kl. 10.05
Møte onsdag den 19. januar kl. 10	Møte torsdag den 12. desember kl. 10
Møte onsdag den 22. mars kl. 10	Forhandlinger i Stortinget nr. 74

Table 2.1: Text sources for NDT (Solberg 2013)

In table 2.1 we can see the different sources of text[1]. The newspaper text is taken from the *The Norwegian Newspaper Corpus (NNC)* [2]. The blog texts were acquired with permission from the individual bloggers, who represent a great variety in blog themes and genres. The NDT is available in both CoNLL format [3], and in Prague Markup Language (Solberg 2013).

[1] Parliament debates: Not all of the text on these pages were used.

[2] `http://avis.uib.no/avis/om-aviskorpuset/english`

[3] For more information regarding the CoNLL format and the reformatting of the treebank see appendix B.

Each token in the corpus contains information about morphological features, syntactic functions and hierarchical structure. The morphological analysis in the corpora follows Faarlund (1997), while the syntactic analysis is based on dependency grammar as described above. The annotation guidelines was developed by the annotators of the treebank (Solberg et al. 2014) and the annotation was manually done by trained linguists. In order to reduce inconsistencies, parts of the text were syntactically annotated by two annotators. A set of experiments done to validate consistency of the annotations shows that the agreement of the treebank gives a score of an α of about 98%, which is extremely high (Solberg et al. 2014). Before annotating the text, part-of-speech tags were added automatically and the text was syntactically parsed. This is a standard practice when annotating syntactic corpora, and the method has been proven to be effective and to provide high quality annotation (Solberg et al. 2014). After the morphological annotation was checked and corrected manually, it was preprocessed by a dependency parser and imported into TrEd, an annotation tool used to correct output from the syntactic preprocessing and create a final treebank (Solberg et al. 2014).

2.2.1 Annotation

As mentioned above, the NDT contains both morphological and syntactic annotation. The annotators of the treebank followed four fundamental principles when creating the annotation guidelines:

1. The annotation should be as linguistically adequate as possible
2. The annotators must be able to to implement the analyses consistently.
3. It must be possible for the annotators to annotate quickly.
4. It must be simple to retrieve specific constructions after annotation.

The morphological annotation follow the Oslo-Bergen Tagger, with some additional morphological tags. The lemmas are taken either from Norsk Ordbank, a lexicographic database for Norwegian, or generated by the Oslo-Bergen Tagger. The morphological tagset of the Oslo-Bergen tagger contains information regarding features related to the inflection of words and whether a token belongs to a certain sub-class of a part-of-speech.

An example of the last is that pronouns are marked as demonstrative, personal, reflexive, etc. while determiners are marked as demonstrative, possessive, etc. There is no distinction between the coarse-grained and fine-grained part-of-speech tags. When the spelling or inflectional form of a token in the corpus does not comply with the official norm of either bokmål or nynorsk, the annotators added the lemma manually and provided the correct tags in addition to the morphological tag *unorm* (short for *unormert*, translates to non-standard) (Solberg 2013), (Solberg et al. 2014). This was also the procedure if a non-compound did not exist as an entry in Norsk Ordbank.

The syntactic annotation guidelines were developed specifically for this treebank by the annotators as an iterative process during the beginning of the project. The annotations of dependencies were to a large extent influenced by choices made in the construction of similar treebanks, such as the Swedish treebank (Talbanken) and the treebank of the old Indo-European languages (PROIEL) (Solberg et al. 2014).

As mentioned earlier the theories agree about some fundamental properties of dependency, but they disagree whether the notion of dependency is sufficient for the analysis of syntactic structures in natural language (Nivre 2005). One issue that divides the different theories is whether they use single-layer or multi-layer frameworks. Single-layer frameworks are found in theories that rely on a single syntactic representation. Multi-layer frameworks are found in theories that use several layers of representation. Most theories of dependency grammar are multi-layer according to Nivre (2005), at least if they consider semantic representations to be a layer. Another distinction between the theories of dependency grammar is the dependency types, that is, the functional categories that are used to label the arcs between the relations in the representation. Most theories use a set of surface-oriented grammatical functions, such as subject, object and adverbial. The NDT is an example of a corpus where the dependency types are based on such grammatical functions (Kinn et al. 2013). Other theories posit sets of semantically oriented types from the tradition of case roles or thematic roles, such as agent, patient and goal. Multi-layer theories often combine these two dependency types, while other theories combine numerical indices and descriptive labels (Nivre 2005).

In dependency grammar there are no unique answers to how relationships such as the head-dependent relationship between complementizers

and verbs, or between function words and lexical words, should be represented. There are several suggestions about what the criteria should be, and this has led to a number of different theories. Some suggest that the concept of head has a prototype structure, while others focus on the need to distinguish between different kinds of dependency relations. One disagreement between the theories is whether a functional word can take the role as a head of lexical word or if it should be the other way around (Kinn et al. 2013). A lexical head belongs to one of the categories noun, verb, adjective and sometimes adverb. These are words that carry meaning. A functional head on the other hand will often be a function word such as a determiner or inflections (Falk 2001).

In the annotation standards from Stanford the lexical word will be given the head-status whenever that is possible. The standard CoNLL conversion of the Penn Treebank on the other hand varies between giving the head-status to the lexical word and function words. In Norwegian complementizers are frequently dropped and therefore, in the NDT, the verb is the head of the complementizer, making the complementizer dependent on the verb. The same principle goes for the noun-determiner relation and for coordination; if possible, the lexical word will be the head. Nouns take determiners as dependents with the function DET. There are of course exceptions. A well-known case where the function word is given the head-status is when a sentence has a finite auxiliary and a lexical verb. In this case the finite auxiliary will be the head of the lexical word, which will be given the function INFV (Solberg et al. 2014).

As an example of a different approaches to determine what should be head and what should be dependents is the distinction between the Danish Treebank and the NDT. A relation between a determiner and a noun will by the NDT's annotation give the noun the role as the head, making it the head, and the determiner will take the role as the dependent, as shown in figure 2.4 (Kinn et al. 2013). The annotation of the Danish Treebank on the other hand, allows the determiner to take the role as the head of a noun, as you can see in figure 2.5 (Kromann and Lynge 2004).

Another problematic construction in dependency grammar is coordination. Coordination is a symmetrical relation, while dependencies, by definition, are asymmetric. Theorists solve this issue in different ways and the annotators of the NDT has solved the issue of coordination by making the first conjunct the head. It also carries the grammatical function for the entire coordinated structure. Conjuncts that are dependent on the

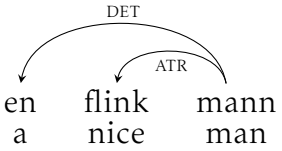

Figure 2.4: Relation between determiner and noun based on the annotation of the NDT.

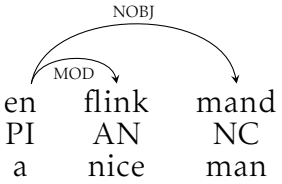

Figure 2.5: Relation between determiner and noun from the Danish Treebank (Kromann and Lynge 2004).

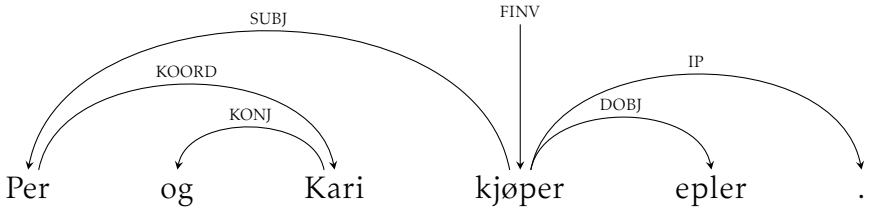

Figure 2.6: Coordination from the NDT (Kinn et al. 2013).

first will receive the function KOORD, while the conjunctions being dependent on the closest conjunct to the right are given the function KONJ. Figure 2.6 is an example of coordination where the relation between *Per og Kari* (Per and Kari) is symmetric.

Unlike other treebanks the annotators of the NDT decided to opt for a relatively shallow analysis of adverbials, therefore, regardless of type, and whether or not they are selected, all adverbials are given the dependency relation ADV. One reasons for this is that a higher level of linguistic detail would make it more difficult for the annotators to implement the analyses consistently and it would also make the annotation process more time consuming. Another reason is that a more fine-grained

analysis of adverbials could make it difficult to infer grammatical patterns and extract meaningful information. This due to the distinction between different types of adverbials often being based on semantic or pragmatic considerations, and not on the difference in syntactic structure. Other constructions given the function ADV are dependents modifying the verb, such as prepositions, adverbs, subordinate clauses and adjectives. E.g. when a verb requires a prepositional phrase, the preposition will be given the ADV function. ADV is also used on construction which modifies adjectives, determiners, adverbs and again, prepositional phrases. Nouns will usually not take ADV dependents, with the exception of constructions where we assume that a verb is omitted.

The tag used for descriptive dependents of nouns is the ATR function. Together with DET, ATR is a descriptor and a determiner. The DET function is mostly used for determiners while the ATR function is primarily a descriptor. When dependents on noun, determiners will usually be analyzed as DET, while adjectives will be analyzed as ATR. ATR can also be supplemental information, age specification or prepositions. When it comes to prepositions, it can be difficult to determine whether it should be ATR on a noun or ADV on a verb. We can see this being a problem from the table of errors on page 35. ATR is also used on relative clauses being dependent on nouns/pronouns and for particles following a noun. Kinn et al. (2013) stress in the annotation guidelines that a preposition will be analyzed as ATR on the determiner in the cases where a determiner is followed by the preposition *av* ('of') and its dependent express a partitive relation.

2.3 Dependency parsing & MaltParser

2.3.1 Data-driven dependency parsing

Parsing is the process of deriving a syntactic structure from an input string and dependency parsing is the process of automatically analyzing the dependency structure of a given input string. Nivre (2006) distinguishes between two different notions of parsing, *grammar parsing* and *text parsing*. *Grammar parsing* can be described as parsing using formal grammars, while *text parsing* on the other hand, can be described as parsing an unrestricted text in natural language L. Nivre (2006) uses this definition of the parsing task: Given a text $T = \{x_1, \ldots, x_n\}$ in L, derive the correct analysis for every sentence $x_i \in T$.

In the field of text parsing Nivre (2006) distinguishes between two complementary but quite different methodological strategies; a grammar-driven approach and a data-driven approach.

The grammar-driven approach to text parsing is based on the idea that a language natural L can be defined by a formal grammar G. The assumption is that by parsing the language *L(G)* we will get a class of analyses in return for each string in the language. A crucial assumption, according to Nivre (2006), in this approach, is that the language *L(G)* is a reasonable approximation of the language L that we desire to process. We will not go into more details about this approach in this thesis since our focus is the other approach, namely data-driven text parsing.

Data-driven text parsing needs no formal grammar. In this approach the mapping from the input strings to the analyses is done by an inductive mechanism applied to a text $T = \{x_1, \ldots, x_n\}$, also called the *training data*, from the language L to be analyzed. The approach consists of three main components which Nivre (2006) describes like this:

1. A formal model M which defines the possible analyses for the sentences in L.

2. A sample of text $T = \{x_1, \ldots, x_n\}$ from L. It may or may not be annotated with representations satisfying the constraints of M.

3. An inductive inference scheme I which defines the actual analyses for the sentences in a text $T = \{x_1, \ldots, x_n\}$ in L, relative to M and T[4].

The system used for the experiments presented in this thesis, MaltParser, is a system for data-driven dependency parsing and has obtained good results[5] across a variety of languages without any language-specific enhancements. We will give a short introduction to the system and the method before we explain this approach to parsing and the components of the MaltParser system in more details.

The methodology of the MaltParser is based on three components:

[4] I is usually based on *supervised machine learning* if T is annotated. If T consists of raw text one can use *unsupervised machine learning*.

[5] A dependency accuracy in the range of 80-90 %

1. Deterministic parsing algorithms used to construct dependency graphs.
2. History-based feature models used to predicting the next action of the parser.
3. Discriminative machine learning, mapping decisions made in the past to parser actions.

The system learns from sets of data, dependency treebanks, and from this information it derives a syntactic analysis for each given sentence. Being a deterministic parser it derives only a single analysis for each input string. MaltParser uses the information it acquires as a guide when it runs into non-deterministic choice-points (Nivre et al. 2007). In order to perform disambiguation deterministically MaltParser uses a classifier trained on a gold standard treebank. This approach is very different from many other approaches in statistical parsing, which are based on non-deterministic parsing techniques.

MaltParser is a what we call a transition-based parsing system, using a greedy parsing algorithm where the search is based on a series of locally optimal decisions that approximate the optimal solution. An example of a contrasting system, MSTParser, a global graph-based parsing system that uses a near exhaustive search (McDonald and Nivre 2007). In other words, MaltParser uses a greedy search algorithm that determines the best parsing decision based on the trained classifier and the current parser history, whereas MSTParser uses an exhaustive search algorithm that chooses the best dependency graph out of all possible combinations. The exhaustive search might lead to more accurate parsing results than the greedy search, but the greedy search algorithm performs with a lower complexity.

With Nivre's algorithm, MaltParser is guaranteed to terminate in $O(n)$, where MSTParser's exhaustive search algorithms gives it a time complexity of at least $O(n^2)$. Another advantage of MaltParser is that it enables the use previously predicted dependency relations as features to predict the current relation. However, this can also be a disadvantage leading to error propagation. Due to the exhaustive inference MSTParser can only use the score of features of one or two close parsing decisions (Zhang and Clark 2008). One last advantage of deterministic parsers is that although the accuracy of more complex statistical models trained on large amounts of data is a bit higher than the accuracy of a deterministic

parser, the deterministic parser will often have a steeper learning curve and can give a higher accuracy when training on small data sets (Nivre et al. 2007).

As mentioned above, the syntactic analysis of a sentence in dependency parsing is represented by a dependency graph. The dependency graph is a labeled directed graph where each node corresponds to a token in the sentence. Given a set $R = \{r_0, r_1, \ldots, r_m\}$ of arc labels (possible dependency relation types) and a sentence $x = (w_i, \ldots, w_n)$, a dependency graph G is a directed graph $G = (V, A, L)$ where:

1. $V = \{w_i, \ldots, w_n\}$
2. $A \subseteq V \times V$
3. $L : A \rightarrow R$

In each sentence, a token index i represents the node corresponding to w_i. There is also a special node 0, the root of the dependency graph. The set V consist of the nodes, while the set A represents the arcs for a set of ordered pairs (i, j), where i is the head in the relation and j is the dependent. As mentioned earlier,Nivre et al. (2007) use the notion $i \rightarrow j$ to represent the arc connecting i and j ($(i, j) \in A$). Kübler et al. (2009, p. 12) illustrates the definition of a dependency graph using figure 2.2 such: $V = V_x = \{$*Economic, news, had, little, effect, on, financial, markets, .*$\}$ and $A = \{$ *(had, SBJ, news), (had, OBJ, effect), (had, P, .), (news, NMOD, Economic), (effect, NMOD, little), (effect, NMOD, on), (on, PMOD, markets), (markets, NMOD, financial)*$\}$ L is a function that assigns a dependency type (arc label) $r \in R$ to each arc $a \in A$, the notation $i \xrightarrow{r} j$ represent the arc label r connecting i to j.

MaltParser assumes that a *well-formed* dependency graph G for a sentence x have the following properties:

1. The node 0 is the root node.
2. The graph G is *connected*.
3. Every node, except the root node, in the graph G has *one head*.

4. The graph G is *acyclic*.

5. The graph G is *projective* if the algorithm used is limited to projective dependency structures.

2.3.2 Deterministic parsing

The most common algorithms used for dependency parsing are different variants of the shift-reduce algorithm. These are bottom-up algorithms analyzing an input sentence from left to right using a queue of input tokens and a stack to store the partially processed tokens. Kübler et al. (2009, p. 21) describes the process of deterministic dependency parsing as creating a valid dependency graph for a given sentence by "complex configurations with internal structure [...] and transitions that correspond to the steps in the derivation of a dependency graph". The process of deriving the dependency graph is a sequence of transitions, starting in the initial configuration for the sentence, and ending in a terminal configuration.

Nivre et al. (2007) stress that most of the algorithms used for practical dependency parsing are restricted to projective dependency graphs, but all deterministic parsing algorithms can be added to the MaltParser system if they are compatible with the system's architecture (Hall et al. 2012). The latest version of MaltParser (1.7.2) at the time when we conducted our experiments implements three types of parsing algorithms:

1. **Nivre**: A linear-time algorithm limited to projective dependency structures. It exists in an arc-eager and an arc-standard version.

2. **Covington**: A "quadratic-time algorithm for unrestricted dependency structures" (Hall et al. 2012). It allows both projective dependency and non-projective (but acyclic) structures.

3. **Stack**: Algorithms similar to Nivre's algorithm, but are able to derive non-projective dependency trees.

As an example, we will describe Nivre's arc-eager algorithm in detail. This algorithm is just one of the algorithms available, we use it to illustrate the general principles. To explain the parsing algorithm we first define its parser *configuration* for a sentence $x = (w_1, \ldots, w_n)$ relative to

$R = \{r_0, r_1, \ldots, r_m\}$, the set of dependency types. Given x and R, Nivre et al. (2007) define the parser *configuration* $c = (STACK, INPUT, H, D)$ for x is like this:

1. *STACK* is a stack of token nodes i ($1 \leq i \leq j$ for some $j \leq n$).
2. *INPUT* is a sorted sequence of token nodes i ($j < i \leq n$).
3. H is a function from token nodes to nodes.
4. D is a function from token nodes to dependency types.
5. For every token node i in the token nodes, $D(i) = r_0$ iff $D(i) = 0$.

A parser configuration represents a partial analysis of the given input sentence. When the parser passes from left to right over the sentence the remaining input tokens will be stored in *INPUT*, while *STACK* contains the partially processed token nodes. The functions H and D represents the dependency graph for the sentence. The parser is initialized with all the nodes of the given sentence yet to be processed, at this point the stack is empty. In the dependency graph all the token nodes are dependents of the root node while the arcs all carry the default label r_0. The parser will conduct one left-to-right pass over the sentence, which will leave the list of input tokens empty, resulting in the termination of the parser. Nivre et al. (2007) describes the way of connecting the configurations and the dependency graphs such:

The configuration $c = (STACK, INPUT, H, D)$ for $x = (w_1, \ldots, w_n)$ will define the dependency graph $G_c = (V_x, A_c, L_c)$, where

1. $A_c = \{(i, j) | H(j) = i\}$.
2. $L_c = \{((i, j), r) | H(j) = i, D(j) = r\}$

A configuration c for $x = (w_1, \ldots, w_n)$ is *initial* if it has the form $c = (\epsilon, (1, \ldots, n), H_0, D_0)$ where $H_0(i) = 0$ for every $i \in V_x$ and $D_0(i) = r_0$ for every $i \in V_x$ The configuration c is *terminal* if it has the form $c = (STACK, \epsilon, D, H)$ for arbitrary STACK, H and D.

The set C represents all the possible configurations relative to the set R of dependency types. A *transition*, which is a partial function, maps non-terminal configurations to new configurations. Nivre's parsing algorithm uses four transitions. The possible transitions for every $r \in R$, given a set of dependency types R is by Nivre et al. (2007) described as:

Left-Arc(r) is the transition applying a dependency type r to the relation between the top token i on *STACK* and the first token j in *INPUT*, $j \xrightarrow{r} i$. It also pops the stack if token i is complete with respect to the left and right dependents. Both *STACK* and *INPUT* must be non-empty and i can not be the root for this transition to be applied.

The transition *Right-Arc(r)* also add a dependency type r to the relation between the top token i on *STACK* and the first token j in *INPUT*, $i \xrightarrow{r} j$. It then pops the stack and replaces j by i in the top of the *INPUT*. At this point j will be complete with respect to its left dependents but it might need new dependents to the right. This transition will only be applied if both *STACK* and *INPUT* are non-empty.

Reduce pops the stack. The system depends on this transition to pop nodes that were pushed in a *Right-Arc(r)* transition, and are complete in terms of its right dependents.

The *Shift* transition on the other hand pushes the next token i into the stack. The transition is needed to process the nodes that have their heads to the right and those nodes still attached to the root node (as when initialized).

This transition system is non-deterministic since most of the time there is more than one transition possible for a configuration. In order to make the system deterministic it is implemented with a mechanism which predicts the next transition at each non-deterministic choice point. The mechanism is also used to to choose a dependency type r for the *Left-Arc(r)* and *Right-Arc(r)* transitions. Nivre et al. (2007) calls this mechanism an *oracle*. There are several alternative algorithms for the oracle, for both projective and non-projective parsing.

2.3.3 History-based feature models

As Kübler et al. (2009) writes "Of course, oracles are hard to come by in real life, so in order to build practical parsing systems, we need to find some other mechanism that we can use to approximate the oracle well enough to make accurate parsing feasible." In the case of MaltParser, the oracle is a classifier. The oracle uses history-based feature models and discriminative machine learning to decide the next action when constructing a dependency graph. That is, it use the features of a partially built dependency structure with combined with the features of the input string to predict the next action when deriving a dependency structure. History-based models are widely used for part-of-speech tagging and syntactic parsing and was first introduced to natural language processing by Black et al. (1992). During the last decade it has completely replaced grammar for a large number of parsers (Nivre et al. 2007).

The basic idea behind history-based models in natural language models is to make a pair (x, y) of an input string x and an analysis y. Each pair (x, y) is mapped to a sequence of decisions $D = (d_i, \ldots, d_{i-1})$. Because MaltParser uses a deterministic parsing strategy this requires only that we estimate the mode of each conditional distribution. The *history* is the conditioning context for each $d_i, (d_1, \ldots, d_{i-1})$, corresponding to the transitions in the case of MaltParser. *Feature vectors* are distinct parser histories represented as series of attributes, and a *feature model* is a sequence of *feature functions* where all relevant features of the history is identified by a function. Nivre et al. (2007) describe the most important features in dependency parsing as the attributes of the input tokens, such as dependency type, part of speech and word form. These attributes are either *static* or *dynamic*. Static attributes (POS, word form, etc.) are the same during the parsing of a sentence, while dynamic attributes (dependency types) on the other hand are defined by the partially built dependency graph.

When we define complex history-based feature models, we refer to "attributes of arbitrary tokens in the parser history, represented by the current parser configuration" (Nivre et al. 2007). To do so we must first introduce a set of address functions: Given a sentence $x = \{w_0, w_1, \ldots, w_n\}$ and a parser configuration $(STACK, INPUT, H, D)$ for x:

1. $STACK_i$, is the ith token from the top of the $STACK$.

2. $INPUT_i$, is the ith token in the remaining input.

3. H_i, is the head of a token i in the graph defined by H.

4. $l(i)$, is the leftmost child of token i in the graph defined by H.

5. $r(i)$, is the rightmost child of token i in the graph defined by H.

MaltParser defines complex feature models by applying *attribute functions*, functions selecting specific attributes of a token, to combinations of address functions, where we typically use part-of-speech features and lexical features with dependency type features. In inductive dependency parsing the feature models used will usually combine static features and lexical features with dynamic dependency features.

2.3.4 Discriminative machine learning

The system uses discriminative learning methods, meaning that it optimize the mapping from input in $x \in X$ to output in $y \in Y$ by estimating the conditional probability distribution $P(X|Y)$. This is done instead of estimating a full generative model for the joint distribution of X and Y. Together with the deterministic parsing strategy, the learning problem becomes, as mentioned earlier, a classification problem, taking the histories (feature vectors) as input and returning the parsing decisions in classes (Nivre et al. 2007).

The learner relies on training data, which is dependency graphs generated from gold standard treebanks, such as the NDT. From the graphs it can reconstruct the correct transitions and extract the correct feature vectors for each configuration, as explained earlier in this chapter. The training data consist of pairs of parser configuration defined by the feature model and the correct transition.

Nivre et al. (2007) mentions that the learning problem posed by inductive dependency parsing can be solved by any learning algorithm capable of inducing a classifier from labeled training data, however most of the work in this area has been based on support vector machines (SVM) and memory-based learning (MBL). MaltParser provides two built-in learners LIBSVM and LIBLINEAR. LIBSVM is a software for classification by support vector machines with different kernels. It usually obtain the best accuracy of the two learners but its time complexity is $O(n^2)$ or $O(n^3)$.

LIBLINEAR only has a time complexity of $O(n)$ due to using linear classification.

2.3.5 Pseudo-projective parsing

Nivre's algorithm is, like most algorithms for dependency parsing, restricted to projective dependency graphs. Meaning that it should be combined with algorithms for pseudo-projective parsing if it were to be used for a treebank with non-projective dependency relations. As mentioned earlier one of the criteria for a *well-formed* dependency graph is that it is *projective*. Most theories of dependency grammar consider projectivity as the norm, but understand the need for non-projective representations, particularly for languages with a free word order where the constraint of projectivity is too rigid for the description of the language (Nivre 2005). Some claim that one of the advantages of dependency grammar is that it is more suitable to represent languages of free word order. This means that systems for dependency parsing must handle non-projective structures in order to come closer in the task of reaching the full potential of dependency-based syntactic parsing (Nilsson and Nivre 2005).

MaltParser is implemented with a system for *pseudo-projective parsing*, which allow us to derive non-projective graphs even when the algorithms are restricted to projective structures. The system operates in 4 steps:

1. The system *projectivizes* the dependency graph in the data set used for training and encodes the information about any transformations in arc labels.
2. The parser is trained using the training set.
3. The projective parser parses new sentences.
4. The system *deprojectivizes* the output generated by the projective parser, guided by the information in arc labels.

In other words, the system transforms a non-projective dependency graph to a projective dependency graph by replacing every non-projective arc *(i, r, j)* by an arc *ANC(i), r', j)* where *ANC(i)* represents the ancestor of *i* so that the new arc is projective. When creating a projective arc, the system will, in order to transform the non-projective graph as as little as possible, let *ANC(i)* be the nearest ancestor from its original head *i*. In figure

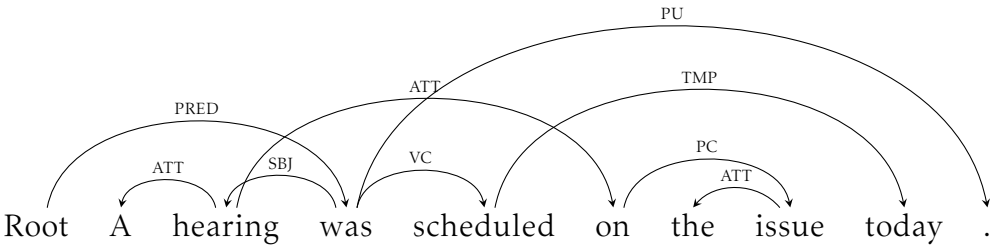

Figure 2.7: A non-projective dependency tree from Kübler et al. (2009).

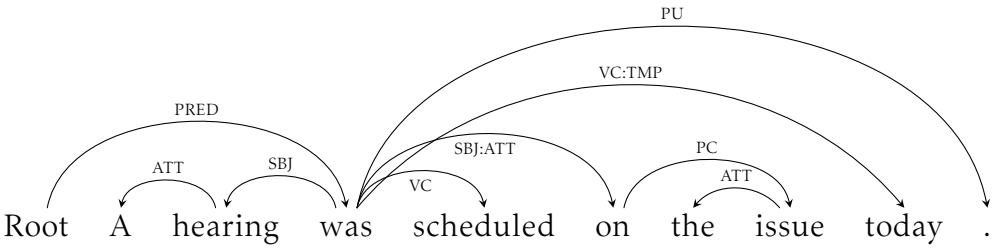

Figure 2.8: A projectivized dependency tree from Kübler et al. (2009).

2.7 from Kübler et al. (2009), we find that the relation *hearing* → *on* with the dependency type ATT and the relation *scheduled* → *today* with the dependency type TMP, are both non-projective. In figure 2.8 we can see the graph after it has been transformed, we see that some of the arc labels (dependency types) now contains additional information about its original head.

Since the projectivization modifies the data in the training set, the parser can be trained and tested as usual, using algorithms restricted to projective structures. For each new sentence being parsed, a projective dependency graph will be generated and the arcs that needs to be replaced in order to reconstruct the non-projective graph are labeled with special argument labels. The transition from a pseudo-projective graph back to a non-projective graph is done by replacing each arc of the form *(i, HEAD:DEP, j)* by an arc *(DESC(i), DEP, j)*, where *DESC(i)* is the descendant of *i* with an ingoing arc labeled HEAD. The algorithm is a breadth-first going from left-to-right. This is a simple algorithm, but it is able

to correctly recover more than 90 % of all non-projective dependencies found in the tested languages (Kübler et al. 2009).

2.3.6 MaltOptimizer

MaltOptimizer is a system for MaltParser optimization. Ballesteros and Nivre (2014) explain that the idea behind the system is to provide a tool for automatic optimization for MaltParser and and other transition-based dependency parsers. The system uses training data to generate an optimal model for the held out data for the parser. The training data must consist of sentences annotated with dependency graphs in the CoNLL data format. To derive the most optimal settings for MaltParser the system trains MaltParser models with a variety of hyper-parameter settings, one at the time, and then evaluate them on a held-out data set. We used MaltOptimizer as a part of our initial experiments to give us a pinpoint to which features might be worth looking into. The optimization process consist of three phases:

1. Validation of the data, data analysis and initial optimization.
2. Parsing algorithm selection
3. Feature selection and LIBLINEAR optimization.

In phase 1 the training data is analyzed as a preparation for the next steps in the optimization process. In this phase the system will gather information regarding the size of the data set, how many non-projective graphs it contains, the number of root nodes covered by a dependency arc and the number of dependency labels used for the root nodes. It will identify the CPOSTAGs and POSTAGs and determine whether the FEATS and LEMMA columns are empty or not. The system will optimize some initial parameters and suggest the most optimal validation strategy based on the size of the data set.

In phase 2 the system will use the result of the analysis from phase 1 to test the parsing algorithms in the MaltParser system with the default feature models and parameters. When deciding on the optimal algorithm it tunes the specific options of each parsing algorithm.

Phase 3 is the final phase and now MaltOptimizer will optimize the feature model for the selected parsing algorithm. Then as a final step the hyper-parameter of the LIBLINEAR classifier is tuned.

Chapter 3

Baseline experiments

In this chapter we will first give a precise description of the experimental protocol and the evaluation metrics used for the experiments presented in this and the following chapter. Then we describe our initial experiments were we use the MaltParser system, as described in the previous chapter, with its default settings. We will also present the results from these experiments together with an error analysis. At the end of the chapter we present the experiments and the results we obtained from running MaltOptimizer with the NDT.

3.1 Experimental protocol

The experiments on the NDT was conducted on the final version released 2014-01-03. The treebank is divided into two partitions, one in bokmål and one in nynorsk. We divided each partition into sets for training, development, testing and final evaluation. The training set was, naturally, used for training the parser, the test set was used for testing during the development and experimentation with the parser. The development set was used with the training set to tune MaltOptimizer, and the final held-out test set was used for the final evaluation presented in the end of this thesis.

The data was divided using two different methods as a mean to observe the resilience of the parser. Each of the partitions was divided into ten non-overlapping parts and then seven of them were used for training, one was used for for development, one for testing and the last part was kept as a held-out final test set. One set was created by placing the first

sentence in the first part, the second in the second part, and so on up to sentence ten, after that we started over, placing sentence eleven in the first part and so on. We call this the round-robin data set, named after the approached being called round-robin. The other data set was created using the first 70 % of the sentences in each source from table 2.1, and then combining them to the training set. The development set is made out of the next 10 % of the sentences in each source. The test set was then created by combining the next 10 % of the sentences in each source, and the held-out test set consists of the remaining 10 % of the sentences. This set is referred to as the split data set in the thesis.

3.1.1 Evaluation

For evaluation we mainly used the standard evaluation metrics in dependency parsing, unlabeled and labeled attachment score. Labeled attachment score (LAS) is an evaluation metric giving us the percentage of tokens the system has assigned both the correct head and the correct dependency relation. The other evaluation metric, the unlabeled attachment score (UAS) is the percentage of tokens which are assigned the correct head by the system. Some of the results are however presented with scores for precision and recall. Precision is a measure of how many dependencies of a given type returned by the parser that are correct. We define this as:

$$\text{Precision} = \frac{\text{Number of correct dependencies of a given type returned by the parser}}{\text{Number of dependencies of a given type returned by the parser}}$$

Recall on the other hand is a measure of how many of correct dependencies of a given type the parser was able to extract from the text. Meaning, it is:

$$\text{Recall} = \frac{\text{Number of correct dependencies of a given type given by the parser}}{\text{Number of correct dependencies of a given type in the text}}$$

The scores were found using the CoNLL-X evaluation script, eval.pl and the MaltEval system. The eval.pl script was created by the organizers of the first CoNLL shared task (Nilsson and Nivre 2008), and evaluates the

output from our system with respect to a gold standard. From the evaluation we receive information about the errors according to their type and context[1]. MaltEval is a relatively new evaluation tool for dependency parsing, and is based on eval.pl. The system combines quantitative and qualitative evaluation of data in the CoNLL format, and facilitates the visualization of the dependency structures. Nilsson and Nivre (2008) stress that this system is more flexible that eval.pl since it is implemented with a range of features that are lacking in eval.pl, and can easily be modified by adding a variety of parameters.

3.2 Baseline experiments & error analysis

The first published results from using MaltParser with the NDT gave Solberg et al. (2014) a labeled attachment score of 84.57 % for bokmål and 83.59 % for nynorsk. The unlabeled attachment score from the same parser was 88.02 % for bokmål and 87.09 % for nynorsk.

	Bokmål		Nynorsk	
	LAS (%)	UAS (%)	LAS (%)	UAS (%)
Malt default	84.57	88.02	83.59	87.09
Malt optimized	89.61	91.96	89.41	91.53
MST	88.37	91.97	87.64	91.23
Bohnet-2010	90.41	92.84	89.54	92.12
Bohnet&Nivre-2012	87.74	90.68	85.90	89.85

Table 3.1: Results from training and testing different parsers on the NDT by Solberg et al. (2014)

Solberg et al. (2014) also used the treebank on other parsers as we can see from table 3.1. The parser in their experiment which obtained the best results was the Bohnet parser (Solberg et al. 2014)[2].

In this section we will present results from our initial experiments and an error analysis, which will be used as a baseline for further experiments. The presentation of our results from these experiments and the

[1] http://ilk.uvt.nl/conll/software.html#eval

[2] The Bohnet&Nivre system is a system for joint part-of-speech tagging and labeled non-projective dependency parsing. Therefore, the results from the Bohnet&Nivre system are not comparable with the results from the other systems.

error analysis will be presented for both of the data sets, in bokmål and nynorsk.

Data set	LAS (%)	UAS (%)
Bokmål (Split)	83.15	86.76
Bokmål (Round-robin)	85.14	88.47
Nynorsk (Split)	83.56	87.60
Nynorsk (Round-robin)	83.82	87.44

Table 3.2: Results from our initial experiments where we trained and tested the NDT using MaltParser's default settings

In table 3.2 we find the results from our initial experiments where we trained and tested the NDT with MaltParser's default settings. When running with default settings MaltParser use Nivre's arc-eager parsing algorithm and the Liblinear learner. The root label is set to ROOT and no pseudo-projective transformation is performed. From table 3.2 we find that the results are, not surprisingly, quite similar to the results the presented by Solberg et al. (2014) in table 3.1.

Our results for the labeled attachment score on the split data set for bokmål is 1.42 % lower than what Solberg et al. (2014) obtained, while the labeled attachment score of the round-robin data set is 0.57 % higher than the results for bokmål in table 3.1. The labeled attachment score on the split data set for nynorsk is basically the same as in Solberg et al. (2014), only 0.03 % lower. The difference between the scores of the round-robin data set in nynorsk and the results for nynorsk in table 3.1 is also minor, with the score for the round-robin data set being only 0.23 % higher.

The unlabeled attachment score from our experiments did not bring any surprising differences from the scores obtained by Solberg et al. (2014). The unlabeled attachment score on the split data set for bokmål is 1.26 % lower than Solberg et al. (2014) while the unlabeled attachment score of the round-robin data set is 0.45 % higher than the results for bokmål in the same table. For nynorsk we found that the scores from our experiments were slightly higher, but the differences are minor. The score for the split data set is 0.51 % higher than in Solberg et al. (2014), while the scores from the round-robin data set is only 0.35 % higher.

These variations might be explained by the differences in the partitioning of the data sets, Solberg et al. (2014) divided the treebank (both bokmål and nynorsk) into 80-10-10 train, development and test sets. From the data we find that MaltParser gets a slightly lower score when parsing the round-robin data set, compared to the results we get from training and testing the parser on the split data set. Differences between the two variations of written Norwegian can partly be explained by the difference in the size of the data sets, the one in bokmål being slightly larger than the one in nynorsk. The versions of Norwegian are quite similar when it comes to syntax, but small variations might contribute to the difference in the results.

Distance	Recall (%)	Precision (%)
to root	95.86	71.71
1	96.19	94.50
2	87.34	90.84
3-6	75.47	86.56
7 -...	67.42	75.48

Table 3.3: Average precision and recall of binned head distance in the initial experiments

Table 3.3 shows the average precision and recall of binned head distance in the initial experiments. That is, how far away from the head is the dependent. From table 3.3 we find that when using MaltParser's default settings, both recall and precision is best when distance to head is 1, then it decreases when the distance to the head increases. We also see that the lowest precision with the default settings is the results for head distance being to root. This is a result of using MaltParser's default settings for root label, parsing algorithm and the marking strategy. With the default settings the root label is set to ROOT, which is not the root tag used in the NDT. This causes MaltParser apply the wrong root node, and in addition to this, all unattached nodes which will be attached to the (wrong) root node at the end of parsing each sentence.

From McDonald and Nivre (2007) we learn that a problem with MaltParser, due to the greedy parsing strategy, is that it is more likely to have problems with error propagation. A common problem with parsing systems is that their accuracy tends to be lower for longer sentences. In an analysis by McDonald and Nivre (2007) on errors when parsing with

Data set	Nr. of tokens	Average sentence length
Bokmål (Split)	21	15.70
Bokmål (Round-robin)	58	15.45
Nynorsk (Split)	68	17.33
Nynorsk (Round-robin)	68	17.26

Table 3.4: The number of tokens in the sentences with the highest number of errors (both word, head and dependency errors), when training and testing MaltParser on the NDT. Followed by the average sentence length (number of tokens / sentences) in each data set of The NDT

MSTParser and MaltParser, we see that MaltParser tends to have a higher accuracy on short sentences. This can be explained by the greedy parsing algorithm having to make fewer decisions, giving it a lower chance of error propagation. This can also explain why MaltParser tends to perform better for shorter dependency arcs, since shorter arcs are created before the long arcs, the chances of making an error is smaller. MaltParser usually constructs the arcs further away from the root early, and therefore the precision usually increases when the arc's distance to the root increases.

From the initial experiments we found that the sentences with the highest number of errors were particularly long. Table 3.4 gives us information regarding the average number of tokens in a sentence in the NDT. The average sentence length is between 15-17 tokens. Where the split set has a shorter average sentence length of 15.6 tokens, and the round-robin data set seems to contain slightly longer sentences, with an average sentence length of 17,29 tokens. In the examples of the sentences with the highest number of errors we find that the sentences in the split data set are shorter than the ones in the round-robin data set, particularly for the split data set in bokmål. In the round-robin data set the average sentence length of those sentences with the highest amount of errors contained 68 tokens, while the same measure for the split data set was only 58 tokens. Some of the errors with longer sentences might be due to the sentence length, but there are certainly other features that cause errors as well.

Table 3.5 and table 3.6 contains information regarding the ten most frequent errors done by the system when applying a dependency type to a relation. The tables describe the frequency of each error and what the

Split			Round-robin		
Frequency	Gold	System	Frequency	Gold	System
368	ADV	ATR	348	ADV	ATR
255	IP	ROOT	260	ATR	ADV
255	ADV	ROOT	253	ADV	ROOT
241	ATR	ADV	201	IP	ROOT
155	FINV	ROOT	119	FINV	ROOT
76	SUBJ	ROOT	83	SUBJ	FSUBJ
71	ADV	PUTFYLL	69	ADV	DOBJ
69	SBU	ROOT	61	SBU	ROOT
65	ADV	DOBJ	58	PUTFYLL	DET
61	SPRED	ADV	58	SPRED	ADV

Table 3.5: Bokmål: the ten most frequent error types in the initial experiments.

Split			Round-robin		
Frequency	Gold	System	Frequency	Gold	System
370	ADV	ATR	378	ADV	ATR
235	ADV	ROOT	286	ADV	ROOT
228	ATR	ADV	236	ATR	ADV
197	IP	ROOT	213	IP	ROOT
133	FINV	ROOT	132	FINV	ROOT
84	SUBJ	FSUBJ	75	SBU	ROOT
75	ADV	PUTFYLL	74	ADV	PUTFYLL
63	SUBJ	ROOT	67	SUBJ	ROOT
61	ADV	SPRED	59	ADV	SPRED
59	SBU	ROOT	56	INFV	SPRED

Table 3.6: Nynorsk: the ten most frequent error types in the initial experiments.

dependency type is in the gold standard test set. With MaltParser's default settings, the root label is set to ROOT, while in the NDT annotation, the main root is labeled with either FINV (finite verb), INTERJ (interjection) or FRAG (fragment) (Kinn et al. 2013). The root label is mostly FINV (*finite verb*) due to the rule stating that if the sentence is a main clause the finite verb is the head of the sentence. From the results presented in table 3.5 and 3.6 we see that a lot of errors in the initial experiments were caused by these alternative root tag in the annotation of the NDT. Another cause of the errors is as mentioned that MaltParser attaches all nodes it fails to attach as modifiers to the root node.

McDonald and Nivre (2007) stress the importance of relating linguistic categories to system accuracy. Linguistic categories in their analyses are parts-of-speech and dependency types. For part-of-speech, McDonald and Nivre (2007) distinguish verbs, nouns, pronouns, adjectives, adverbs, adpositions and conjuncts, while for dependency types they distinguish between a root category, a subject category, an object category and several categories related to coordination. The length of a dependency from word w_i to word w_j is the same as $|i - j|$. Long dependencies usually represent the main verb in a sentence or the modifiers of the root. Modifiers of nouns such as determiners, adjectives and pronouns often represent short dependencies. In the study they find that MaltParser has a slightly higher accuracy for nouns and pronouns compared to the MSTParser, but a lower accuracy on other categories (McDonald and Nivre 2007). This is due to the greedy parsing procedure of MaltParser creating the shorter arcs before longer arcs, and as mentioned above are shorter dependencies less prone to error propagation.

This made us curious to see if there was any difference in accuracy regarding the linguistic features when applying MaltParser on the NDT. The finding are presented in table 3.7 which gives us the percentage of errors for each linguistic category in each data set. From table 3.7 we find that the results from McDonald and Nivre (2007)'s study are valid for the accuracy of the linguistic categories in Norwegian too. We find that adjectives (adj), pronouns (pron) and determiners (det) have a lower error rate than prepositions (prep) or adverbs (adv). This can be linked to the results mentioned above due to the fact that nouns (subst) and pronouns (pron) are usually attached to verbs, which makes them occur lower in the graph, with shorter distance. MaltParser tends to have a lower precision for arcs that are predicted with more siblings, which can explain why it has a lower accuracy on adverbs, which tend to have a

	Bokmål		Nynorsk	
Part-of-speech	Split (%)	Round-robin (%)	Split (%)	Round-robin (%)
subst	11	9	9	8
verb	11	9	10	11
prep	17	16	17	16
adj	7	6	8	8
det	3	3	4	3
pron	4	3	4	5
adv	13	13	11	14
konj	4	4	5	6
sbu	8	6	7	8
inf-merke	8	7	7	8
ukjent	13	2	1	2
clb	10	6	14	13
interj	12	25	3	3

Table 3.7: Errors and their distribution over POSTAGs

high number of siblings.

We wanted to focus a bit on the errors concerning dependency labels, and we see from table 3.5 and table 3.6 that the parser frequently confuse the ADV function and the ATR function. As mentioned in chapter 2, all adverbials are given the dependency relation ADV, and so are other dependents modifying the verb, (prepositions, adverbs, subordinate clauses and adjectives). ADV is also used on construction which modifies adjectives, determiners, adverbs and prepositional phrases (Kinn et al. 2013).

Nouns will usually not take ADV dependents, but there are exceptions, such as when a verb is omitted, for example in elliptical constructions. The tag used for descriptive dependents of a noun is the ATR function. Adjectives being dependent of a noun will be analyzed as ATR, but ATR is also used on prepositions when it is dependent on an elided noun. In addition, the ATR function will be applied when relative clauses are dependent on nouns or pronouns and for particles following a noun. The PP-attachment problem, one of the most frequent ambiguities in the processing of natural languages, occur when a system is trying to make the correct attachment of prepositional phrases. This is one of the big issues with computer systems for Natural Language Processing, and we find, not surprisingly, that the task of determining whether a preposi-

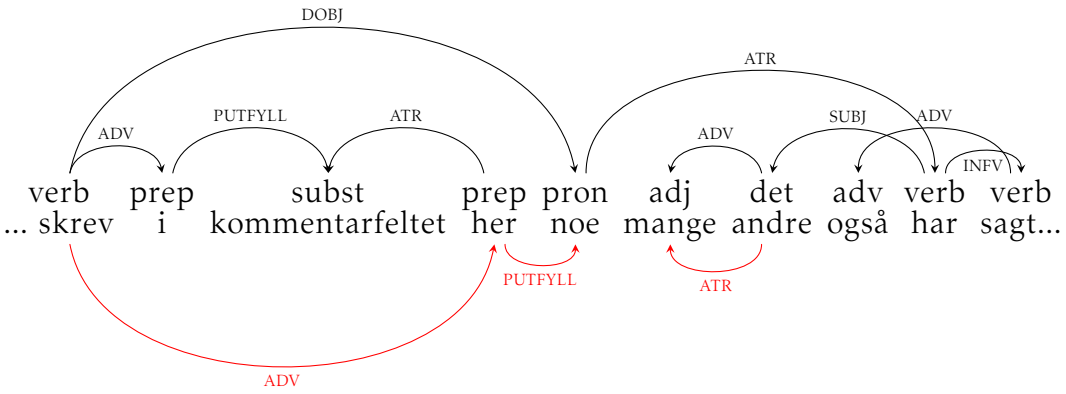

Figure 3.1: An example of the confusion of attachment and dependency relations.

tion should be ATR on a noun or ADV on a verb (Kinn et al. 2013) is a problem with the NDT. There is also a confusion regarding PUTFYLL (*preposisjonsutfylling*, prepositional complement) and ADV. PUTFYLL is used for the relation between a transitive preposition and its complement when the complement is a nominal, a clause, an adverb or another preposition. This is another example where there is ambiguity surrounding the prepositional attachment.

An example of the issue is illustrated in figure 3.1 which is a long sentence (58 tokens!) from the test set in bokmål where the errors mentioned above occur, together with other errors. The figure represent just parts of the sentence, due to the length of the sentence, the entire graph would be to big to fit on a page[3]. The black arcs above the sentence in figure 3.1 represents the gold standard analysis of the text, while the red arcs below illustrates the errors made by MaltParser when parsing the sentence. The ADV/ATR confusion is illustrated with the arc between the word *mange* (adj) and *andre* (det), where the adjective is the dependent of

[3]This is the entire sentence in Norwegian: *"Jeg fikk noen fine tilbakemeldinger på det, blant annet lurte Mammadamen på om jeg ønsket meg en bloggplakat og Ylvalia skrev i kommentarfeltet her noe mange andre også har sagt andre steder; enhver får ta ansvar for seg selv, hvis man får dårlig selvbilde av å lese om andres lykke er det ens eget problem."*. Which translates to something like this: "I got some great feedback on it, including Mammadamen wondered if I wanted a blog poster and Ylvalia wrote in the comment field here something many others have said elsewhere; everyone must take responsibility for themselves, if you get bad self-esteem by reading about other people's happiness, it is your own problem.".

the determiner, with the function ADV, since it is a construction which modifies the determiner. MaltParser on the other hand analyzed this relation as an ATR function. The other two arcs showing errors made by MaltParser shows how it not only applies the wrong label to relations, but also makes errors regarding the head and dependent relation. The word *skrev* (verb) is the head of the word *noe* (pron), with the relation DOBJ (direct object), but MaltParser did an entirely different analysis. In MaltParser's analysis *skrev* is the head of the preposition *her*, and the relation is given the label ADV, and the word *noe* is the dependent of *her*.

Norwegian is, like most Scandinavian languages, a V2-language meaning, the finite verb is the second constituent in a declarative main clause. The canonical word order in Norwegian is SVO, but when topicalization occurs, the word order changes to xVSO. The sentence-initial position can be occupied by pretty much any constituent, but nominals are the most common. The annotation of the NDT distinguishes between three different types of subjects, and also between referential subjects and non-referential subjects. Referential subjects are subjects in the subject position and are given the tag SUBJ. Non-referential subjects in subject position will be given other function such as FSUBJ or PSUBJ. The subject is defined using word order criteria: In main clauses the subject will be positioned right before or after the finite verb. In subordinate clauses on the other hand, the subject will appear before the finite verb[4].

Formal subjects (FSUBJ) have no clear reference and are described as semantically empty elements by Kinn et al. (2013). Their mission is to fulfill the requirement that finite sentences are required to have an overt subject. Some of the errors made by the system when applying the FSUBJ relation rather than the SUBJ relation can be caused by the fact that a lot of sentences starts with the word *det (that* or *it)*, which is a formal subject, while it is positioned like a subject.

Another cause for confusion regarding the subject (SUBJ) are the potential subjects. Potential subjects (PSUBJ) occur together with formal subject in presentational sentences. Unlike the other subjects mentioned above, a potential subject does not occur in the usual subject position. It will appear in the position where we usually find the object, while the formal subject is found in the subject position. Kinn et al. (2013)

[4]Not necessarily immediately before — the subject and the finite verb can be separated by an adverbial

mention that it might be difficult to distinguish a potential subject from adverbials and subject predicates. The errors we see MaltParser making regarding subjects might be caused by the different forms of subject and the way they are positioned.

3.3 MaltOptimizer

The initial experiments and the error analysis gave us an indication of the most frequent errors which occur when training and testing the NDT with MaltParser. The results from the experiments and the errors we found in the analysis made us curious to what the automated system for MaltParser optimization, MaltOptimizer, could improve and what choices it would make regarding the settings and the use of linguistic features. We were also curious too see which issues would be solved by MaltOptimizer and which would prove harder to solve. The data presented in this section was found testing and tuning each data set with the MaltOptimizer system described in chapter 2. After the training and tuning, we trained and tested MaltParser with each data set just as in the baseline experiments, but we modified MaltParser with the settings suggested by MaltOptimizer. We will present the changes MaltOptimizer suggested to the settings and some of the most interesting findings from training and tuning with MaltOptimizer.

	Split		Round-robin	
	LAS (%)	UAS (%)	LAS (%)	UAS (%)
Baseline	83.15	86.76	85.14	88.47
Malt Optimized[a]	89.61	91.96	89.61	91.96
Malt Optimized	88.69	91.14	89.89	92.18

[a] Solberg et al. (2014)

Table 3.8: Bokmål: results from our initial experiments where we trained and tuned the NDT using MaltOptimizer before parsing it with MaltParser supplemented with new settings

Tables 3.8 and 3.9 show us the results from the training and testing of MaltParser with the NDT and the optimized settings compared to the results from the baseline experiments. It also contains the results from the experiments Solberg et al. (2014) did with MaltOptimizer. We find that the optimized settings suggested by MaltOptimizer increased both the labeled attachment score and unlabeled attachment score for both bok-

	Split		Round-robin	
	LAS (%)	UAS (%)	LAS (%)	UAS (%)
Baseline	83.56	87.60	83.82	87.44
Malt Optimized[a]	89.41	91.53	89.41	91.53
Malt Optimized	88.93	91.58	89.45	91.80

[a] Solberg et al. (2014)

Table 3.9: Nynorsk: results from our initial experiments where we trained and tuned the NDT using MaltOptimizer before parsing it with MaltParser supplemented with new settings

mål and nynorsk compared to the results from the baseline experiments. When we compare our scores to those in the baseline experiments we see an increase in the labeled attachment score by 5.54 % for the split data set in bokmål and a 4.75 % increase for the round-robin data set in bokmål. For nynorsk we find a 5.37 % increase of the labeled attachment score for the split data set, and an increase of 5.63 % with the round-robin data set. The unlabeled attachment score increased by 4.38 % for the split data set and 3.71 % for the round-robin data set in bokmål. When we compare our scores with the scores from Solberg et al. (2014) we see that there are minor differences between the results from the round-robin data set and Solberg et al. (2014), while the split data set obtains slightly lower scores.

	MaltOptimizer		Baseline	
Distance	Recall (%)	Precision (%)	Recall (%)	Precision (%)
to root	96.71	95.86	95.86	71.71
1	97.06	95.89	96.19	94.50
2	93.02	92.75	87.34	90.84
3-6	85.03	88.96	75.47	86.56
7 -...	79.20	80.17	67.42	75.48

Table 3.10: Average precision and recall of binned head distance in the experiments with MaltOptimizer.

As in the baseline experiments, we wanted to see if the optimization process had an effect on the precision and recall of binned head distance. Table 3.10 shows the average precision and recall from the experiments with MaltOptimizer compared to the results from the baseline experiments. In the table we find that both precision and recall still decreases

as we get further away from the root, however both recall and precision is higher than the scores from the baseline experiments. The recall for distance being "to root" is only slightly higher, 0.85 %, than the baseline results. Precision for the same distance is on the other hand much more noteworthy with an increase in of 24.15 % from the baseline experiments.

As in the baseline experiments, the scores are highest when the distance is one, giving us a recall of 97.06 % in the experiments with MaltOptimizer, while the precision is at 95.89 %. The recall drops from 97.06 % to 93.02 % and the precision decreases from 95.89 % to 92.75 % when the distance is two. This is a drop in recall of 4.04 % and a drop of 3.11 % in precision. However, this drop in recall and precision when the distance increased from one to two is much lower than in the baseline experiments. When the distance is between three and six the recall decreases to 85.03 % and the precision decreases to 88.96 %. When the distance is seven or higher, the recall decreases down to 79.20 %, which is 11.78 % higher than the recall on the same distance in the baseline experiments. The precision also decreases when the distance becomes this high and our analysis show an average precision of 80.17 %, which is 4.69 % higher than in the baseline experiments.

The increase in precision in the "to root" results might be explained by the choice of root label, parsing algorithm, the marking strategy and feature models. These differ a lot from the settings in the baseline experiments. In the experiments with MaltOptimizer the root label was set to FINV rather than ROOT, as it was in the baseline experiments. The default parsing algorithm in MaltParser is Nivre's arc-eager parsing algorithm and was used in the baseline experiments. MaltOptimizer suggested we should continue to use Nivre's arc-eager algorithm for the split data set. For the round-robin set it opted for the projective Stack algorithm. MaltOptimizer suggest we use the Stack Projective parsing algorithm and Nivre's arc-eager parsing algorithm, which are both restricted to projective dependency structures and will therefore require the use of the projectivize feature in MaltParser to create pseudo-projective structures.

This due to the analysis made by MaltOptimizer on the training data revealing that the data set in bokmål consist of approximately 18.5 % non-projective trees, while the data set in nynorsk contains approximately 20.5 % non-projective trees. The exact numbers are presented in ta-

Data set	Non-projective trees
Bokmål (Split)	18.8065
Bokmål (Round-robin)	18.2776
Nynorsk (Split)	20.3641
Nynorsk (Round-robin)	20.8901

Table 3.11: The percentage of non-projective trees in each data set of The NDT

ble 3.11. MaltOptimizer therefore set the marking strategy for pseudo-projective transformation from being *none* in the baseline experiments, to *head*.

MaltOptimizer also suggested other feature models than MaltParser's default model[5]. In the experiments presented later in this thesis we therefore use the feature models which MaltOptimizer suggested. For the split data sets we use the same model for both bokmål and nynorsk, while for the round-robin data sets we use two other feature models, one model for the set in bokmål and another one for the set in nynorsk. The difference in the settings indicates that even though bokmål and nynorsk are syntactically similar, the two variation might need different settings to achieve optimal parsing results.

We also wanted to look at what the changes in the settings of MaltParser would result in when it came to type confusion, just as we looked at in the baseline experiments. The results are presented in table 3.12 and 3.13 which presents the wrong type applied by MaltParser, the correct type taken from the gold standard test set and the frequency of the given confusion in each data set. From the results presented in table 3.12 and 3.13 we find that the errors caused by the alternative root tag as seen in initial experiments are reduced due to giving the right root label as a parameter when training and parsing with MaltParser. We also see the most frequent error is still the distinction between the ATR function and the ADV function. In the split data set for bokmål MaltParser applies the ATR function 310 times when is was supposed to be the ADV function. And the system labels a relation as an ADV relation when the gold standard analysis is an ATR relation 277 times for the same set.

[5]The feature models suggested by MaltOptimizer came with the MaltOptimizer system.

Split			Round-robin		
Frequency	Gold	System	Frequency	Gold	System
310	ADV	ATR	322	ADV	ATR
277	ATR	ADV	236	ATR	ADV
56	ADV	SPRED	75	SUBJ	FSUBJ
53	SPRED	ADV	66	ADV	DOBJ
51	SUBJ	FSUBJ	59	ADV	PUTFYLL
51	ADV	PUTFYLL	48	ADV	SPRED
45	ADV	DOBJ	43	PUTFYLL	ADV
43	FRAG	FINV	40	SPRED	ADV
35	ATR	FLAT	33	PSUBJ	SPRED
33	SUBJ	DOBJ	31	SUBJ	DOBJ

Table 3.12: Bokmål: the ten most frequent error types in the experiments with MaltOptimizer.

Split			Round-robin		
Frequency	Gold	System	Frequency	Gold	System
322	ADV	ATR	355	ADV	ATR
221	ATR	ADV	245	ATR	ADV
73	SUBJ	FSUBJ	73	ADV	PUTFYLL
47	ADV	SPRED	60	PUTFYLL	ADV
46	SPRED	ADV	54	SUBJ	FSUBJ
45	ADV	PUTFYLL	53	ADV	SPRED
41	ADV	DOBJ	43	SUBJ	DOBJ
41	DOBJ	ADV	37	DOBJ	PUTFYLL
36	PSUBJ	SPRED	36	ADV	DOBJ
33	SUBJ	DOBJ	32	SPRED	ADV

Table 3.13: Nynorsk: the ten most frequent error types in the experiments with MaltOptimizer.

The system also confuses the ADV and PUTFYLL function. On average the system analyze a relation as a PUTFYLL relation when it is an ADV function 57 times, which is much lower than the ADV/ATR confusion. The confusion between ADV and PUTFYLL can partly be explained by the prepositional attachment problem mentioned in the previous section. In general we see that other analysis related to ADV is troublesome, 27 of the 40 errors presented in the tables are related to confusion between the ADV function and other types. Another common error is the *subject* (SUBJ) being analyzed as a *formal subject* (FSUBJ) as we saw in the error analysis. We also see that there is a confusion between the subject (SUBJ) and direct object (DOBJ) relation. This is again most likely from the property of Scandinavian where the object can hold the sentence-initial position as a result of topicalization. This will result in a change in the word order, shifting from the canonical SVO to OVS.

Chapter 4

Linguistic features

In this chapter we present the experiments conducted with focus on the effect of linguistic features when parsing Norwegian. We will briefly introduce some of the research carried out this field which has given us inspiration and cues to the design of each experiment. We will also describe how the experiments were conducted, what kind of results we obtained and briefly discuss these findings. We start by presenting the experiments focusing on extended part-of-speech tags and dependency relations, followed by the experiments conducted with focus on morphological features. After that we will present the experiments where we made use of other linguistic features or combinations of features. Finally, we present how well the system perform when we apply a final held-out test set and an how well it adapts to an unfamiliar domain.

The experiments carried out in this chapter are constructed with the purpose of investigating the effect of linguistically motivated features on parsing accuracy for Norwegian. We used the settings and the feature models suggested by MaltOptimizer as a base for our experiments. Apart from the feature models and the parsing algorithm, all the experiments in this chapter use the same settings. The split data set used a different parsing algorithm than the round-robin data set, however, both in combination with pseudo-projective transformation. The marking strategy for the pseudo-projective transformation was at all times set to head, meaning that the parser projectivized the input data with head encoding for labels. The root label was set to FINV (finite verb), while the attachment strategy for covered roots was set to none, making the parser treat covered roots as any other node. The only parameter we changed in the experiments were the feature models, and in those we mostly made

variations regarding the features involving the FEATS column.

For the split data sets we used the same feature model for both the data set in bokmål and the one in nynorsk. The parsing algorithm used was Nivre's arc eager parsing algorithm, and the feature model we used as our base can be found in the MaltOptimizer-1.0.3 distribution. The feature model allow MaltParser to gather information and learn from the features of the first and the second token in the input list, and the features of the token on top of the stack.

The round-robin data sets used the Stack projective parsing algorithm. The data set in bokmål used one model as a base, while the feature model used for the data set in nynorsk is based on another model. The difference between these two feature models is minor. The main difference is an additional feature where the POSTAG of the top two tokens on stack and the the FORM of the third token on stack are merged into a feature. In both models MaltParser extracts information about the features of the first and the second token on stack, and the features of the two first tokens from the start of the lookahead buffer.

4.1 Extended part-of-speech tags

In this section we describe the process and the results of our experiments conducted with the intention of understanding the use of part-of-speech tags and dependency relations as features in dependency parsing. We will first introduce some concepts and thoughts regarding the theme before we present the experiments and the results.

The tagset in the NDT is quite similar to the universal tagset constructed by Petrov et al. (2012). They propose a universal tagset consisting of twelve part-of-speech categories based on the idea that there is a set of syntactic part-of-speech categories in similar form across most languages. The part-of-speech tagset in the NDT also consists of twelve categories, a limited amount compared to other tagsets such as the one for the Penn Treebank. The part-of-speech tagset of the NDT are made up from the basic word classes: *adj* (adjective), *adv* (adverb), *det* (determiner), *inf-merke* (infinitive marker), *interj* (interjection), *konj* (conjunction), *prep* (preposition), *pron* (pronoun), *sbu* (subjunction), *subst* (noun), *ukjent* (unknown) and *verb* (verb).

Split		Round-robin	
Focus words	Frequency	Focus words	Frequency
i (in)/ prep	217	i (in)/ prep	212
på (on)/ prep	108	til (to)/ prep	95
til (to)/ prep	93	på (on)/ prep	87
for (for)/ prep	79	det (that)/ pron	78
som (as)/ prep	69	for (for)/ prep	77

Table 4.1: Bokmål: the frequency of the top five words were the most errors occur in the experiments with MaltOptimizer.

As we can see from this list, the part-of-speech tags in the NDT does not hold any information regarding inflection etc. In the NDT, a verb can only carry the part-of-speech tag *verb*, whereas a verb annotated by the Penn Treebank annotation will be given one of these tags: *VB* (base form), *VBD* (past tense), *VBG* (gerund or present participle), *VBN* (past participle), *VBP* (non-3rd person singular present) or *VBZ* (3rd person singular present). This made us wonder if a more detailed set of part-of-speech tags would reduce the need for including morphological features. We worried that the limited part-of-speech tagset in the NDT would be to coarse for dependency parsing, however Petrov et al. (2012) found in their experiments that the accuracy of an English dependency parser decreased only 0.6 % when they used the twelve universal POS tags instead the forty-five tags from PennTreebank which it used originally. Extracting and adding more information to a data set will also often have a negative impact on efficiency. A small, and preferably universal part-of-speech tagset would make it easier to evaluate parsers across languages, and would also make the task of cross-linguistic parsing easier.

Still, we wondered if a more detailed set of part-of-speech tags could increase the parsing accuracy while it decreased time spent on preprocessing, training and parsing. If so, this is a method that would require less resources than feature extraction of morphological features.

Experiments with part-of-speech tags and prepositions

The initial experiments shows that prepositions are difficult to get right for the parser, which is a common problem in NLP.

Split		Round-robin	
Focus words	Frequency	Focus words	Frequency
i (in)/ prep	219	i (in)/ prep	242
på (on)/ prep	87	til (to)/ prep	112
til (to)/ prep	84	på (on)/ prep	98
det (that)/ pron	77	det (that)/ pron	75
med (with)/ prep	60	for (for)/ prep	68

Table 4.2: Nynorsk: the frequency of the top five words were the most errors occur in the experiments with MaltOptimizer.

In table 4.1 and 4.2 showing the frequency top five words were the most errors occur, most of the words are prepositions. The word which the parser seems to struggle the most with is the preposition *i* (in), with an error frequency between 212 and 242. As mentioned earlier, getting a parser to make the correct attachment of prepositional phrases is one of the big issues with computer systems for NLP. There are no simple formal rules for the use of each particular prepositions in Norwegian and the choice of preposition is often determined by the context and not the grammatical properties. Therefore it can be difficult to determine whether a preposition should be ATR on a noun or ADV on a verb, and the high frequency of errors made in the distinction between ADV and ATR shown in chapter 3 might be linked to the lack of morphological information for prepositions.

Since prepositions are not inflected, and they carry no morphosyntactic tags to be used for parsing in the NDT. This made us curious to see if more fine-grained part-of-speech tags for preposition would reduce the problems MaltParser encountered with this class of words. To see if we could get an improvement on the accuracy of parsing prepositions we opted to see if we would get any promising result with part-of-speech tags and lexicalization.

In our first experiments with part-of-speech tags we merged the form with the part-of-speech tag for every preposition in the training data for each data set. In addition to this we did a very similar experiment where we merged the lemma, rather than form, of each preposition with the part-of-speech tags. After those experiments we decided we wanted to try merging the the dependency relations with the lemma and the

	Split		Round-robin	
	LAS (%)	UAS (%)	LAS (%)	UAS (%)
Baseline	83.15	86.76	85.14	88.47
POS + FORM	87.06	89.82	88.69	91.26
POS + LEMMA	87.10	89.85	88.73	91.28
DEPREL + FORM	86.99	89.81	88.47	91.09
DEPREL + LEMMA	87.01	89.81	88.51	91.09

Table 4.3: Bokmål: results from merging the part-of-speech tag or dependency relation with the lemma or form of the prepositions in the NDT.

form of the prepositions to see if we would get different results. Our first experiments with merging lemma or form with dependency relations caused a quite substantial decrease in the labeled attachment scores compared to the scores from the baseline experiments. The average labeled attachment score obtained by the experiments was at first 77.59 % for bokmål and 76.48 % for nynorsk. At the same time as the scores for the labeled attachment dropped, the unlabeled attachment score increased quite a lot. The baseline labeled attachment score was in the range 83.15 - 85.14 %, while merging dependency label and lemma gave an unlabeled attachment score between 91.34 - 91.93 %. This was probably caused by us creating a larger output set, due to the dependency relation being a category in the output format of MaltParser. When we changed the dependency relation tags in the synthetic data set back to the original dependency tags we got the results presented below.

The reformatting of the data was done using a system we wrote that checks the part-of-speech tag for each token in the data set given as input. For each preposition (prep) it found, it merged the part-of-speech tag or dependency label with the form or lemma of the token and wrote this to a new output file. All the lines where the token was not altered, due to not being a preposition, were simply written to the output file. We made additions to the feature models used for these experiments, however, in order to evaluate the impact of this change in isolation from any morphological features, MaltParser did not make any use of the morphological information in the FEATS column.

The results from training and testing with the more fine grained part-of-

	Split		Round-robin	
	LAS (%)	UAS (%)	LAS (%)	UAS (%)
Baseline	83.56	87.60	83.82	87.44
POS + FORM	87.59	90.54	87.98	90.78
POS + LEMMA	87.59	90.54	88.01	90.78
DEPREL + FORM	87.30	90.45	87.81	90.62
DEPREL + LEMMA	87.36	90.48	87.78	90.60

Table 4.4: Nynorsk: results from merging the part-of-speech tag or dependency relation with the lemma or form of the prepositions in the NDT.

speech tags or dependency relations are presented in table 4.3 and 4.4 together with the results from the baseline experiments. The unlabeled attachment score increased with 2.81 - 3.34 %, while the labeled attachment score increased with 3.59 - 4.19 % . These results are a bit lower then the the ones obtained using MaltOptimizer, but it is worth noting that an increase of 4.19 % is quite a lot considering that we made no use of morphological features. We also found that there does not seem to be any difference regarding the use of lemma or form. This is due to preposition not being inflected. What surprised us a bit was that there was no substantial change in the numbers of errors made regarding the five focus words where most of the errors occur seen in table 4.1 and 4.2. From table 4.3 and 4.4 we find that the merging of the form or lemma with the dependency relation resulted in a slightly lower accuracy than the experiments were lemma or form were merged with the part-of-speech tag.

Experiments with part-of-speech and morphological features

The previous experiments made us curious to what effect the different morphological features combined with a part-of-speech tag could have on parsing accuracy. If this proved successful it might be a solution which is more cost efficient, both in training and parsing time due to the simple feature model, and that less work would have to be used on feature engineering. We mostly used the same procedure as in the experiments above. However, this time we merged the part-of-speech tag with the morphological tags rather than the form or lemma of the token. For each experiment in this section we merged the morphological tag, e.g. for case the tag would be *nom*, *akk* or *gen*, with the part-of-speech tag. If no

	Split		Round-robin	
	LAS (%)	UAS (%)	LAS (%)	UAS (%)
Baseline	83.15	86.76	85.14	88.47
POS + case	86.92	89.57	88.56	91.03
POS + definiteness	88.34	90.95	89.29	91.71
POS + gender	87.51	90.31	89.12	91.56
POS + number	88.16	90.78	89.39	91.77
POS + person	86.88	89.52	88.48	90.99
POS + tense	86.85	89.58	88.79	91.26

Table 4.5: Bokmål: results from the experiments with morphological features merged with a part-of-speech tag.

	Split		Round-robin	
	LAS (%)	UAS (%)	LAS (%)	UAS (%)
Baseline	83.56	87.60	83.82	87.44
POS + case	87.94	90.76	87.87	90.55
POS + definiteness	88.26	91.24	88.54	91.18
POS + gender	87.72	90.67	87.78	90.58
POS + number	88.27	91.18	88.23	90.93
POS + person	87.57	90.59	87.62	90.35
POS + tense	87.74	90.75	87.67	90.28

Table 4.6: Nynorsk: results from the experiments with morphological features merged with a part-of-speech tag.

tag were found for the given token, no merging would be performed and therefore we would not alter the part-of-speech tag. The feature models used in these experiments are the same models as those applied on the experiments in the section above.

The results from the experiments can bee seen in table 4.5 for bokmål and in table 4.6 for nynorsk. Both the labeled attachment score and the unlabeled attachment score are better than baseline, and in some cases slightly higher than the scores from the experiments with part-of-speech tags merged with the lemma or form of prepositions. What we found interesting is that some morphological features seem to have a larger impact on parsing accuracy than others. Those features that might prove more important when parsing Norwegian are *definiteness*, *number* and

Part-of-speech	Morphosyntactic tags
Adjective	*gender, number, type, definiteness and degree*
Determinative	*gender, number, type and definiteness*
Conjunction	*type*
Pronoun	*gender, number, type, person and degree*
Complementizer	*type*
Noun	*gender, number, type, definiteness and case*
Verb	*tense*

Table 4.7: Morphosyntactic tags from the Oslo-Bergen Tagger. For more information see appendix A

gender for bokmål. *Definiteness*, *number* and *case* for nynorsk. The difference in the features effect on parsing might be due to the amount of information carried by these tags. The different morphological types contains different amount of information and some features, and therefore the tags, are more frequent than other. For more information about this see appendix A where information regarding the morphological tagset from the Oslo-Bergen tagger is presented.

4.2 Morphological features

The experiments presented in this section was designed with the intention of gaining a better understanding of the effect of morphological features on parsing accuracy. We based the following experiments on our findings in the previous section and on research on morphological features in field of dependency parsing for Scandinavian languages.

The Norwegian Dependency Treebank implements some morphological information as you can see in table 4.7. In order to better understand the effect of each individual feature presented in table 4.7 on parser accuracy we constructed the experiments by modifying the corpus and the feature models suggested by MaltOptimizer so that MaltParser would only look at a single morphological feature at the time. Further in this section we will give a brief explanations to the different morphological features before we present the results from our experiments. We will also explain how we proceeded with further experimentation with the morphological features based on our finding and information regarding what others have accomplished using morphological features in dependency parsing.

4.2.1 Morphological features

Gender Bender (2013) writes *"in all languages with grammatical gender, all nouns are assigned to a particular noun class, and this class is reflected in morphological properties of other elements in the sentence that agree with nouns"* Gender is in many languages, such as in Norwegian, a grammatical property associated with nouns. In Norwegian, all nouns have a correspondence to a grammatical gender. Words related to a noun agree with the gender of the noun, i.g. *fin bygning* (nice building), *fint hus* (nice house). Norwegian has a three-class gender system, where the classes are masculine (hankjønn in Norwegian), feminine (hunkjønn/hokjønn in Norwegian bokmål/nynorsk) and neuter (intetkjønn/inkjekjønn in Norwegian bokmål/nynorsk) (Faarlund 1997).

Number Bender (2013) explains that the property of number is "related to the cardinality of the set picked out by the referent". A singular noun phrase refers to a single individual, e.g. *Here is a small car*, while a plural noun phrase refers to a set with zero or more than one individual. *Number* is either *singular* (entall in Norwegian, ent. in the NDT) and *plural* (flertall in Norwegian, fl. in the NDT). Words in singular are equal to their lemma while the plural form is usually obtained by adding a suffix to the singular form of the word (Faarlund 1997). As we can see from table 4.7, determiners, adjectives, pronouns and nouns carry morphosyntactic information regarding number.

Definiteness In the Norwegian Dependency Treebank nouns, adjectives and determiners are inflected or declined in definiteness (*ubestemt* indefinite and *bestemt* definite). *Definiteness* in Norwegian can be expressed in two ways. One is through a definite article, or through a suffix expressing definiteness, which changes in gender and number in accordance with the word it is attached to. One example taken from the Faarlund (1997) is *båt* + *en* (masculine, singular, definite) (the boat). Definiteness can also be expressed through inflection of adjectives.

Tense *Tense* is commonly used as an indicator of temporal relationships and gives us information about grammatical location in time (Bender 2013). *Tense* is grammatical property frequently used to mark temporal information via verbs. In the Norwegian Dependency Treebank the only class marked with morphological tense is verb.

Person In the Norwegian Dependency Treebank person is a grammatical category only for pronouns. Person refers to participants in an event. We can say that first person is a reference referring to the acting agent, while second person refer to the patient. Third person is used to reference someone who is neither agent nor patient.

Case *Case* is in many languages an important grammatical property in the relationship between a noun and the sentence it appears in. In Norwegian the use of case is marginal. Nominative case is usually preferred when the pronoun is modified. In the Norwegian Dependency Treebank nouns distinguish only genitive case. Pronouns are marked for nominative or accusative case.

Degree In Norwegian only adjectives are declined in comparison (*positive*, *comparative or superlative*). Positive is expressed by the root form of the word (en *grønn* drage, a green dragon), while the two other are expressed by an additional suffix (*en grønnere drage*, *den grønneste dragen*, a greener dragon, the greenest dragon).

For our experiments we reformatted the treebank quite a bit. We used the information in the FEATS column and distributed those features over ten new columns that were added to the data format specification file. We created new columns for each morphosyntactic tag in the tagset of the Oslo-Bergen tagger; gender, number, type, tense, definiteness, person, case and degree. We also created a separate column for a hum tag and an additional column carrying information about whether a verb is finite or infinite, we will describe this in more details later in the thesis. A table of the morphosyntactic tags in the Oslo-Bergen tagset can be viewed in appendix A. In these experiments the feature models were altered so that MaltParser would look at a individual feature, rather than all the features at once such as in the base models. An example of this reformatting is in the feature model for the round-robin data set in nynorsk, were we altered the features regarding FEATS from looking at all the features for the top two tokens on stack and lookahead, to only look at e.g. number or gender for the tokens in the same positions. In the end we created additional feature models so that MaltParser would gather information from all the features at once.

From table 4.8 and table 4.9 and we can see the effect of the individual features when testing and training MaltParser on the reformatted data

	Split		Round-robin	
	LAS (%)	UAS (%)	LAS (%)	UAS (%)
Baseline	83.15	86.76	85.14	88.47
Gender	87.58	90.30	89.14	91.55
Number	87.89	90.58	89.12	91.56
Definiteness	87.85	90.50	89.17	91.63
Tense	87.03	89.64	88.78	91.19
Person	87.32	90.00	88.51	90.98
Case	86.89	89.63	88.54	91.01
Degree	86.97	89.70	88.60	91.05
All	88.30	90.81	89.76	92.05

Table 4.8: Bokmål: the effect of the individual features.

	Split		Round-robin	
	LAS (%)	UAS (%)	LAS (%)	UAS (%)
Baseline	83.56	87.60	83.82	87.44
Gender	87.99	90.92	87.85	90.55
Number	88.17	91.11	88.15	90.77
Definiteness	88.24	91.13	88.19	90.83
Tense	87.65	90.63	87.88	90.52
Person	87.40	90.19	87.54	90.25
Case	87.67	90.61	87.50	90.22
Degree	87.65	90.59	87.83	90.44
All	88.88	91.52	89.19	91.60

Table 4.9: Nynorsk: the effect of the individual features.

	Split		Round-robin	
	LAS (%)	UAS (%)	LAS (%)	UAS (%)
Baseline	83.15	86.76	85.14	88.47
All	88.30	90.81	89.76	92.05
Gender	88.13	90.78	89.47	91.80
Number	88.06	90.70	89.57	91.89
Definiteness	88.17	90.81	89.56	91.88
Tense	88.04	90.75	89.33	91.74
Person	88.10	90.70	89.58	91.91
Case	88.15	90.74	89.58	91.88
Degree	88.16	90.76	89.51	91.84

Table 4.10: Bokmål: the effect of removing one feature at the time on parsing accuracy

sets. As we can see the individual features affect the two written variations of Norwegian and the splits a bit different. From our results we find that the features with the most effect on parsing accuracy are the same as those we found in the experiments with part-of-speech tags. That is, the most prominent features over all are *gender*, *number* and *definiteness*. When we combine all of the features, the scores are quite close to those in the experiments with MaltOptimizer.

Further we wondered how parsing accuracy would be affected if we combined the features in the table above. We therefore decided to do an ablation study where we removed one feature at the time to see how much this feature influenced the parsing accuracy. We used the feature model used to obtain the results for *All* in table 4.8 and table 4.9. We then conducted a series of experiments were one feature was removed while we kept all the others, and then we trained and tested MaltParser with this model. After the training and testing with one feature missing, we would put the feature back in the model and remove another feature, and then repeat the training and testing. E.g., when we removed the columns that used information regarding gender from the model, but kept the columns for all the other features, then we tested and trained MaltParser on this model. Afterward the information regarding gender would be put back into the model, and information regarding another features would be removed.

	Split		Round-robin	
	LAS (%)	UAS (%)	LAS (%)	UAS (%)
Baseline	83.56	87.60	83.82	87.44
All	88.88	91.52	89.19	91.60
Gender	88.64	91.38	88.89	91.28
Number	88.54	91.30	88.85	91.39
Definiteness	88.57	91.34	88.80	91.31
Tense	88.63	91.35	88.59	91.10
Person	88.72	91.40	88.92	91.37
Case	88.67	91.38	88.92	91.41
Degree	88.56	91.25	88.61	91.21

Table 4.11: Nynorsk: the effect of removing one feature at the time on parsing accuracy

The results are presented in table 4.10 and table 4.11 and indicate that removing only one feature does not impact the parsing accuracy much. The scores when removing one feature were all a bit lower then the scores obtained by combining all the features, but there were no feature that seemed to have such big impact on parsing accuracy that it would cause a big drop in accuracy if removed.

4.3 Linguistic features

In this section we will present our further experimentation with linguistic features. They are based on the information gained from the experiments in the previous section, the available information in the NDT and again, inspired by the work on dependency parsing done by others.

Swedish is a language relatively similar to Norwegian and in the field of dependency parsing, a lot more research has been done on the Swedish Talbanken05[1]. Øvrelid and Nivre (2007) use additional linguistically motivated features to target specific error types, leading to substantial improvements on both specific grammatical functions and overall parsing accuracy when parsing Swedish using MaltParser. Their initial error analysis showed that the most frequent errors involved adverbial relations, this due to the PP-attachment problems, which we also found in

[1] http://stp.lingfil.uu.se/~nivre/research/Talbanken05.html

our error analysis. The parser also, just as we found with Norwegian, consistently confused subject and object constituents. This was caused by ambiguities in the word order and morphological marking. To resolve these ambiguities they examined grammatical features in addition to syntactic categories and linear word order. They focused on properties of animacy, definiteness, person, case, finiteness and voice, in addition to information on pronoun type, and experimented with combinations of these features. The final experiment resulted in an error reduction of about 50 % for the errors specifically targeted following the initial error analysis. Unlike the NDT, Talbanken05 contains information about animacy for nominals, voice and multiple tags for the various adverbs.

Inspired by the work of Øvrelid and Nivre (2007) we decided to investigate the effect of morphological agreement and finiteness of verbs on parsing accuracy. In addition to this we decided to see how the parsing accuracy would be affected by the type feature and the hum tag for pronouns.

4.3.1 Type, animacy & combinations of features

Type

This category contains information regarding whether or not a token belongs to a certain sub-class, meaning, determiners will carry information about whether it is possessive (possesiv), demonstrative (demonstrativ) etc., while pronouns will be marked as demonstrative, quantifying, personal, reflexive etc. This is not a morphological feature as such and we therefore decided to look at this linguistic feature separately from the other morphological features. We decided to run the same experiments with the type feature as we have done previously with the other morphological features in order to compare its effect on parsing accuracy. We reused the feature models from the previous experiments, but altered them to apply for the type tag when necessary.

The results are presented in table 4.12, the POS + type shows the results from merging the part-of-speech tag with the type information, while the Type(individual) is the result of looking at the effect the type tag has on parsing accuracy. We find that the scores from our experiments with type (Type (individual) in table 4.12) and its effect on parsing accuracy is close to the highest scores in our experiments where we looked at the effect of the individual morphological features. The results from combining type

	Split		Round-robin	
	LAS (%)	UAS (%)	LAS (%)	UAS (%)
Bokmål				
Baseline	83.15	86.76	85.14	88.47
POS + type	88.33	90.88	89.51	91.98
Type (individual)	87.98	90.74	89.43	91.84
Nynorsk				
Baseline	83.56	87.60	83.82	87.44
POS + type	88.86	91.64	88.73	91.25
Type (individual)	88.55	91.43	88.87	91.34

Table 4.12: Results from the experiments with type

(POS + Type in table 4.12) with part-of-speech, such as in table 4.5 and 4.6, are also similar to the best results obtained in the experiments with morphological features merged with part-of-speech tags.

From previous experiments we found that the morphological features giving the highest scores overall was gender, number and definiteness, though number and definiteness seemed to have a bigger impact on parser accuracy than gender. From the experiments with type we found that this feature also seemed to have a bit higher impact on parsing accuracy than other features. Therefore it felt like a natural thing to investigate the effect of number (N), definiteness (D) and type (T) together. We also wanted to see if adding gender (G) to this combination of features would have an effect on the parsing. For the experiments we constructed feature models making MaltParser extract information only about these features. Other than that, these experiments were conducted in the same way as previous experiments.

We found that the combination of *number*, *type* and *definiteness* (in table 4.13 and 4.14 it is presented as N + T + D) gave an overall increase in parsing accuracy for both bokmål and nynorsk. The combination of *gender*, *number*, *type* and *definiteness* (in table 4.13 and 4.14 it is presented as G + N + T + D) gave a very minor increase in accuracy compared to the combination of *number*, *type* and *definiteness*. These results are to us quite interesting since they are close to the results we got in the experiments with morphological features were we combined all the morphological

	Split		Round-robin	
	LAS (%)	UAS (%)	LAS (%)	UAS (%)
Baseline	83.15	86.76	85.14	88.47
N + T + D	88.29	90.91	88.94	91.35
G + N + T + D	88.45	90.99	88.96	91.39

Table 4.13: Bokmål: experiments with morphological features combined

	Split		Round-robin	
	LAS (%)	UAS (%)	LAS (%)	UAS (%)
Baseline	83.56	87.60	83.82	87.44
N + T + D	88.46	91.18	88.26	90.81
G + N + T + D	88.62	91.26	88.29	91.10

Table 4.14: Nynorsk: experiments with morphological features combined

features.

Animacy: Hum

Øvrelid (2005) describes animacy as an inherent property of the referents of nouns, indicating how alive the referent is. It is a semantic property in Norwegian, and for some languages it is considered a grammatical property as well. In NLP animacy is usually a binary property (animate vs. inanimate). Animacy is known as an influencing factor in a range of different grammatical phenomena in various languages and it seems to be correlated with important linguistic concepts such as agentivity and discourse salience. Animacy of a noun is thought to be relevant for several different kinds of Natural Language Processing applications such as conference resolution, parsing and generation. Øvrelid (2005) investigates animacy by looking at classification based on morphosyntactic corpus frequencies with Norwegian nouns. Unlike us, Øvrelid (2005) extracts information about animacy for a set of Norwegian nouns and experimented with automatic classification using decision-tree classifiers. The only morphological tag indicating animacy in the Norwegian Dependency Treebank is the *hum* tag for pronouns, indicating whether or not the pronoun's antecedent is human. We decided to include this as an

	Split		Round-robin	
	LAS (%)	UAS (%)	LAS (%)	UAS (%)
Bokmål				
Baseline	83.15	86.76	85.14	88.47
POS + hum	86.85	89.53	88.47	91.02
Hum (individual)	86.91	89.63	88.55	91.01
Nynorsk				
Baseline	83.56	87.60	83.82	87.44
POS + hum	87.64	90.61	87.56	90.30
Hum (individual)	87.53	90.53	87.62	90.28

Table 4.15: Results from the experiments with the hum tag

individual feature and look into the effect of the hum tag on parsing. The experiments were conducted using the same methods as those for type, however, this time we changed the feature models from the previous experiments looking at individual features, to apply for the column containing information regarding hum.

From table 4.15 we find that the hum tag does have a positive effect on parsing accuracy compared to the baseline experiments, but it does not obtain quite as good results as the type tag in table 4.12. The highest labeled attachment score for the hum tag in the experiments looking at the effect individual features (Hum (individual)) on parsing accuracy is 88.55 % in the round-robin data set for bokmål, while the highest score for type is 89.43 %, also in the round-robin data set for bokmål. The unlabeled attachment score for the hum tag is also lower than the score for type, while type obtained an unlabeled attachment score of 91.84 % in the round-robin data set for bokmål, the highest UAS for the hum tag is 91.01 % for the same data set.

Agreement

Almost all languages, although to a varying degree, show some morphological agreement via the inflection of noun, adjectives, verbs, and determiners (Hohensee and Bender 2012). Norwegian shows some degree of morphological agreement between nouns and adjectives. The adjective can be positioned both before and after the noun it agrees with, and

	Agreement		Baseline	
	LAS (%)	UAS (%)	LAS (%)	UAS (%)
Bokmål (Split)	88.16	90.83	83.15	86.76
Bokmål (Round-robin)	89.50	91.89	85.14	88.47
Nynorsk (Split)	88.48	91.38	83.56	87.60
Nynorsk (Round-robin)	88.93	91.43	83.82	87.44

Table 4.16: Result from experiments done with the intention of capturing agreement using gender, number and definiteness on a token and its head.

will agree in gender and number with the noun. There is also agreement between verbs in the perfect participle acting as adjectives for a noun. In the Norwegian Dependency Treebank nouns, adjectives and determiners are tagged with information regarding gender, number and definiteness, while pronouns are marked with information regarding gender and number. Inspired by the results obtained by Hohensee and Bender (2012), we wanted to see if we could capture agreement by only altering the feature models, and if this would have an impact on the parsing accuracy. We decided to find out if we could capture the effect of agreement using gender, number and definiteness and part-of-speech.

The feature model in these experiments were designed so that MaltParser would look at the features mentioned for each token. At the same time the system was asked to look at the same features for the head of the given token. To capture agreement we experimented with combining features for a token and its head into one feature using the feature map functions *merge* and *merge3* in MaltParser. That is, the features were constructed by nesting merge functions so that we would look at both the definiteness, gender and number of a token at the same time as we looked at definiteness, gender and number of the tokens head.

The results from a series of experiments are presented in table 4.16. As we can see from table 4.16, the experiments resulted in an increase in parsing accuracy. In table 4.16 the results from our experiments are presented along side the results from the baseline experiments. Our effort to capture agreement gave us an increase in labeled attachment score of 5.01 % for the split data set in bokmål, and the unlabeled attachment score increased with 4.07 % for the same data set, which to us is quite a great improvement. We got a slightly lower increase of 4.36 % in the

	Split		Round-robin	
	LAS (%)	UAS (%)	LAS (%)	UAS (%)
Baseline	83.15	86.76	85.14	88.47
N + T + D	88.14	90.72	89.55	91.94
G + N + T + D	88.12	90.79	89.55	91.93

Table 4.17: Bokmål: experiments with morphological features combined with agreement. N + T + D = Agreement using number, type and definiteness. G + N + T + D = agreement using gender, number, type and definiteness.

labeled attachment score for the round-robin data set in bokmål. In the data sets in nynorsk the labeled attachment score increased by 4.92 % in the split data set, and by 5.11 % in the round-robin data set. The unlabeled attachment score increased by 3.78 % for the split data set and we also see an increase in the score for the round-robin data set of 3.99 %. We were a bit surprised to find an increase of more than 5 % in the labeled attachment score, which we think a quite substantial increase.

We further wanted to see if we could use the feature models from the experiments with agreement in combination with other features too see if this would obtain an even greater increase in parsing accuracy or not. From the experiments combining type with morphological features we found that the combination of *number*, *type* and *definiteness* (in table 4.13 and 4.14) gave an overall increase in parsing accuracy for both bokmål and nynorsk. The combination of *gender*, *number*, *type* and *definiteness* (in table 4.13 and 4.14 it is presented as G + N + T + D) gave a small increase in accuracy compared to the combination of *number*, *type* and *definiteness*. In these experiments we combined the feature models from the experiments with agreement, and the experiments with combination of the features mentioned above. This resulted in feature models where we looked at the features individually, but also where we merged the features into new features. When merging the features into new features, we combined that with looking at the same features in the head of the features we merged. E.g. when creating a feature looking at number, gender, definiteness, and type for the top token on the stack, we combined, using the merge option, that with looking at the same features on the head of the top token on stack.

	Split		Round-robin	
	LAS (%)	UAS (%)	LAS (%)	UAS (%)
Baseline	83.56	87.60	83.82	87.44
N + T + D	88.53	91.36	88.75	91.28
G + N + T + D	88.47	91.36	88.91	91.46

Table 4.18: Nynorsk: experiments with morphological features combined with agreement. N + T + D = Agreement using number, type and definiteness. G + N + T + D = agreement using gender, number, type and definiteness.

The combination of *gender*, *number*, *type*, *definiteness* and agreement gave us an increase in both labeled and unlabeled attachment score compared to the score from the baseline experiments as we can see in table 4.17 and 4.18. However, compared to the results from the experiments with agreement, seen in table 4.16 the additional morphological information hardly made any difference to the scores. When we compare the results to the experiments only looking at the features *gender*, *number*, *type* and *definiteness* presented in table 4.13 and 4.14 we only see a slight increase in the scores.

4.3.2 Finiteness

Øvrelid (2008) investigates the effect of verbal features on data-driven dependency parsing of Swedish. The paper illustrates how information on the morphosyntactic properties of tense (present, past, imperative, subjunctive, infinitive and supine) and voice (active or passive) for all verbs can give significant improvements in the analysis of syntactic arguments. To test the extent to which tense may be reduced to finiteness, she mapped the tense features to features expressing the binary category of finiteness, which resulted in improvement on accuracy.

In the NDT verbs carry information about tense. In Norwegian we divide conjugated forms of verbs in five categories, which can be either finite or infinite. The finite forms of the verb is *present tense*, *past tense* and *imperative*, while *infinitive* and *past participle* are infinite forms of the verb. We made some experiments where used information about finiteness of verbs. As mentioned earlier, we created a separate column in the reformatted treebank for information regarding finiteness. The only morpho-

	Finiteness		Baseline	
	LAS (%)	UAS (%)	LAS (%)	UAS (%)
Bokmål (Split)	86.76	89.54	83.15	86.76
Bokmål (Round-robin)	88.47	90.98	85.14	88.47
Nynorsk (Split)	87.44	90.52	83.56	87.60
Nynorsk (Round-robin)	87.62	90.29	83.82	87.44

Table 4.19: Result from experiments done with the intention of capturing finiteness of verbs.

logical feature used in this experiment is the the information from this column. The settings for MaltParser is as mentioned the same as in the other experiments.

This resulted in a slight increase of parsing accuracy as we can see from table 4.19. Compared to the results from the baseline experiments, we see an increase in both labeled and unlabeled attachment score, but these score are lower the those from our previous experiments.

4.4 Final evaluation

We wanted to see if we could create a feature model using the combination of the features that gave us the best result from the experiments presented above. We found that merging the form of a preposition with its part of speech tag gave quite a high increase in parsing accuracy. We also found that the combination of the features *gender*, *number*, *type* and *definiteness* had a positive effect on parsing accuracy without using the same amount of resources as one would if we combined all the features. We also obtained fairly good results from our experiments trying to capture agreement. Therefore, we decided to combine the feature models from these experiments to see if we could obtain a higher parsing accuracy than what we have been able to obtain in the previous experiments. During the entire process of experimenting with linguistic features we used three different feature models. The split data set used the same feature model for both bokmål and nynorsk. The round-robin data set used different feature models for bokmål and nynorsk. While the feature models for the split data set used Nivre's arc eager algorithm, the round-robin data sets both used the Stack projective algorithm. We wanted to investigate if we had found the best feature models for the data sets, or

	Split		Round-robin	
	LAS (%)	UAS (%)	LAS (%)	UAS (%)
Split	88.83	91.30	89.03	91.63
Round-robin BM	89.06	91.58	89.31	91.76
Round-robin NN	89.05	91.64	89.24	91.69

Table 4.20: Bokmål: Dependency Parsing results with our feature models. Round-robin BM: the round-robin feature model for the data set in bokmål. Round-robin NN: the feature model used for the round-robin data set in nynorsk

	Split		Round-robin	
	LAS (%)	UAS (%)	LAS (%)	UAS (%)
Split	88.66	91.07	89.21	91.81
Round-robin BM	89.00	91.32	89.36	91.97
Round-robin NN	88.97	91.33	89.43	92.03

Table 4.21: Nynorsk: Dependency Parsing results with our feature models. Round-robin BM: the round-robin feature model for the data set in bokmål. Round-robin NN: the feature model used for the round-robin data set in nynorsk

if one model would give the best results for all the data sets. We decided to test the models with the final held-out test set.

The results are presented in table 4.20 and table 4.21. Based on our result it seems like the best choice for parsing nynorsk might be the round-robin feature model used for the experiments in nynorsk combined with the stack projective algorithm. Based on our results we see that one of the round-robin feature models together with the stack projective algorithm is the best choice also when parsing bokmål.

We then decided to test MaltParser with the default settings used in the baseline experiments, the models from the experiments with MaltOptimizer, and our models from the experiments above, in order to compare our models to something. The results are presented in table 4.22 and table 4.23. The model from the baseline experiments was partly able to obtain a better result then it obtained in the baseline experiments. The settings and models from the experiments with MaltOptimizer resulted in score that were close to the score we got in the initial experiments with

	Split		Round-robin	
Model	LAS (%)	UAS (%)	LAS (%)	UAS (%)
Baseline	84.32	87.80	84.15	87.75
MaltOptimized	89.30	91.74	89.65	91.98
Our	88.80	91.30	89.32	91.73

Table 4.22: Bokmål: Dependency Parsing results with the final held-out test set.

	Split		Round-robin	
Model	LAS (%)	UAS (%)	LAS (%)	UAS (%)
Baseline	83.42	86.94	84.11	87.84
MaltOptimized	89.25	91.41	89.36	91.89
Our	88.66	91.02	89.43	92.03

Table 4.23: Nynorsk: Dependency Parsing results with the final held-out test set.

	Split		Round-robin	
Model	BM (ms)	NN (ms)	BM (ms)	NN (ms)
Baseline	53561	65971	58828	50752
MaltOptimizer	183871	208354	121326	143390
Our	325257	343827	192834	198042

Table 4.24: Learning time from the models tested on the final held-out data sets. BM = bokmål. NN = nynorsk.

MaltOptimizer. The scores from our models was on one occasion higher than the scores from the experiments with MaltOptimizer, other than than our score were a bit lower. The labeled attachment score for our models were 88.80 % and 89.32 % for bokmål and 88.66 % and 89.43 % for nynorsk. The unlabeled attachment score for bokmål is 91.30 % and 91.73 %, while it is 91.02 % and 92.03 %.

In this thesis we have mentioned time and efficiency of the parser on several occasions. How much time used for training and testing a parser is to many important, and we all wish to have efficient systems that require a minimum of resources. Below we present the time MaltParser used for training and parsing the models from the experiments above.

Model	Split		Round-robin	
	BM (ms)	NN (ms)	BM (ms)	NN (ms)
Baseline	4912	4833	5222	4887
MaltOptimizer	9055	9176	10527	8519
Our	14607	12641	15239	10567

Table 4.25: Parsing time from the models tested on the final held-out data sets. BM = bokmål. NN = nynorsk.

From table 4.24 we find that our models take a lot more time learning than the models used for the baseline experiments and the experiments with MaltOptimizer. Our models are much bigger and more detailed, so this comes as no surprise. In table and 4.25 we can see that our models also use more time testing than the other models.

4.5 Domain sensitivity

Domain adaption is the task where the goal is to adapt parser from a source domain (usually with a large amount of available data) to a target domain of interest (which usually has a small amount of available resources). The most common training material used in dependency parsing is newspaper articles, and for this domain parsers often obtain more than 90 % accuracy for English. However, we see a drop in parsing accuracy for the same parsers when parsing out-of-domain text such as blogs or web pages. That indicates that most dependency parser are still not robust enough for practical use in Natural Language Applications such as machine translation or relation extraction.

In 2007 the Shared Task organized by The annual Conference on Computational Natural Learning (CoNLL) was, as in 2006, multilingual dependency parsing, but the task from previous year was extended by adding a second track for domain adaptation (monolingual). The task was to adapt a parser for English news text to other domains. The participants could use unlabeled data from the target domains, which was biomedical, chemical abstracts and parent-child dialogs. There were both a closed class and an open class. In the closed class the parser had to be trained using the English training set for the multilingual track. The open class allowed for the parser to be trained on any external resources. The best results on this task was a labeled attachment score of 81.1 % on

the test set of chemical abstracts, from a parser who got labeled attachment score of 89.0 % for the English test set in the multilingual track[2] (Nivre 2007).

Domain adaptation is an aspect of parser robustness. Hassel et al. (2011) studies the effect of using an existing parser, pre-trained on general Swedish, to parse clinical text in Swedish. Since syntactic parsers that are tailored to accommodate for the distinctive properties of clinical language are both time consuming and costly to build, it is interesting to see whether a pre-trained model is directly transferable to a new domain. Clinical text differs from general text in terms of both language, vocabulary and content. The main finding in this paper is that the morphological characteristics of Swedish clinical language do not differ greatly from general language. They therefore conclude that existing tools can be used successfully when it comes to PoS information.

Plank and Søgaard (2012) stress the fact that sampling training and test data from similar sources (same domain or genre), may cause us to overfit our models. This will again lead to an over-adaption of our parser, which in turn will cause a drop in parsing accuracy when the parser is tested on a different domain.

Just as clinical text differ from general text, newspaper text differ from legal documents and blog posts, which is what the NDT consists of. Using the settings from previous experiments, we wanted to investigate if the parser was sensitivity to different domains in the Norwegian Dependency Treebank. We decided to use the text from newspapers and legal documents as training data, and then use the blog text as test data. Blogs are user generated content and usually very different from newspaper text and legal documents, that is, they can be considered to belong to different domains. In these experiments we divided the data from the NDT based on source. We used the blog texts as test data and the training set was constructed by combining the texts from newspapers and legal documents. The feature model (Our models) constructed for this experiments combined the settings giving us the highest score in the experiments mentioned earlier in this chapter. We also tested the model we got from the experiments with MaltOptimizer and the baseline model, so that we would have something reasonable to compare our result with.

[2]It is worth mentioning that the biggest issue when adapting existing parsers to a new domain was not necessary the domain, but the mapping from the native annotation used of the parser to the annotation provided in the data set used in the shared task.

	Split		Round-robin	
	LAS (%)	UAS (%)	LAS (%)	UAS (%)
Baseline	83.15	86.76	85.14	88.47
Baseline D	79.82	87.03	79.54	83.90
MaltOptimized	86.36	91.09	83.84	87.34
Our models	84.33	89.97	84.07	88.07

Table 4.26: Bokmål: experiments with domain sensitivity. Baseline = the results from our baseline experiments. Baseline D = the model and settings used in the baseline experiments, trained and tested on the domain data sets. MaltOptimized = the model and settings used in the experiments with MaltOptimizer, trained and tested on the domain data set. Our models = Our models trained and tested on the domain data sets.

	Split		Round-robin	
	LAS (%)	UAS (%)	LAS (%)	UAS (%)
Baseline	83.56	87.60	83.82	87.44
Baseline D	77.06	82.21	77.97	83.40
MaltOptimized	82.36	85.80	80.22	84.41
Our models	80.34	84.30	79.98	83.94

Table 4.27: Nynorsk: experiments with domain sensitivity. Baseline = the results from our baseline experiments. Baseline D = the model and settings used in the baseline experiments, trained and tested on the domain data sets. MaltOptimized = the model and settings used in the experiments with MaltOptimizer, trained and tested on the domain data set. Our models = Our models trained and tested on the domain data sets.

The results from the experiments with domain sensitivity is presented in table 4.26 and table 4.27. From the experiments we found a reduction in parsing accuracy for all the data sets in both nynorsk and bokmål. When we tested the baseline model on the blog text the labeled attachment score decreased from 83.15-85.15 % to 77.07-79.82 %. The labeled attachment score which was between 86.76 % and 88.47 % in the baseline experiments, decreased to 82.21-87.03 %.

The models from the MaltOptimizer experiments handled the new do-

main better than the baseline model. The labeled attachment score for bokmål is 86.36 % for the split data set, and 83.84 % for the round-robin data set. The unlabeled attachment score for the split data set is 91.09 %, while it score for the round-robin data set is 87.34 %. In nynorsk, we got slightly lower scores where the split data set got a labeled attachment score of 82.36 % and a unlabeled attachment score of 85.80 %. The round robin data set in nynorsk obtained a labeled attachment score of 80.22 %, while the unlabeled attachment score is at 84.41 %.

The results for the model constructed by us is mostly a bit lower than the scores obtained by the MaltOptimizer models. For bokmål the labeled attachment score ended at 84.33 % for the split data set and 84.07 % for the round-robin data set. The unlabeled attachment score for the split data set is 89.97 % and the score for the round-robin data set is 88.07 %. The data sets in nynorsk got lower scores in the domain experiments with our model as well. The split data set got a labeled attachment score of 80.34 % and an unlabeled attachment score of 84.30 %. The round-robin data set obtained a labeled attachment score of 79.98 % and an unlabeled attachment score of 83.94 %.

Chapter 5

Conclusion

This thesis and the work done during the production of it was made with the intention of gaining a better understanding of the effect linguistically motivated features can have on data-driven dependency parsing for Norwegian. The study of linguistically motivated features and their effect on parsing accuracy in dependency parsing has been studied in a range of different languages, but since there has been no dependency treebanks like the NDT this has not been done for Norwegian. In this thesis we have worked with the new dependency treebank for Norwegian, NDT, and MaltParser. Chapter 2 gave an introduction to both the treebank and data-driven dependency parsing with MaltParser. In chapter 3 we presented the first results from running MaltParser and MaltOptimizer with the NDT, combined with an error analysis. This gave us resources we used when conducting the experiments presented in chapter 4 where we experimented with extended part-of-speech tags, various linguistic features and domain sensitivity. In this chapter we will first sum up our main findings from the experiments and briefly discuss the work we have done. Finally, we will consider the opportunities for future work.

5.1 Conclusion

From the initial experiments and the error analysis we found that common problems in NLP for Scandinavian languages are, not surprisingly, valid for parsing Norwegian. The PP-attachment problem is one of the issues and so are the problems regarding word order confusion. The error analysis revealed that MaltParser's default settings were not ideal for parsing Norwegian. The results from testing and tuning the NDT with

MaltOptimizer revealed that there might be differences between parsing nynorsk and bokmål.

When working on the initial experiments and the experiments with linguistic features we noticed that the split data set in bokmål tended to obtain lower scores than the round-robin data set in bokmål. We noticed the same in the data sets for nynorsk, but here the differences were smaller. However, this tendency was weaker in the final evaluation. In the experiments on domain sensitivity on the other hand, the split data set scored higher than the round-robin data set. This is interesting and can be seen as an indication of the importance of dividing the data right, and also a reminder that that a corpus can only be an approximation of a language.

From the experiments with linguistic features we found that using part-of-speech tags combined with the form of prepositions gave quite a high increase in both labeled and unlabeled attachment score, while this method required very little preprocessing and no need for morphological features. We also found that the most prominent morphosyntactic features for parsing accuracy seem to be definiteness, number, gender and type. Definiteness, number and gender are the features we used to capture agreement between a token and its head, which improved the labeled and unlabeled attachment scores quite a bit. All of these features combined gave us final results that were very close to those obtained when using the MaltOptimizer system, as we can see in table 5.1 and 5.2 which gives an overview of the best results from our experiments compared to the baseline experiments.

We found that features such as case which are seen as important for parsing accuracy, did not have a great impact on parsing Norwegian. This is most likely due to the fact that the use of case in Norwegian is marginal. We did not obtain good results with features capturing animacy or finiteness either. This is probably due to the hum tag in the NDT not really representing animacy but rather being an indication of whether or not the pronoun's antecedent is human. Our experiments with finiteness were quite shallow due to the lack of morphological information regarding verbs. If one were to extract more information regarding verbs, one could possibly find this to be a more important feature for parsing accuracy.

The merging of prepositions with its part-of-speech tag, and the use

	Split		Round-robin	
Model	LAS (%)	UAS (%)	LAS (%)	UAS (%)
Baseline	83.15	86.76	85.14	88.47
Malt Optimized	88.69	91.14	89.89	92.18
POS + FORM	87.06	89.82	88.69	91.26
Number	87.89	90.58	89.12	91.56
Definiteness	87.85	90.50	89.17	91.63
Type	87.98	90.74	89.43	91.84
Final evaluation				
Baseline	84.32	87.80	84.15	87.75
MaltOptimized	89.30	91.74	89.65	91.98
Our	88.80	91.30	89.32	91.73

Table 5.1: Bokmål: a summary of the best results.

of the linguistic features *definiteness*, *number*, *gender* and *type* combined with the features used to capture agreement improved the accuracy when parsing Norwegian compared to MaltParser's default model. When testing on the final held-out data set our model obtained results comparable to those obtained by MaltOptimizer.

In our final evaluation, the model made from the results of our experiment obtained a labeled attachment score for the split data set in bokmål which was 4.48 % higher than the score obtained by the baseline model. The unlabeled attachment score for this data set had the lowest increase from the baseline model with an increase of 3.5 %. The round robin data set in bokmål obtained a labeled attachment score which was 5.17 % higher than that of the baseline model, while the unlabeled attachment score increased by 3.90 %.

For the data sets in nynorsk the increase in labeled attachment score for the split data set was 5.24 % from the baseline model and the unlabeled attachment score for the same data set increased by 4.08 %. The biggest increase in both labeled and unlabeled attachment score was obtained by the round-robin data set for nynorsk. This set obtained a labeled attachment score which was 5.32 % higher than the baseline and the unlabeled attachment score increased by 4.19 %.

Model	Split LAS (%)	Split UAS (%)	Round-robin LAS (%)	Round-robin UAS (%)
Baseline	83.56	87.60	83.82	87.44
Malt Optimized	88.93	91.58	89.45	91.80
POS + FORM	87.59	90.54	87.98	90.78
Number	88.17	91.11	88.15	90.77
Definiteness	88.24	91.13	88.19	90.83
Type	88.55	91.43	88.87	91.34
Final Evaluation				
Baseline	83.42	86.94	84.11	87.84
MaltOptimized	89.25	91.41	89.36	91.89
Our	88.66	91.02	89.43	92.03

Table 5.2: Nynorsk: a summary of the best results.

Something common to ask oneself at the end of a research project is whether or not one focused on the right things and if one measured what one intended to measure. In this thesis we decided to mainly use the resources available in the NDT rather than use a lot of time and effort on preprocessing. The consequences of this choice is of course that several features that have been proven important in NLP have been left unexplored, partially or completely. These are features such as animacy, finiteness and voice. However, the advantage of doing less preprocessing is that it will be easier for others to remodel our experiments, and this increases the reliability of our our research. We believe that our experiments did indeed capture the linguistic phenomena we intended to capture, but again, this should always be verified by someone else. What is worth thinking about when it comes to the evaluation of our experiments is that there is a discussion regarding whether or not labeled and unlabeled attachment scores are actually suitable measure of accuracy. Labeled and unlabeled attachment scores are as mentioned, the traditional evaluation metrics in dependency parsing and are the evaluation scores mainly used in this thesis. However, Tsarfaty et al. (2011) argue that these evaluation methods might not be good enough. The say that:

> As it turns out, however, such evaluation procedures are sensitive to the annotation choices in the data on which the parser was trained. [...] The consequence of such annotation

> discrepancies is that when we compare parsing results across different experiments, even ones that use the same parser and the same set of sentences, the gap between results in different experiments may not reflect a true gap in performance, but rather a difference in the annotation decisions made in the respective treebanks.

Several methods have been proposed to resolve this issue, but they have a number of disadvantages. They suggest a procedure for comparing dependency parsing results for different experiments based on the idea of converting dependency trees to functional trees, generalizing functional trees and using distance-based metrics (Tsarfaty et al. 2011). This system has been produced and is named TedEval[1]. The disadvantages of labeled and unlabeled attachment score could have an effect on the results presented in this thesis and is something that affects the validity of our results.

5.2 Future work

The opportunities for future work on dependency parsing for Norwegian seem to us at this point never ending. There is the possibility to process the NDT to extract more information regarding morphological features and in that way enrich the treebank. Øvrelid and Nivre (2007) investigate the exact influence of features on the parsing accuracy for specific linguistic constructions in Swedish. By looking at features for animacy, definiteness, person, case, finiteness, voice, tense and pronoun type, and combinations of these features. In the final experiment they obtained an error reduction of about 50 % for the errors specifically targeted. The errors found in their error analysis are similar to those we found in the baseline experiments and we think it would be interesting to replicate the experiments they did, targeting the effect of features on specific constructions using the NDT. However, this will require some feature extraction and additions to the corpus since the annotation of the NDT is less detailed than the Swedish Talbanken05.

There are also a lot options available in MaltParser that have been unexplored. It would be interesting to see the effect of using other parsing algorithms, we did not find the time to explore the Covington algorithm.

[1] http://www.tsarfaty.com/unipar/

Kübler et al. (2009, p. 36) suggest the use of Yamada's algorithm which "performs multiple passes over the input while still exploring only a single path through the transition system in each pass", rather than the single-pass strategy in Nivre's algorithms. It would also be interesting to use *beam search* in combination with MaltParser which is a competitive approach to dependency parsing as shown by Zhang and Clark (2008). It would also be interesting to evaluate contrasting parsers on the NDT since there are constantly being developed new techniques for dependency parsing. I find the dependency parsing systems using perceptrons and beam search to be particularly interesting.

Another field in dependency parsing is the use of semantic features, but this again would require more resources. Extracting semantic information would have been interesting to look into since dependency relations often resembles semantic relations and semantic information have been proved to increase the parsing accuracy in dependency parsing.

Appendix A

Table of morphosyntactic tags for the Oslo-Bergen Tagger

The Oslo-Bergen tagger (OBT) is a morphological and syntactic tagger. It was developed by the University of Oslo and Uni Computing in Bergen. The OBT basically consists of a preprocessor with multitagger (uses the lexicon Norsk ordbank) combined with a compound analyser, a grammar module used to solve morphological and/or syntactic disambiguation and a statistical module to remove remaining morphological ambiguity (University of Oslo and Uni Computing in Bergen and University of Southern Denmark, Odense). The table A.1 show most of the morphosyntactic tags used in the NDT.

Morphosyntactic tags	
Adj	
Gender	m/f, nøyt, fem
Number	enf, fl
Type	<adv>, <ordenstall>, <perf-part> <pres-part>, fork
Definiteness	ub, be
Degree	pos, kom, sup
Adv	
Det	
Gender	mask, nøyt, fem
Number	enf, fl
Type	dem, dem <adj>, <adj> forst, <adj> kvant, kvant, poss, poss res, poss høflig, sp, forst
Definiteness	ub, be
Inf-merke	
Interj	
Konj	
Type	<adv>, clb
Prep	
Pron	
Gender	fem, mask, nøyt, mask fem
Number	enf, fl
Type	hum res, hum sp, pers, pers hum, pers høflig, poss hum sp, refl, sp, res
Definiteness	ub, be
Person	1, 2, 3
Case	nom, akk
Sbu	
Type	<spørreartikkel>
Subst	
Gender	mask, nøyt, fem
Number	enf, fl
Type	appell, prop, fork
Definiteness	ub, be
Ukjent	
Verb	
Tense	pres inf pass inf, pres, pret, perf-part, imp, pass

Table A.1: Table of morphosyntactic tags for the Oslo-Bergen Tagger. Taken from http://tekstlab.uio.no/obt-ny/english/tags.html

Appendix B

Data format

B.1 CoNLL data format

As mentioned in the thesis, Maltparser use treebanks in CoNLL format as input. A treebank is a large collection of annotated dependency trees like the one shown in the B.1. CoNLL is a format used for dependency treebanks.

Table B.1 is a representation of parts of the sentence *Det sies at menneskets rettferdighetsfølelse er medfødt*. from the Norwegian Dependency Treebank in CoNLL format, while figure B.1 show the representation of the same sentence as a dependency structure. The CONLL format is a 10 column tab-separated table. Each line in the table represents a token. The columns (from left to right) consists of a token index (ID), word form (FORM), lemma (LEMMA), coarse-grained part-of-speech tag (CPOS) , fine-grained part-of-speech tag (POS), morphological features (FEATS), index of the head (HEAD), dependency relation (DEPREL), projective head (PHEAD) and projective dependency relation (PDEPREL). The two final columns are left blank in the Norwegian Dependency Treebank and

ID	FORM	LEMMA	CPOS	POS	FEATS	HEAD	DEPREL
1	Det	det	pron	pron	nøyt|ent|pers|3	2	FSUBJ
2	sies	si	verb	verb	pres|pass	0	FINV
3	at	at	sbu	sbu	_	6	SBU

Table B.1: An example of a sentence from the Norwegian Dependency Treebank represented in ConLL format

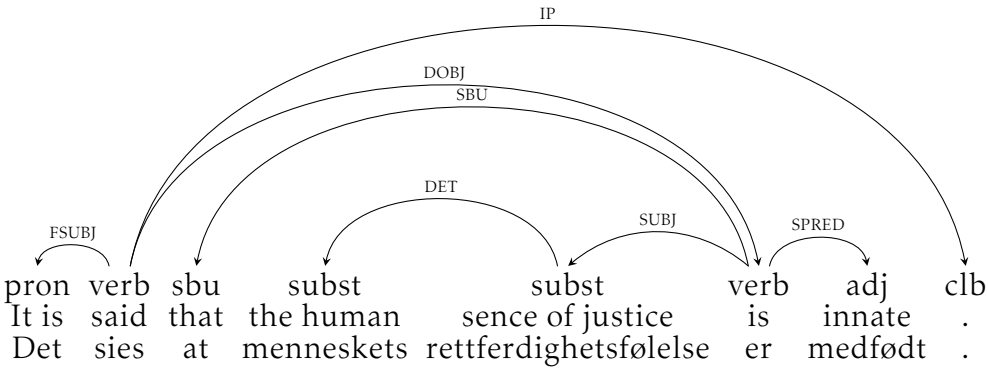

Figure B.1: An example a sentence from the Norwegian Dependency Treebank represented as a dependency graph.

are therefore not presented in the table.

B.2 Reformatting the Norwegian Dependency Treebank

As we can see from tableB.1 all the morphological features of a token is merged into one column. MaltParser does not include a functionality to look at one individual feature at the time. So, in order to investigate the effect of the different morphological features on parsing accuracy we had to reformat the treebank. MaltParser comes with the opportunity to change the input format. The files specifying the data format is a quite simple XML file. The file has one data format element with the attribute name. This element has one or more column attributes, containing all the information of each column. We can see an example of the default XML file in figure B.2.

In order to reformat the treebank, we used the information in the FEATS column and distributed those features over ten new columns that were added to the data format specification file. We created new columns for

```
<?xml version="1.0" encoding="UTF-8"?>
<dataformat name="conllx">
       <column name="ID" category="INPUT" type="INTEGER"/>
       <column name="FORM" category="INPUT" type="STRING"/>
       <column name="LEMMA" category="INPUT" type="STRING"/>
       <column name="CPOSTAG" category="INPUT" type="STRING"/>
       <column name="POSTAG" category="INPUT" type="STRING"/>
       <column name="FEATS" category="INPUT" type="STRING"/>
       <column name="HEAD" category="HEAD" type="INTEGER"/>
       <column name="DEPREL" category="DEPENDENCY_EDGE_LABEL"
           type="STRING"/>
       <column name="PHEAD" category="IGNORE" type="INTEGER"
           default="_"/>
       <column name="PDEPREL" category="IGNORE" type="STRING"
           default="_"/>
</dataformat>
```

Figure B.2: The CoNLL data format specification file taken from the MaltParser user guide

ID	FORM	...	GENDER	NUMBER	TYPE	...
1	Det	...	nøyt	ent	pers	...
2	sies	...	_	_	_	...
3	at	...	_	_	_	...

Table B.2: An example of a sentence from the Norwegian Dependency Treebank represented in ConLL format

each morphosyntactic tag; gender, number, type, definiteness, person, case and degree. We also created a separate column for the hum tag and an additional column carrying information about whether a verb being finite or infinite. Table B.2 gives us an idea of how the files in the treebank ended up looking. In the experiments regarding part of speech tags we added the form or lemma of each token to the part of speech tag. A part of speech tag e.g. *pron* would end up looking like this *pron-Det* if the token was *Det*. This was done using a similar system as the one we used to create the additional columns.

We created a system for the reformatting, written in Python. Since we wanted to measure the effect of individual features on parsing accuracy,

we had to create a system that converts the extended version of the corpus back to its original format. To validate the reformatting we used the validation system facilitated by MaltOptimizer, validateFormat.py. As a part of the experiments we modified each feature model so that we could control which feature MaltParser looked at.

References

Ballesteros, M. and J. Nivre (2014, 8). Maltoptimizer: Fast and effective parser optimization. *Natural Language Engineering FirstView*, 1–27.

Bender, E. M. (2013). *Linguistic Fundamentals for Natural Language Processing: 100 Essentials from Morphology and Syntax*. Morgan & Claypool Publishers.

Black, E., F. Jelinek, J. Lafferty, D. M. Magerman, R. Mercer, and S. Roukos (1992). Towards history-based grammars: Using richer models for probabilistic parsing. In *Proceedings of the Workshop on Speech and Natural Language*, HLT '91, Stroudsburg, PA, USA, pp. 134–139. Association for Computational Linguistics.

Chomsky, N. (1957). *Syntactic structures*. Mouton.

Faarlund, J. T. (1997). *Norsk referansegrammatikk*. Universitetsforl.

Falk, Y. N. (2001). *Lexical-Functional Grammar An Introduction to Parallel Constraint-Based Syntax*. CSLI Publications.

Hall, J., J. Nilsson, and J. Nivre (2012). Maltparser user guide. `http://www.maltparser.org/userguide.html`. Accessed: 2014-08-18.

Hassel, M., A. Henriksson, and S. Velupillai (2011, May 11-13). Something Old, Something New – Applying a Pre-trained Parsing Model to Clinical Swedish. In *Proc. 18th Nordic Conf. on Comp. Ling. - NODALIDA '11*.

Hohensee, M. and E. M. Bender (2012). Getting more from morphology in multilingual dependency parsing. In *HLT-NAACL*, pp. 315–326.

Hudson, R. A. (1990). *English word grammar*. Blackwell.

Kinn, K., P. K. Eriksen, and P. E. Solberg (2013). Retningslinjer for morfologisk og syntaktisk annotasjon i språkbankens gullkorpus. Technical report, National Library of Norway.

Kromann, M. and S. Lynge (2004). Danish dependency treebank v. 1.0. Department of Computational Linguistics, Copenhagen Business School.

Kübler, S., R. McDonald, and J. Nivre (2009). Dependency parsing. *Synthesis Lectures on Human Language Technologies 2*(1), 1–127.

Lewis, P. M., G. F. Simons, and C. D. Fennig (2014). Ethnologue: Languages of the world, seventeenth edition. http://www.ethnologue.com/statistics. Accessed: 2014-08-20.

McDonald, R. and J. Nivre (2007). Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 122–131.

McDonald, R., F. Pereira, K. Ribarov, and J. 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*, HLT '05, Stroudsburg, PA, USA, pp. 523–530. Association for Computational Linguistics.

Nilsson, J. and J. Nivre (2005). Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, Stroudsburg, PA, USA, pp. 99–106. Association for Computational Linguistics.

Nilsson, J. and J. Nivre (2008). Malteval: An evaluation and visualization tool for dependency parsing. In *In Proceedings of the Sixth International Language Resources and Evaluation. LREC*.

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

Nivre, J. (2006). Two strategies for text parsing. In M. Suominen, A. Arppe, A. Airola, O. Heinämäki, M. Miestamo, U. Määttä, J. Niemi, K. K. Pitkänen, and K. Sinnemäki (Eds.), *A Man of Measure: Festschrift in Honour of Fred Karlsson on his 60th Birthday*, pp. 440–448. A special supplement to SKY Journal of Linguistics 19.

Nivre, J. (2007). Data-driven dependency parsing across languages and domains: perspectives from the conll 2007 shared task. In *Proceedings of the 10th International Conference on Parsing Technologies*, pp. 168–170. Association for Computational Linguistics.

Nivre, J., J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kübler, S. Marinov, and E. Marsi (2007, 5). Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering 13*, 95–135.

Øvrelid, L. (2005). Animacy classification based on morphosyntactic corpus frequencies: some experiments with norwegian nouns. In *Proc. of the Workshop on Exploring Syntactically Annotated Corpora*.

Øvrelid, L. (2008). Finite matters: Verbal features in data-driven parsing of Swedish. In A. . B. N. Ranta (Ed.), *Proceedings of the International Conference on NLP, GoTAL 2008*, LNCS/LNAI Volume 5221. Springer.

Øvrelid, L. and J. Nivre (2007). When word order and part-of-speech tags are not enough – Swedish dependency parsing with rich linguistic features. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP)*, pp. 447–451.

Petrov, S., D. Das, and R. McDonald (2012, may). A universal part-of-speech tagset. In N. C. C. Chair), K. Choukri, T. Declerck, M. U. Doğan, B. Maegaard, J. Mariani, A. Moreno, J. Odijk, and S. Piperidis (Eds.), *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

Plank, B. and A. Søgaard (2012, January). Experiments in newswire-to-law adaptation of graph-based dependency parsers. In *Working Notes of EVALITA 2011*, Rome, Italy. CELCT a.r.l.

Solberg, P. E. (2013). Building gold-standard treebanks for norwegian. In *Proceedings of NODALIDA*, pp. 459–464.

Solberg, P. E., A. Skjærholt, L. Øvrelid, K. Hagen, and J. B. Johannessen (2014, may). The norwegian dependency treebank. In N. C. C. Chair), K. Choukri, T. Declerck, H. Loftsson, B. Maegaard, J. Mariani, A. Moreno, J. Odijk, and S. Piperidis (Eds.), *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland. European Language Resources Association (ELRA).

Tsarfaty, R., J. Nivre, and E. Andersson (2011). Evaluating dependency parsing: robust and heuristics-free cross-annotation evaluation. In *Proceedings of the Conference on Empirical Methods in Natural Lan-*

guage Processing, pp. 385–396. Association for Computational Linguistics.

University of Oslo and Uni Computing in Bergen and University of Southern Denmark, Odense. The oslo-bergen tagger-a grammatical tagger for bokmål and nynorsk. `http://tekstlab.uio.no/obt-ny/english/index.html`. Accessed: 2014-10-13.

Zhang, Y. and S. Clark (2008). A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing using beam-search. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, Stroudsburg, PA, USA, pp. 562–571. Association for Computational Linguistics.

Zwicky, A. M. (1985, 3). Heads. *Journal of Linguistics 21*, 1–29.