UiO : Department of Informatics
University of Oslo

Optimizing a PoS Tag Set for Norwegian Dependency Parsing

Petter Hohle

Master's Thesis Spring 2016

Abstract

This thesis presents a systematic, empirical investigation of how an existing PoS tag set can be modified and optimized for the task of syntactic dependency parsing of Norwegian. The tag set of the Norwegian Dependency Treebank is modified and optimized through experiments with the morphological features in the treebank. The experiments are complemented by evaluation of a range of state-of-the-art PoS taggers and syntactic parsers applied to Norwegian. The results of our work are concrete contributions to the Norwegian NLP community: (i) a data set split (training/development/test) of the Norwegian Dependency Treebank; (ii) a PoS tag set optimized for syntactic dependency parsing of Norwegian; (iii) a PoS tagger model trained on the treebank; and (iv) a syntactic parser model trained on the treebank.

Acknowledgements

First of all, I want to express my sincere gratitude to my supervisors, Erik Velldal and Lilja Øvrelid, for their exceptional guidance and for going out of their way to assist me in overcoming obstacles, large or small. The importance of their meticulous editing and feedback cannot be overstated, and their encouragements and confidence in my research inspired me greatly.

I want to thank my fellow students for the sorely needed coffee and lunch breaks and the staff at the Language Technology Group for including me and making me feel like a peer.

I am very grateful to the National Library of Norway for granting me a scholarship for my thesis work. Furthermore, I want to thank them for their development of the Norwegian Dependency Treebank, which made this thesis possible.

I would also like to thank the developers of the taggers and parsers I utilized, for making their tools and data open-source and publicly available.

Last, but certainly not least, I want to extend my gratitude to my family, for always believing in me and providing encouraging words and their continuous support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Part-of-speech (PoS) tagging is a central component in many Natural Language Processing (NLP) tasks, as it provides morphosyntactic analysis by assigning each word in a particular sentence a part-of-speech tag taken from a predefined set of potential tags, called the tag set. In particular, the resulting PoS tags allow for syntactic parsing, which involves analyzing the structure of the sentence. PoS tag sets are generally developed with the linguistic considerations and preferences of their developers in mind rather than being based on empirical evaluation of what tags provide the most useful linguistic information. Furthermore, PoS tag sets are generally taken for granted and used in an unaltered form instead of being optimized for the task at hand, syntactic parsing.

There are few off-the-shelf NLP tools for Norwegian today. Until now, the most widely used PoS tagger for Norwegian has been the rule-based Oslo-Bergen Tagger, which has served the Norwegian NLP community well over the years. However, numerous aspects of the tagger call for the release of and transition to new tools closer to the state-of-the-art. It is not open-source, nor is it actively maintained, which severely impair its usability. Additionally, due to its rule-based nature, it cannot learn from or be trained on annotated data. In the pre-processing before PoS tagging, it performs tokenization and lemmatization, both of which are useful for several NLP applications. However, as it is not modular, we cannot isolate any of the individual components and run them separately on running text without having to run the entire pipeline.

The newly developed Norwegian Dependency Treebank is the first freely available treebank for Norwegian and contains manually annotated morphosyntactic analyses of sentences in both official written standards of Norwegian, i.e., Bokmål and Nynorsk. The treebank contains information about parts-of-speech and morphological properties such as definiteness, gender and number, and may therefore be used to train and evaluate PoS taggers for Norwegian. Furthermore, the treebank is annotated with syntactic analyses represented in the dependency gram-

mar scheme, which allows for the training and evaluation of syntactic dependency parsers for Norwegian. Due to its very recent development and the lack of a standardized data set split of the treebank, it has not been extensively used to train and evaluate PoS taggers and dependency parsers, until now.

This thesis seeks to identify an optimized part-of-speech tag set for syntactic parsing of Norwegian based on the Norwegian Dependency Treebank as well as present a comparative analysis of the performance of a range of state-of-the-art PoS taggers and syntactic dependency parsers when applied to Norwegian. Through experiments with linguistically motivated tag set modifications, we will examine how different granularities of categories, e.g., different sized tag sets, affect the results of both PoS tagging, and, more importantly for this thesis, syntactic dependency parsing. This allows us to assess the effects of various linguistic features on the tagging and parsing of Norwegian and identify the most morphosyntactically informative linguistic features in the treebank. As there is currently no readily available PoS tagger model or syntactic parser model trained on the treebank, we will train and evaluate a variety of state-of-the-art PoS taggers and dependency parsers on the treebank to identify tools and provide accompanying models that can be applied to running text in Norwegian. The results of our work are concrete contributions to the Norwegian NLP community: a data set split (training/development/test) of the Norwegian Dependency Treebank, a tailored PoS tag set optimized for syntactic dependency parsing, and clear recommendations regarding tools for PoS tagging and syntactic parsing of Norwegian.

1.1 Overview

Chapter 2 provides a brief introduction to the Norwegian Dependency Treebank and the tasks of PoS tagging and syntactic parsing together with their respective approaches, exemplified by concrete tools. It also provides an overview of previous work on PoS tagging for Norwegian and Swedish as well as the effects of tag set granularity on PoS tagging.

Chapter 3 describes the various parts-of-speech found in Norwegian and their respective morphological properties followed by how these are represented in the PoS tags and morphological features of the Norwegian Dependency Treebank. This is complemented by linguistic considerations for designing and modifying PoS tag sets as well as a qualitative comparison of the tag set of NDT and those of other comparable treebanks.

Chapter 4 outlines the experimental setup used in our experiments with linguistically motivated tag set modifications, which includes a proposed data set split of

NDT, our implemented tag set mapping and the choice of PoS tagger and syntactic parser for the development phase, and how the sum of these components are conjoined in a pipeline under which the tag set experiments are run.

Chapter 5 details our experiments with modifying the PoS tag set of NDT using linguistic insights and assesses the results of these experiments. For the most promising tag set modification for each category, we perform in-depth error analysis of tagging and parsing performance. Finally, we present our optimized tag set, obtained by combining the most promising tag set modification for each category.

Chapter 6 presents a comparative analysis of a wide array of state-of-the-art PoS taggers and dependency parses on the task of tagging and parsing the Norwegian Dependency Treebank using our optimized tag set. The best tagger and parser will then be employed on the held-out test data set using our optimized tag set, the results of which will be compared to the results obtained using the original tag set to contrast the tag sets and demonstrate the effects of our optimization.

Chapter 7 serves as a summary and conclusion of this thesis, also discussing future work related to this thesis.

Chapter 2

Background

To start things out, we briefly outline the newly developed Norwegian Dependency Treebank and provide an introduction to PoS tagging and syntactic parsing together with their respective approaches, exemplified through state-of-the-art tools. With this established, we will look at previous work related to this thesis, which includes PoS tagging for Norwegian and investigating the effects of tag set granularity on the performance of PoS taggers for English.

2.1 The Norwegian Dependency Treebank

The Norwegian Dependency Treebank (NDT) (Solberg et al., 2014) is the first publicly available treebank for Norwegian and serves as the basis for this thesis. It was developed at the National Library of Norway in collaboration with the University of Oslo, and contains manual syntactic and morphological annotation. The treebank contains 311 000 tokens of Bokmål and 303 000 tokens of Nynorsk, the morphological annotation of which follows that of the Oslo-Bergen Tagger (Hagen, Johannessen, & Nøklestad, 2000; Solberg, 2013), which in turn is largely based on the work of Faarlund, Lie, and Vannebo (1997). The annotated texts are mostly newspaper text, but also include government reports, parliament transcripts and excerpts from blogs. The annotation process of the treebank was supported by the Oslo-Bergen Tagger, described in Section 2.2.1, and then manually corrected by annotators.

2.2 Part-of-Speech Tagging

Part-of-speech tagging is concerned with assigning a single part-of-speech to each word in a sentence. Parts-of-speech, also known as PoS or word classes, are used to group words with similar grammatical properties, e.g. verbs, nouns and

adjectives. The words in a particular PoS often function similarly in terms of what they can occur nearby (i.e., syntactic properties) or the affixes they can take (i.e., morphological properties). Furthermore, parts-of-speech can be divided into two broad classes: closed class types, which have a relatively fixed membership and generally do not get assigned new words (e.g., pronouns and prepositions), and open class types, which have an open membership, continuously being assigned new words (e.g., nouns and verbs).

Different granularities are used in different tag sets, and while one can use as few as eight parts-of-speech for a very high-level approach, some systems operate with several hundred tags. The more is not necessarily the merrier, as while fine-grained tags provide more information, it becomes increasingly difficult to determine the correct tag with each of them being so fine-grained (Jurafsky & Martin, 2009, pp. 157–158).

To evaluate the performance of a PoS tagger, we compare its predictions on a held-out test set with a so-called 'gold standard', which is a manually annotated version of the test set. If the tagger agrees with the human annotator on the tag of a given word, the tagger has made a correct prediction. We iterate the entire data set and compare the predictions of the tagger with the human annotation in order to accumulate the number of correct predictions. The typical evaluation metric for measuring tagger performance is *accuracy*, which is a simple fraction of the number of correctly tagged tokens divided by the total number of tagged tokens.

Part-of-speech tagging plays an important role in many Natural Language Processing (NLP) applications as a pre-processing step, i.e., it lays the ground for further processing of the text on which the tagging was performed. Many parsers, dependency parsers in particular, require the input text to be PoS tagged. This is by no means a trivial task, even though it may seem fairly straightforward; simply looking the word up in the dictionary is not sufficient. The main challenge here, like in so much of computational linguistics, is ambiguity, which is a pervasive phenomenon in languages. One word can potentially belong to several parts-of-speech, and while most words in English are unambiguous, many of the most common words are in fact ambiguous (Jurafsky & Martin, 2009, p. 167). When faced with ambiguity, a tagger has to be able to disambiguate, i.e., resolve the ambiguity. This can be done in several ways, e.g., by looking at the surrounding words in the sentence forming the context, which can actually be very informative of the word and provide useful information as to what tag should be assigned to the word in question. For instance, if the preceding word is a determiner, chances are that the word in question is a common noun.

2.2.1 Approaches to PoS Tagging

PoS tagging is performed by PoS taggers, which can generally be divided into two groups: rule-based and data-driven taggers. Where rule-based taggers rely on handwritten rules in order to assign each word the correct tag, the data-driven (also known as stochastic) taggers determine the correct tag based on gathered statistics. While rule-based taggers can be directly applied to new text, the data-driven taggers need to be trained first in order to gather the statistics used in the tagging. One could argue that rule-based taggers are (indirectly) data-driven, as it is highly likely that existing rules are altered and/or new rules are added as more data is processed. However, these changes are carried out by trained linguists, as opposed to the machines used in data-driven approaches, which is based on the method of machine learning. In the following sections, I will go into detail about these two approaches, discussing their respective merits and how they differ as well as provide examples of PoS taggers employing them.

Data-driven PoS Taggers

Data-driven taggers are taggers that are trained on annotated data, with this training consisting of processing and analyzing the input data to recognize and generalize patterns that are used to find the most likely tag for a given word in a given context. This is done by applying machine learning techniques, which can learn from and subsequently make predictions on data. The trained models and the gathered statistics are then used in the tagging of new data. The training data sets are typically manually annotated corpora, i.e., large text collections in which the words are labeled with part-of-speech tags by human annotators. The most widely used corpus for English is the Penn Treebank (Marcus, Santorino, & Marcinkiewicz, 1993), which consists of 4.5 million words gathered from sources including the Wall Street Journal, IBM computer manuals and texts from the Library of America.

TnT One of the most well-known data-driven PoS taggers is TnT[1] (Brants, 2000), short for Trigram'n'Tags, which uses Hidden Markov Models (HMMs) combined with a smoothing technique and handling of unknown words. HMM-based taggers employ two kinds of probabilities to determine the correct tag: emission probabilities, which express the probability of a tag emitting a certain word, i.e., the probability of seeing a particular word given a particular tag, and transition probabilities, which express the probability of a particular tag given the preceding tag(s). As is standard in HMMs, the emission probabilities and transition probabilities are estimated using Maximum Likelihood Estimation (MLE),

[1] `http://www.coli.uni-saarland.de/~thorsten/tnt/`

which is based on relative frequencies encountered in the training. After this initial probability estimation, TnT smooths contextual and lexical frequencies and implements support for handling unknown words, which I will return to shortly.

Smoothing is a technique used to try to the solve the problem of sparse data, which is caused by the fact that the data in any corpus will be sparse in the sense that it can not include all acceptable, grammatical sentences or sequences of tags. The probability of a sequence containing an unknown word will be 0 if smoothing is not applied, as the probabilities are estimated using maximum likelihood estimation, and the maximum likelihood of a word that has not been seen will be 0, even if it is a perfectly fine word in a perfectly fine sentence (Jurafsky & Martin, 2009, pp. 131–132). Smoothing resolves this by reserving some of the probability mass to unseen words. A number of smoothing techniques can be employed to achieve this, but Brants (2000) argues that the smoothing paradigm that yields the best results in TnT is linear interpolation of unigrams, bigrams and trigrams.

The handling of unknown words in TnT is carried out by using suffix analysis. Tag probabilities are set according to the word's ending, as Brants (2000) argues that the suffix is a strong predictor for PoS classes. The probability distribution for a particular suffix is obtained by looking at all words in the training set that share the same suffix. Furthermore, Brants (2000) argues that using suffixes of infrequent words is a better approximation for unknown words than using suffixes of frequent words, and the suffix handling procedure is thus restricted to words appearing less than 10 times.

HunPos HunPos[2] (Halácsy, Kornai, & Oravecz, 2007) is an open-source PoS tagger based on TnT. Its main feature, departing slightly from TnT, is augmenting the parameters of the traditional emission probability estimation, additionally taking into account the preceding tag, thus incorporating more context into the standard HMM model. The authors of HunPos argue that this can lead to as much as 10% reduction in the error rate. HunPos reaches an accuracy of 96.58% on the Wall Street Journal data of the Penn Treebank (Marcus et al., 1993), while the original TnT obtains an accuracy of 96.46%. Halácsy et al. (2007) argue that the increase in the emission order clearly improves the performance of the tagger for English.

Stanford The Stanford tagger[3] (Toutanova, Klein, Manning, & Singer, 2003) is based on maximum entropy, also known as multiclass logistic regression, which generalizes logistic regression to multiclass problems. Their novel approach combines this with cyclic (bidirectional) dependency networks, where the probability

[2] `https://code.google.com/archive/p/hunpos/`

[3] `http://nlp.stanford.edu/software/tagger.shtml`

of a given node (word or tag) is conditioned on both the preceding and the following word(s) or tag(s), yielding bidirectional inference. Toutanova et al. (2003) make extended use of lexical features focusing on surrounding words, which they found to be very useful, and not incorporated into previous models. They report an accuracy of 97.24% on the Wall Street Journal data of the Penn Treebank, which at the time of publication was the best automatically learned PoS tagging result.

SVMTool SVMTool[4] (Giménez & Màrquez, 2004) is a data-driven PoS tagger that employs Support Vector Machines (SVMs) to perform the tagging. An SVM model is geometric rather than probabilistic, and represents the objects (corresponding to words in the case of PoS tagging) as points in a vector space. It is a max-margin classifier, meaning that it aims to make the decision boundary separating the classes as wide as possible. SVMTool is highly flexible and tunable, and offers a rich feature set which can be modified to fit different needs. These features include word features and PoS features for the three preceding and following words and tags, respectively, as well as both preceding and following bigrams and trigrams. They also take into account affixes and orthographic features, such as capitalization, hyphenation and similar information related to the word form. The tagger reached an accuracy of 97.16% on the Wall Street Journal data of the Penn Treebank.

Rule-based PoS Taggers

In contrast to data-driven taggers, rule-based taggers do not use probabilities to determine the correct tag, but instead have a large set of handwritten disambiguation rules for all the possible ambiguities they might encounter. If a rule-based tagger encounters an ambiguity to which there is no applicable rule, it is simply unable to perform the disambiguation. The tagging is performed in two stages: first, the word in question is looked up in the dictionary in order to obtain a list of potential parts-of-speech. This list is then (hopefully) narrowed down to a single part-of-speech by applying the relevant disambiguation rule(s) (Jurafsky & Martin, 2009, pp. 170–172).

Oslo-Bergen Tagger The Oslo-Bergen Tagger[5] (Hagen et al., 2000), henceforth OBT, is an example of a rule-based tagger. It was developed for Norwegian and uses a formalism called Constraint Grammar (CG). Taggers employing the CG approach applies a large set of linguist-written constraints to the input sentence to

[4] `http://www.cs.upc.edu/~nlp/SVMTool/`
[5] `http://www.tekstlab.uio.no/obt-ny/english/`

rule out incorrect parts-of-speech, with each rule either adding, removing, selecting or replacing a tag or a set of a tags.

The authors behind OBT chose to develop a large number of detailed constraints for each possible correct reading. When facing ambiguous readings, it finds all possible readings for each word, and then applies the appropriate constraint in order to disambiguate and find the correct reading. However, the constraints do not necessarily produce one single reading per token. As the OBT only concentrates on grammatical ambiguity, and does not look at ambiguity between lemmas, it may output several readings, thus not fully disambiguating. This, combined with the fact that continued rule writing gave diminishing returns, motivated the addition of a statistical module.

This resulted in OBT+Stat (Johannessen, Hagen, Nøklestad, & Lynum, 2011), a statistical module that resolves all ambiguities left in the OBT output, including the grammatical ambiguities and the lemma ambiguities originally left on purpose as well as the unfortunate ambiguities. This statistical module uses the aforementioned HunPos tagger to perform the disambiguation, which is run independently of the CG-based disambiguation when the CG tagger leaves more than one reading per token, in which case the results from the two taggers are unified. If the two taggers agree on a reading, that reading is selected. Otherwise, an attempt is made to disambiguate using the lemma. This lemma disambiguation scheme works by using a word frequency list derived from NoWaC (Guevara, 2010), a corpus of Norwegian documents published on the Internet. It selects the lemma with the highest frequency in the corpus, based on the assumption that most lemmas will appear as words in a large corpus as Norwegian lemmas correspond to uninflected word forms. However, if the lemma disambiguation fails, an arbitrary reading out of the possible readings is selected.

After the CG-driven disambiguation had been performed, 8.6% ambiguous tokens were still present, which were all resolved by the statistical module. The OBT+Stat tagger achieved an overall accuracy of 96.56%, which Johannessen et al. (2011) consider to be a good result, considering that they removed all remaining ambiguity and at the same time kept a large, detailed tag set and disambiguated lemmas with identical tags.

2.3 Syntactic Parsing

Syntactic parsing is concerned with analyzing sentence structure, one aspect of which is assigning syntactic functions to words in a sentence, e.g., finding the subject and object in a sentence. Syntactic parsers typically require the input sentence to be labeled with parts-of-speech, as these provide important linguistic information about the words in a sentence and are often used as features in the

Head	Dependent
Preposition	Prepositional complement
Finite verb	Complementizer
First conjunct	Subsequent conjuncts
Finite auxiliary	Lexical/main verb
Noun	Determiner

Table 2.1: Annotation choices in NDT, taken from Solberg et al. (2014).

parser. Just as for PoS tagging, ambiguity is the main obstacle in parsing. Most sentences have more than one possible structure, hence the parser must be able to determine the correct analysis out of a potentially very large set of possible analyses. Syntactic parsing serves as an important step in a range of NLP tasks where the structural analysis of sentences is crucial.

There exists a wealth of linguistic theories to represent syntactic structures, but as Norwegian Dependency Treebank is based on dependency grammar, we will focus on dependency grammar.

2.3.1 Dependency Grammar

Dependency grammar assumes a binary, asymmetrical relation between words in a sentence, so-called *dependencies*, where each node has a single head, of which it is the dependent. These dependencies are represented as arcs from a head to its dependent, where the label on the arc denotes the name of the dependency relation, e.g., SUBJ (subject) or ADV (adverbial). The head is regarded as the superior node in the relation in that it is more prominent and in some way governs its dependent, 'allowing' for its presence (Nivre, 2005).

The syntactic annotation choices in NDT are largely based on the Norwegian Reference Grammar (Faarlund et al., 1997). The annotation choices are outlined in Table 2.1, taken from Solberg et al. (2014), providing overview of the analyses of syntactic constructions that often distinguish dependency treebanks, such as coordination and the treatment of auxiliary and main verbs. See an example of a dependency graph taken from NDT in Figure 2.1.

2.3.2 Data-Driven Dependency Parsing

Dependency parsing is the process of parsing a sentence and assigning it syntactic structure represented in terms of dependency grammar. Similarly to the

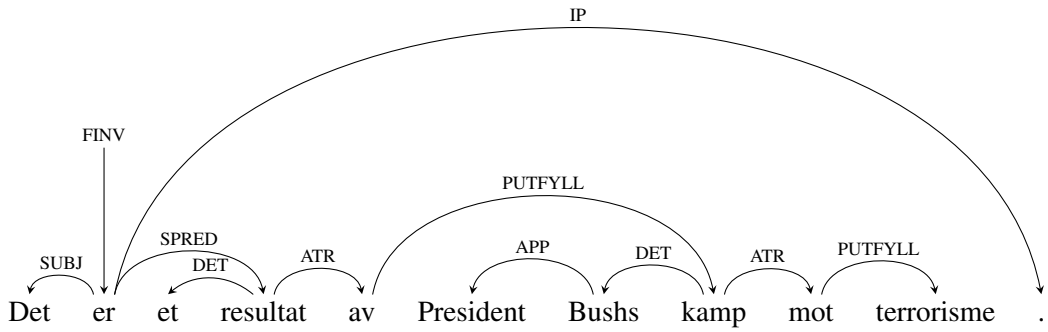

Figure 2.1: Example of a dependency graph representation of a sentence taken from NDT.

approaches to PoS tagging, the most successful dependency parsers today employ statistical modeling and machine learning, resulting in data-driven dependency parsers. Given an input sentence, we want to derive the correct, i.e., the most probable, dependency graph out of a set of candidate graphs.

Data-driven dependency parsing can be broadly divided into two main approaches: graph-based models and transition-based models (McDonald & Nivre, 2011). Parsers employing the *graph-based* approach parameterize models over dependency subgraphs and learn these parameters to score an entire dependency graph for a given sentence in order to find and locate the highest-scoring, optimal parse from a set of candidate parses. They are hence globally trained and often employ exhaustive search to find and locate the most probable parse for a given sentence. *Transition-based* approaches, on the other hand, parameterize models over transitions between states in an abstract state machine and use local training and greedy search to find the best transition (the locally optimal choice) from any given state.

Data-driven dependency parsing has seen a surge in recent years, with several shared tasks dedicated to advance the state of the art of dependency parsers. We will now turn to four dependency parsers which will be trained and evaluated on NDT in later experiments in Chapter 6.

Malt MaltParser[6] (Nivre et al., 2007) is a transition-based dependency parser based on a deterministic parsing strategy along with treebank-induced classifiers

[6] `http://maltparser.org/`

for predicting parser actions, i.e., transitions. It uses history-based feature models, which involves using previous transitions as a means to guide the parsing. It comes with a variety of parsing algorithms, feature models and machine learners to choose from, the optimal of which can be found by employing MaltOptimizer (Ballesteros & Nivre, 2012), used to optimize the parameters of the parser.

Mate The Mate parser[7] (Bohnet, 2010) is graph-based and uses maximum spanning trees and third-order features to find the best dependency parse. The maximum spanning tree of a sentence is the highest scoring tree spanning the whole sentence, where the score of a dependency tree is the sum of the scores of its dependencies. Third order features involve the parser being able to evaluate substructures containing three dependencies. It has been shown to perform very well for English (Choi, Tetreault, & Stent, 2015), which Bohnet (2010) attributes mainly to the use of the Hash Kernel.

RBG The RBG parser[8] (Lei, Xin, Zhang, Barzilay, & Jaakkola, 2014) is a graph-based dependency parser using tensor decomposition and a low-rank factorization method to map high-dimensional feature vectors to low-dimensional representations, as described by (Lei et al., 2014). It can be run with either first-order or third-order features, where the first-order feature model is the fastest but least accurate.

Turbo The Turbo parser[9] (Martins, Almeida, & Smith, 2013) is graph-based and employs dual decomposition and specialized head automata to handle third-order features in producing non-projective dependency graphs, i.e., graphs with crossing arcs. Dual decomposition splits the original problem into local subproblems and seek an agreement on the overlapping variables.

2.4 Previous Work

2.4.1 PoS Tagging for Norwegian

Marco (2014) developed an open-source part-of-speech tagger for Norwegian based on an existing processing library, motivated by the scarcity of PoS taggers for Norwegian, with the author citing the Oslo-Bergen Tagger (Hagen et al., 2000) as the only available tool for this purpose. The tagger is mostly based on

[7] https://code.google.com/archive/p/mate-tools/
[8] https://github.com/taolei87/RBGParser
[9] http://www.cs.cmu.edu/~ark/TurboParser/

statistics and makes use of a simple and standard tag set, hence clearly departs from the rule-based and tag-comprehensive approach seen in OBT.

Methodology and Data The part-of-speech tagger was created using the FreeLing tool, an open-source text processing tool offering a number of language analysis services, with its part-of-speech tagger being the focus of this paper. The tagger uses an adaptation of *Norsk ordbank* as its dictionary, which is a large database of lexical units with morphosyntactic and argument structure information. It was trained and evaluated on a pre-release version of NDT using the FreeLing-included HMM tagger, which is a trigram tagger based on TnT. The tag set in both the dictionary and the gold standard corpus was simplified and standardized, with the simplification consisting of combining certain original tags into a single tag and omitting some of the arguably less important grammatical information. The adapted tag set is mostly based on the EAGLES standard, where the first letter of each tag indicates the part-of-speech of the word and the remaining letters specify more fine-grained morphosyntactic and semantic information.

FreeLing analyses forms not found in the dictionary, i.e., unknown words, by use of a compound module and an affixation module that checks whether the word is a compound or a derived form, respectively. The compound module works by simply concatenating words found in the dictionary to check whether the word in question is a compound of existing words, and the affixation module detects whether a word is a derived form, based on a list of affixes applied to words in the dictionary. Additional modules customized for the purpose of this tagger include the tokenizer and the multiword expression module.

Error Analysis Most errors seen in the tagging were due to ambiguities in the dictionary. These include ambiguities in morphological features, such as nominative versus accusative case ambiguity in pronouns, number ambiguity in nouns and gender ambiguity in adjectives. Words not included in the dictionary, such as many proper nouns, acronyms and compound words, also constitute a large portion of the errors.

Evaluation The accuracy of the tagger was evaluated using five-fold cross validation over the gold standard . Three types of morphological information with increasing degree of detail were evaluated: (i) main part-of-speech (PoS-1); (ii) main part-of-speech and information about the subtype of the part-of-speech, e.g., auxiliary versus main verb (PoS-2); and (iii) detailed morphosyntactic information given by the tag, e.g., gender, number and case (PoS-3). The tagger's performance was evaluated as quite close to that of state-of-the-art taggers for English (between 96% and 98%) reaching an accuracy of 97.3%, 96.2% and 92.4% for

PoS-1, PoS-2 and PoS-3, respectively. It yielded a higher accuracy in morphosyntactic tagging than the OBT+Stat tagger, but as pointed out by Marco (2014), this is not necessarily a valid comparison, considering that the two taggers use different tag sets and data sets. It is worth noting that the performance of taggers are not necessarily directly comparable; unless the taggers are trained and tested on the same premises and resources, a direct comparison is rather futile. Moreover, even if one were to evaluate them in a near-identical manner, the results wouldn't necessarily be all that revealing, as a tagger developed and trained for tagging a specific type of text performing poorly at tagging a completely different type of text does not provide useful information about the tagger. We want to find out how the tagger performs on the data to which we want to apply it. With this in mind, I will turn to a paper investigating how taggers can be evaluated and compared in a systematic manner.

2.4.2 Comparing PoS Taggers for Swedish

Megyesi (2001) argues that the manner in which PoS taggers are evaluated by researchers differ greatly, making it difficult to compare the performance of the systems. Motivated by this, Megyesi set out to compare four data-driven learning algorithms on the task of PoS tagging of Swedish, and to investigate what types of errors they made as well as the effects of tag set size and training set size. The four algorithms in question are Hidden Markov Model (HMM), Maximum Entropy (MaxEnt or ME), Memory-Based Learning (MBL) and Transformation-Based Learning (TBL). These taggers had all previously been tested for English with an average accuracy between 95% and 97%, and were used with their default settings as the main goal was to evaluate the systems out-of-the-box. The experiments were run on the second version of the Stockholm-Umeå Corpus (SUC) (Gustafson-Capková & Hartmann, 2006), which was annotated with a Swedish version of PAROLE tags, with the tag set totaling 139 tags. The corpus was randomly divided into ten roughly equal parts to get subsets containing different parts.

Taggers Memory-based learning is a case-based approach where new items are classified based on their similarities to the items already stored in memory during learning (Daelemans, Zavrel, Berck, & Gillis, 1996). The maximum-entropy framework called MXPOST is a probabilistic classification-based approach based on a maximum-entropy model where contextual information is represented as binary features that are used simultaneously in order to predict the PoS tag. It uses beam search to find the most probable sequence of tags, and then chooses the tag sequence with the highest probability (Ratnaparkhi, 1996). Transformation-based

learning is a rule-based approach that learns by detecting errors, and generates new rules based on the observed errors and the subsequent resolution of these errors (Brill, 1994). TnT is used as presented in section 2.2.1.

The evaluated taggers have in common their approach to representing data, which is to generate a feature vector for each word in the corpus, consisting of the words they appear with and their respective tags, i.e., the context. However, these taggers differ with respect to what features they store in the vectors. MBL takes into account the current word, the preceding and following word forms, the two preceding tags and the immediately following tag; ME includes the current word, the following and preceding two words and the preceding two tags; TBL uses a context of three preceding and following words and/or tags; and finally, TnT is trigram-based and works as previously described.

Evaluation PoS taggers can be evaluated in a number of ways, and Megyesi (2001) decided to evaluate the taggers from three different aspects to see how they compare: (i) the accuracy of each classifier was evaluated by using the entire tag set and 10% of the SUC as training data; (ii) the original tag set was divided into smaller subsets of varying size to investigate the effects of tag set size (See Section 2.4.3); (iii) the size of the training corpus was varied from one thousand up to one million tokens for each tagger to see how the various sizes affect the error rates.

To ensure a fair comparison, each tagger was trained on the same portion of SUC for each of the experiments, and the resulting classifiers were then evaluated on the same separate test set. The taggers were only allowed to assign a single tag (i.e., not a set of possible tags) to each token in the test data set.

TnT had the highest overall accuracy of 93.55%, and also succeeds best in the annotation of known and unknown words, with the MXPOST tagger coming in second at 91.20%. The memory-based learner (89.28%) performs slightly better than the transformation-based learner (89.06%), which performs poorly at tagging unknown words, but manages to disambiguate known words quite well. The different taggers make different types of errors; whereas TBL and MB more often make mistakes in the morphological analysis of categories, TnT and ME more frequently confuse PoS categories for ambiguous words.

Each tagger was trained ten times on the same data of various sizes, including 1 000, 10 000, 100 000 and 1 millions tokens. The corresponding test set was subsequently annotated by each of the resulting classifiers. The error rate decreases as the size of the training data increases, unsurprisingly, as the ratio of unknown words to known words is much lower in larger data sets. The impact of the size of the training data on the taggers differ greatly; whereas ME was highly sensitive to the data set and shows a great decrease in error rate (88%), TBL was

less sensitive, and the error rate decreased by 50% when going from 5 000 to 500 000 tokens. TnT outperforms the other taggers on the task of annotating unknown words when the training set consists of 20 000 or more tokens, and when there are less than 20 000 tokens, TBL has the lowest error rate. TnT consistently outperforms all the other taggers in annotating known words, showing an error rate of only 3.5% when the training set consists of one millions tokens. It is also worth taking into consideration the run times of the different taggers. Whereas TnT can learn from 100 000 tokens in one second, and manages to tag a text containing the same amount of data in three seconds, ME and TBL spends roughly a day training on 100 000 tokens.

HunPos In a round of follow-up experiments, Megyesi (2009) evaluated the aforementioned HunPos tagger on the task of PoS tagging for Swedish. The tagger was trained on SUC, and in a similar fashion to the experiments carried out in Megyesi (2001), trained on different subsets of the corpus, varying in size from 1000 tokens to 1 million tokens. The separate test set contained 117 685 tokens in 7 464 sentences. The annotation was done using the PAROLE tag set, which at the time of writing contained 156 tags. Various feature settings were used in training the tagger to investigate what settings serve as the most appropriate for Swedish, among them the order of tag transitions, i.e., the number of preceding tags included in the estimation of transition probabilities, using either bigram tagging or the default trigram tagging. Different estimations of emission probabilities were also tested, where the probability of a word is conditioned only on the tag of the word itself or both the current tag and the immediately preceding tag, with the former being used in TnT and the latter being the default in HunPos. Megyesi (2009) found that bigram models perform better and are more appropriate when the training data contains less than 50 000 tokens, while trigram models are to be preferred when the training data contains more than 50 000 tokens. Furthermore, the size of the suffixes was varied, experimenting with setting the size to 5 and 9 in addition to the default value of 10 to investigate whether a decrease in suffix length can increase performance. Megyesi (2009) argues that the results prove this not to be the case, and that the larger suffix window is always to be preferred for Swedish. She finally experimented with varying the maximum frequency with which a word can occur in order to be added to the suffix tree and used in the suffix analysis, either 5, 9 or the default value of 10. For training data containing less than 100 000 tokens, reducing the frequency requirement for words to be added to the suffix tree can lead to an improvement in tagger performance. For larger training data sets, however, the experiments by Megyesi (2009) suggest that the default value of 10 can be used, as there are no significant improvements in decreasing the frequency requirement.

HunPos was then compared to the taggers evaluated in Megyesi (2001), namely MBT, MXPOST, TBL and TnT. These were all run with default settings, while HunPos was tested both with default settings and optimal settings, where the optimization involved choosing the feature settings that led to the highest accuracy. Consistent with the results presented in Megyesi (2001), the HMM-based taggers performed best, with HunPos and TnT performing closely to one another on all training data sizes. HunPos had the highest accuracy when the training data contained less than 20 000 tokens, while TnT performed best on the remaining data sets, except for when the training data set consisted of 1 million tokens, where HunPos was marginally better than its predecessor, showing an accuracy of 95.90% compared to TnT's 95.89%.

2.4.3 The Effects of Tag Sets

Part-of-speech tag sets are often taken for granted, and many NLP applications for English use the tag set supplied by the de facto standard Penn Treebank (Marcus et al., 1993), totaling 45 tags, in an unaltered form. It is therefore interesting to investigate whether more fine-grained tag sets can lead to improvements in tagger performance, as they certainly increase the linguistic utility, emitting more linguistic information than their more coarse-grained counterparts. It is known, and rather apparent, that more complex tag sets may complicate the tagging and lead to drops in tagger accuracy, as the likelihood of ambiguity is higher.

Mapping Fine-Grained Tag Sets to Coarse-Grained Tag Sets In addition to the complete tag set of 139 tags in the Stockholm-Umeå Corpus (Gustafson-Capková & Hartmann, 2006), Megyesi (2001) trained the taggers on four subsets of these tags, consisting of 26, 39, 44 and 48 tags, respectively. In Megyesi (2002), the tag sets are slightly altered, consisting of 25, 39, 43 and 48 tags, respectively. These experiments were carried out because a tag set with complete morphological tags is not necessarily needed for all NLP applications (Megyesi, 2002). For applications where a complete fine-grained tag set is excessive, one can instead reduce the size of the tag set, which in turn leads to a decrease in error rate as the tagger has an easier task determining the correct tag. A subset of tags with some more general properties, perhaps only the main PoS tags, may suffice. Due to this, Megyesi (2002) chose to map the original tag set into smaller tag sets in the training data to investigate how this affects the performance, with the goal being to develop tag sets that can be useful for various applications. The smaller subsets differ with respect to the type of morphological features included within each PoS category. The smallest subset totaling 26 tags consists of the main PoS tags as well as some subcategorization information, while the larger subsets of

39, 44 and 48, respectively, are extensions of this. For the tag set containing 39 tags, the extension involved including distinctions between present and perfect participle and between verbs in various tenses. This tag set was then used as basis for further distinctions for the tag sets of 44 and 48 tags, leading to increasingly fine-grained tag sets.

The number of errors made by each tagger is higher when using a large tag set, which is not surprising, considering that it is more challenging to determine the correct tag when the set of possible tags is larger. TnT consistently performs best, again with ME coming in second, MB third and TBL last. TnT and MB seem to be less sensitive to the size of the tag set than ME and TBL, as the changes in error rate are smaller. Megyesi (2002) experimented with using tag sets of different sizes in training and testing, using the complete tag set of 139 tags in the training and the smallest subset of 25 tags in the testing. This lead to a decrease in error rate for TnT and MB, while causing more errors for ME and TBL, compared to when both training and testing were done on the smallest tag set.

Mapping Coarse-Grained Tag Sets to Fine-Grained Tag Sets While Megyesi (2001) mapped the original tag set into smaller subsets, MacKinlay (2005) mapped a coarse-grained tag set into more fine-grained subsets, thus augmenting the tag set. The approaches do, however, share a common goal, which is to investigate the effects of tag sets on tagger performance. MacKinlay used the Penn Treebank and its accompanying tag set, totaling 45 tags, as basis for the experiments, from which new, more fine-grained tag sets were mapped using the Natural Language Toolkit (NLTK) in Python. The algorithms used for carrying out the experiments were Transformation-Based Learning (TBL), Support Vector Machines (SVM) and Maximum Entropy (MaxEnt or ME).

The primary goal of MacKinlay (2005) was to improve tagger performance using linguistic insight. The tags were subdivided in a linguistically sensible way into finer-grained tag sets, which yield additional information that may be used to assist the tagger in making syntactic generalisations which are not apparent either from the coarse PoS tags or from the sparsely populated lexical feature vector. Finer-grained tag sets increase its linguistic utility as more linguistic information is present. However, complex tag sets may in turn lead to increased difficulty in determining the correct tag due to a potentially increased number of possible tags for a word, between which the tagger has to disambiguate.

MacKinlay (2005) attempted to resolve the problem of complex tag sets leading to an increase in error rate by mapping the new tags back to the original tags before evaluation, hence using the mapped tags only internally as a means to increase the tagger performance, not to be used in the final evaluation. The modifications adhered to certain restrictions for this inverse mapping to be carried out

sufficiently: there can be no discrepancies between the original and the inversely mapped tags, i.e., the original tags had to be unambiguously recoverable from the new tags.

Motivation It may seem rather pointless to strive for a small performance improvement, e.g., increasing the accuracy from 97.01% to 97.05%. Sentence-level accuracy, however, increases much more radically even when there is only a small increase in token-level accuracy. MacKinlay (2005) argues that a tagger which yields 97% accuracy for tokens, achieves 49% at the sentence level, while a tagger performing at 98% yields an accuracy of 62% for entire sentences. This indicates that it certainly is worth striving for a slight increase in word token accuracy, as it leads to a substantially higher performance over sentences.

Experimental Evaluation The linguistically motivated modifications, i.e., tag mappings, were conditioned on both lexical features and syntactic features. It is interesting to investigate modifications of both types as they clearly differ in how they use linguistic insight.

The syntactically conditioned modifications involved looking at the head of the phrase or the preceding or following words to determine the correct tag. One of these modifications involved introducing a new, separate class for subordinate conjunctions, which are originally grouped with prepositions in the *IN* class, as well as mapping the prepositional uses of *to* into *IN*, while being left as *TO* when used as an infinitive marker. Other modifications include attempting to resolve the difficulty of distinguishing adjectives from verb participles, as they can be hard or even impossible even for humans to disambiguate when extended knowledge of the context is not provided. However, MacKinlay (2005) argues that the word in question can unambiguously be classified as an adjective when paired with a degree adverb, as verb participles cannot be modified by an adverb.

The tag set modifications conditioned on lexical features include attempts to resolve the ambiguity between *IN* and *RP*, as prepositions function as predicate complement in verb particles. There are words in the *IN* class that do not have a corresponding homograph in *RP*, meaning that they cannot be the predicate complement in a verb particle, hence can be unambiguously tagged as *IN*. They therefore mapped the ambiguous members of *IN* to a separate class as a pre-processing step to distinguish them from those which are unambiguous. Other lexically conditioned modifications include mapping determiners and articles into classes based on how they indicate the number of the noun phrase they precede, i.e., if the noun is singular (e.g., *a*) or plural (e.g., *some*), or if the number simply cannot be inferred from the determiner (e.g., *the*).

In addition to the primary approach, MacKinlay (2005) pursued an alterna-

tive, more data-driven approach by using machine learning to determine clusters within the tag sets corresponding to patterns of syntactic regularities. A range of syntactic and collocational features were introduced to assist in determining these patterns. The features for each <*word type,tag*> pair are derived from syntactic and collocational patterns seen in the training. For instance, one feature was *Par_is_VBP*, which simply states the probability of the parent of the word being *VBP*. The values of these features are probability distributions indicating their relative frequency. The features and their values were then used to generate clusters, which assist the tagger in assigning words the correct tag.

The results of these experiments were not of the promising kind, as the introduction of new classes and more fine-grained tag sets rarely led to an improvement in performance, in most cases in fact having a negative impact on the accuracy. Even in the cases where they saw an improvement in performance, the improvements were for the most part marginal at best. The most promising modifications were generally those resulting from the clustering approach rather than the linguistic modifications. They argue that it seems like the linguistic modifications are less data-dependent while the data-driven modifications have a slight tendency to overfit. The modifications which had no impact on the performance, i.e., that neither led to an increase nor a decrease in accuracy, may still be used in other NLP applications where the increased linguistic utility by itself is useful.

Chapter 3

Parts-of-Speech & Tag Sets

This chapter provides an overview of the parts-of-speech in Norwegian and the tag set of the Norwegian Dependency Treebank, complemented by linguistic and computational considerations for designing and modifying tag sets. We will discuss how the tag set of the Norwegian Dependency Treebank is designed with regard to parts-of-speech in Norwegian and present a qualitative comparison of the tag set of NDT and tag sets of other comparable treebanks to see how they relate.

3.1 Considerations for Designing Tag Sets

We are looking to optimize the existing tag set of NDT for dependency parsing of Norwegian, thus identify a tag set that yields high parser performance while also providing us with useful linguistic information. As tagging is an important preprocessing step for parsing, the performance of the tagger will likely have great impact on the parser performance.

The granularity of a tag set is directly related to its size; a coarse-grained tag set consists of a relatively small number of tags, and as tag sets increase in size, the more fine-grained they become. Smaller, more coarse-grained tag sets tend to lead to higher tagger accuracy because it is easier to infer the correct tag when the number of possible choices is smaller. Indeed, simply opting for a tag set consisting of a single tag would lead to 100% accuracy. This would, however, be entirely useless for NLP tasks, as we want to find a tag set that also models the relevant morphological and syntactic information.

Leech (1997) noted the conflict between linguistic and computational considerations one often encounters when designing tag sets. The linguistic quality of a tag set is determined by the extent to which it represents all important grammatical information in the language, while the computational tractability of a tag set is

determined by how easy it is to infer the correct tag for a particular word and how useful a particular tag is in aiding the disambiguation process (Leech, 1997).

We are looking for tag sets that are of high linguistic quality and are highly computationally tractable. Some more fine-grained distinctions may actually improve the tagger performance, as the extra information available may assist the tagger in disambiguating ambiguous and unknown words. However, tag sets of very high linguistic quality are likely to lead to diminishing returns and compromise the computational tractability, as the tagger is unable to properly learn very fine-grained distinctions for which there is not sufficient training data. Hence, there is often a trade-off between the two aspects, and we want to find the 'golden mean' between coarse-grained and fine-grained tag sets.

The best tagging does not necessarily lead to the best parse, and a tag set that is optimized in terms of tagger performance alone is not necessarily optimized for parsing. We will therefore modify the tag set of NDT in various ways to see how the tag set granularity affects the tagging and, more importantly in our case, the syntactic parsing in order to a identify a tag set tailored for dependency parsing of Norwegian.

3.2 Existing Tag Sets

The tag set of a given treebank is the set of tags that can be assigned to tokens in the treebank. The number of tags and thus the granularity varies greatly between treebanks, and in order to limit ourselves to a small number of tag sets to which we will compare the tag set of NDT, we chose Penn Treebank (Marcus et al., 1993), Stockholm-Umeå Corpus (Gustafson-Capková & Hartmann, 2006) and Universal Dependencies (Nivre, 2015). It is worth discussing what grammatical information is represented in the tags of the various tag sets and the authors' rationale when developing the tag set.

Penn Treebank was chosen due to its status as the de facto standard treebank for English. It is widely used in NLP and has also served as inspiration for other treebanks. Stockholm-Umeå Corpus was chosen as Swedish is closely related to Norwegian and thus very similar and comparable in terms of morphology and syntax. Finally, we compare the tag set of NDT to the tag set of Universal Dependencies because it is an attempt to define a cross-lingual tag set.

3.2.1 Penn Treebank

Penn Treebank (Marcus et al., 1993) is the de facto standard treebank for English. It was developed at the University of Pennsylvania from 1989 to 1996, and its tag set is based on that of the Brown corpus (Francis & Kučera, 1979). The Brown

corpus tag set comprises 87 simple tags, but additionally allows the formation of compound tags, giving a total of 187 tags. In order to reduce the size of the tag set, Marcus et al. (1993) set out to eliminate the redundancy present in the Brown corpus tag set by taking into account both lexical and syntactic information in the development of tags. The final morphosyntactic tag set of Penn Treebank consists of 36 PoS tags, disregarding tags for punctuation, symbols, etc..

Penn Treebank introduces more fine-grained distinctions mainly for nouns and verbs, which are inflected and agree with words in their context. The tag set consists of fairly fine-grained tags for nouns (four tags), verbs (six tags) and wh-words (four tags), as well as separate tags for adjectives and adverbs of comparative and/or superlative degree. Apart from these, the categories are mostly rather broad, such as `DT` (determiner).

State-of-the-art PoS taggers trained and evaluated on Penn Treebank report an accuracy between 97% and 98%[1].

3.2.2 Stockholm-Umeå Corpus

Stockholm-Umeå Corpus (Gustafson-Capková & Hartmann, 2006) is a treebank for Swedish, developed in collaboration between Stockholm University and Umeå University in the 1990s. It was the first generally accessible, annotated corpus of the Swedish language. Its tag set is based on the tag set of SWETWOL (Karlsson, 1992) and extensively discussed in Ejerhed, Källgren, Wennstedt, and Åström (1992). The tag set of SUC comprises 22 morphosyntactic tags and is thus quite coarse-grained. It consists of mostly very general categories, such as `JJ` (adjective), `NN` (noun) and `VB` (verb).

Östling (2013) presented Stagger, an open source part-of-speech tagger for Swedish, which reached an accuracy of 96.58% on the second version of SUC. In comparison, TnT (Brants, 2000) obtained an accuracy of 95.9%.

3.2.3 Universal Dependencies

As parts-of-speech vary greatly between languages, properly comparing the performance of taggers across languages is often impossible, as the tag sets are highly dissimilar and therefore incomparable. In order to try to alleviate the issues caused by these differences, Petrov, Das, and McDonald (2012) set out to develop a universal tag set, universal in the sense that it is cross-lingual and applicable to a range of languages. This is based on the idea that there exists a set of coarse syntactic

[1] `http://aclweb.org/aclwiki/index.php?title=POS_Tagging_(State_of_the_art)`

PoS categories in similar forms across languages, so-called universals. They defined a tag set consisting of twelve universal PoS categories: NOUN (noun), VERB (verb), ADJ (adjective), ADV (adverb), PRON (pronoun), DET (determiners), ADP (adpositions, i.e., prepositions and postpositions), NUM (numeral), CONJ (conjunction), PRT (particle), '.' (punctuation mark) and X (a catch-all for categories such as abbreviations and foreign words). These categories are based on what Petrov et al. (2012) consider to be the most frequent parts-of-speech that exist in most languages. Moreover, these categories are what they expect to be the most useful for users of PoS taggers.

To evaluate the universal tag set, they trained a supervised tagger based on a trigram Markov model (Brants, 2000) on the treebanks of 22 different languages. For each of these treebanks, they created a mapping from the original treebank tag set to the universal PoS tags, based on annotation guidelines and PoS tag definitions. They experimented with three combinations of training and testing: training and testing on the original tag set, training and testing on the universal tag set, and finally, training on the original tag set and testing on the universal tag set by mapping the predictions to the universal tag sets. Accuracies for each of these experiments were presented, and in all cases, training on the original tag set and testing on the universal tag set resulted in the highest accuracy.

Nivre (2015) revised and extended the universal tag set with five additional tags, namely INTJ (interjection), PROPN (proper noun), AUX (auxiliary verb), SCONJ (subordinate conjunction) and SYM (symbol), resulting in a total of 17 tags. Additionally, PRT was renamed PART and '.' was renamed PUNCT. This was done as part of the Universal Dependencies (UD) project[2], which is developing cross-linguistically consistent treebank annotation for a wide array of languages.

3.3 Parts-of-Speech in the Norwegian Dependency Treebank

The Norwegian Dependency Treebank operates with 12 morphosyntactic PoS tags, 7 of which additionally have features for more fine-grained morphological information, such as gender, number, definiteness and tense. We will take advantage of these features in the development of new tag sets, where we will create new tags which are concatenations of a coarse tag and one or more of these features. This allows us to assess what morphological information is useful in tagging and parsing of Norwegian.

As the tag set of the NDT borrows heavily from the Oslo-Bergen Tagger, which in turn is based on Faarlund et al. (1997), we mainly rely on Faarlund

[2] http://universaldependencies.org

et al. (1997) in discussing the parts-of-speech and their respective properties. All examples are taken from the Norwegian Dependency Treebank, unless otherwise noted. Feature values in brackets, e.g., `<adv>` (used for adjectives which may function as adverbials), give additional information about the token that might be relevant, but does not concern the morphological form directly (Kinn, Solberg, & Eriksen, 2013).

In the following, we will look at the parts-of-speech of Norwegian, how they are represented and used in the Norwegian Dependency Treebank, and compare the tag set of NDT to three other arguably comparable tag sets, namely those of the aforementioned Penn Treebank (Marcus et al., 1993), Stockholm-Umeå Corpus (Gustafson-Capková & Hartmann, 2006) and Universal Dependencies (Nivre, 2015), outlined in Section 3.2. See Table 3.1 for complete overview of their tag sets, compared to the tag set of NDT.

3.3.1 Nouns

Nouns are words that denote entities and objects, either real or abstract. In Norwegian, nouns have gender (masculine, feminine or neuter) and are inflected for number (singular or plural) and definiteness (definite or indefinite). For instance, a masculine noun such as *bil* 'car' have four different forms: indefinite singular *bil*, definite singular *bilen*, indefinite plural *biler* and definite plural *bilene*. Introducing rather fine-grained tags for nouns may prove useful due to the nature of nouns, as they have modifiers with which they agree (in gender, number and definiteness). Example (3.1) shows this agreement, where the form of the determiner and adjective is dependent on the properties of the noun they modify.

(3.1) *Det* *var* *et* *kraftfullt* *svar*
It was a.NEUT powerful.NEUT answer.NEUT
'It was a powerful answer'

In NDT, nouns can also have three different types: `appell` 'common noun', `prop` 'proper noun' or `fork` 'abbreviation'. Nouns in genitive case, e.g., *Karis* 'Kari's', are tagged as noun and marked with the feature `gen`. This is quite different from the Penn Treebank, where genitives have the possessive ending split off and tagged separately as a possessive ending (`POS`).

Penn Treebank operates with four PoS tags for nouns, e.g., `NN` for mass or singular nouns and `NNP` for singular proper nouns. SUC distinguishes between common and proper nouns, `NN` and `PM`, respectively. Similarly, the UD tag set has `NOUN` for common nouns and `PROPN` for proper nouns. As they all distinguish between proper and common nouns, investigating how the noun type may affect tagging and parsing of Norwegian is of interest.

Description	NDT	PTB	SUC	UD
Adjective	adj	JJ JJR JJS	JJ PC	ADJ
Adverb	adv	RB RBR RB$ WRB	AB HA	ADV
Determiner	det	DT PDT PRP$	DT PS	DET NUM
Infinitive marker	inf-merke	TO	IE	PART
Interjection	interj	UH	IN	INTJ
Conjunction	konj	CC	KN	CONJ
Preposition	prep	IN	PP	ADP ADV
Pronoun	pron	PRP PRP$ WP WP$	PN PS HP HS	PRON
Subordinate conjunction	sbu	IN	SN	SCONJ
Noun	subst	NN NNS NNP NNPS	NN PM	NOUN PROPN
Unknown	ukjent	FW	UO	X
Verb	verb	VB VBD VBG VBN VBP VBZ MD	VB	VERB AUX

Table 3.1: Qualitative comparison of the tag sets of NDT, PTB, SUC and UD.

3.3.2 Verbs

Verbs are words that convey actions, e.g., *spille* 'play' and *lese* 'read'. Verbs are inflected for tense in Norwegian, and can also have the imperative mood, indicating a command. For instance, the verb *skrive* 'write' has five different forms: imperative *skriv*, infinitive *skrive*, present *skriver*, preterite *skrev* and past perfect *(har) skrevet*. We furthermore see passive voice marked either through inflection (See Example (3.2)) or periphrastically with an auxiliary verb (e.g., *ble skrevet*) (Faarlund et al., 1997). The usage and distribution of auxiliary verbs in Norwegian is very similar to that of English, however, the treebank does not have a separate category, or even feature, for auxiliary verbs, but instead simply treats them as verbs.

(3.2) *Fondet* *bør* *forvaltes* *av* *Norfund*
Fund.DEF should manage.PASS by Norfund
'The fund should be managed by Norfund'

Perfect participles, e.g., *skrevet* 'written', can function either as verbs, e.g., *Han hadde skrevet boken* 'He had written the book', or as adjectives, e.g., *En skrevet bok* 'A written book'. Note that perfect participles appearing as adjectives can be inflected for definiteness in Norwegian, e.g., *Den skrevne boken* 'The written book'. To determine whether a perfect participle is an adjective or a verb, the authors of NDT utilized certain syntactic tests. If the participle is an attribute preceding a head noun, it should be tagged as adjective, but when the participle is a complement to the auxiliary verbs *ha* 'have' or *få* 'get', or a modal verb, it should be assigned the `verb` tag. However, if the participle is complement to the auxiliary verbs *være* 'be' or *bli* 'become', it could be either verb or adjective. If a degree adverb such as *fort* 'quickly' can precede the participle, it should be tagged as verb, and if it can be modified by a manner adverb such as *veldig* 'very', it should be given the `adj` tag. The treebank uses the bracket tag `<perf-part>` as a feature for perfect participles which are adjectives (Kinn et al., 2013). As present participles are generally regarded as adjectives (Faarlund et al., 1997), we discuss them in Section 3.3.3.

Penn Treebank has a total of six verb tags, among those `VB` for verb base form, `VBD` for verbs in past tense and `VBZ` for verbs in third person singular present. In addition to these tags, they have a separate tag for modal auxiliary verbs, `MD`. It makes sense to have more fine-grained verb tags for English, as verbs in English agree with their arguments (subject and object) in person and number (cf. *I am*, *you are*, *he is*). The tag set of SUC is far more coarse-grained with only a single tag for verbs, `VB`, while the UD tag set includes `VERB` and a separate `AUX` tag for auxiliary verbs.

3.3.3 Adjectives

Adjectives are words that describe or modify other words, mostly nouns, by assigning them certain attributes. In Norwegian, adjectives agree with the noun they modify in terms of gender, number and definiteness. A given adjective generally takes on three different forms: indefinite masculine/feminine singular (e.g., *stor* 'big'), indefinite neuter singular (e.g., *stort*) and definite/plural (e.g., *store*). Most, but not all, adjectives can furthermore be inflected for degree, viz., positive (e.g., *stor*), comparative (e.g., *større* 'bigger') and superlative (e.g., *størst* 'biggest'). Comparative and superlative can be expressed either through inflection or periphrastically with a degree adverb (e.g., *mer tilfreds* 'more satisfied').

The treebank operates with five different features for adjectives: gender, number, type, definiteness and degree. Gender, number, definiteness and degree are as just described. The type feature is use to denote the type of the adjective, and applies to adjectives which are in some sense 'special', hence most adjectives will not be assigned a type. The five available types are `<adv>` 'adverbial', `<ordenstall>` 'ordinal number', `<perf-part>` 'perfect participle', `<pres-part>` 'present particle' and `fork` 'abbreviation'. Adjectives that can occur as adverbials include temporal adverbs that can be inflected, such as *ofte* 'often' and *snart* 'soon', cf. *veldig ofte* 'very often' and *ganske snart* 'pretty soon', respectively. As previously mentioned, bracket tags give additional information about the token that might be relevant, but does not concern the morphological form directly (Kinn et al., 2013).

Participles are verb forms that can function as adjectives and hence modify nouns, e.g., *en patentert oppfinnelse* 'a patented invention' (perfect participle) or *en levende person* 'a living person' (present participle). Generally, if the perfect participle is an attribute preceding a noun, it is tagged as adjective, otherwise, it is tagged as verb. However, these are just the general principles, and there are exceptions; see section 3.3.2 for the full discussion on the treatment of perfect participles in the treebank. Present participles are almost exclusively used as adjectives in Norwegian, where they generally appear in the same syntactic positions as other adjectives, i.e., as complements to nouns, predicates, etc., and can hence best be described as an adjective. The exceptions are restricted to very specific syntactic constructions with continuous aspect in which the present participle follows the auxiliary verb *bli* 'become' (Faarlund et al., 1997).

Penn Treebank operates with three adjective tags: `JJ` for adjectives, `JJR` for comparative adjectives and `JJS` for superlative adjectives. Furthermore, they have separate tags for participles: `VBG` for present participle (also known as gerund) and `VBN` for past participle. In SUC, similarly to Penn Treebank, we see the `JJ` tag for adjectives and a separate tag for participles, `PC`. In the UD tag set, there is a single tag for adjectives, namely `ADJ`.

3.3.4 Determiners

Determiners are words that occur with a noun or noun phrase, where they determine or specify the reference of the noun or noun phrase (Faarlund et al., 1997), e.g., *noen* 'some' or *min* 'my'. Determiners in Norwegian can have gender (feminine, masculine or neuter), number (singular or plural) and definiteness (definite or indefinite), agreeing with the noun or noun phrase they modify. NDT also has a number of possible types for determiners: possessives (denoting possession or belonging), demonstratives (referring to an entity in the context), quantifiers (expressing quantity of some kind), amplifiers (amplifying degree of ownership, etc.) and interrogatives (introducing a question).

In some cases, the noun or noun phrase is not explicitly expressed, but instead implicit, e.g., *Min (bil) er nyere enn din* 'My (car) is newer than yours' (Faarlund et al., 1997). Possessives may also follow the noun, cf. Example 3.3.

Both Penn Treebank and SUC use the `DT` tag for determiners and have a separate category for possessive pronouns (`PRP$` and `PS`, respectively). Determiners in the PTB tag set include articles such as *every* and *no*, indefinite determiners such as *any*, *each* and *those*, and instances of *all* and *both* when they do not precede a determiner or possessive pronoun (Santorino, 1990). The tag set of the Penn Treebank furthermore have a separate category for predeterminers, `PDT`. The English language distinguishes between possessive determiners (e.g., *my*) and possessive pronouns (e.g., *mine*). This is not the case for Norwegian, which uses the same form in both cases, thus simplifying the tagging.

The UD tag set has a single category for determiners, `DET`. It does, however, have a separate tag for numerals, `NUM`, which includes quantifiers, which function as determiners and are tagged as such in NDT.

(3.3) *Jeg* *kjente* *ikke* *konen* *hans* *(...)*
I knew not wife.DEF his
'I did not know his wife'

3.3.5 Pronouns

Pronouns are words that can take the place of a noun or noun phrase, e.g., *hun* 'she' or *vi* 'we'. Norwegian pronouns can exhibit gender (feminine, masculine or neuter), number (singular or plural), person (first, second or third) and case (nominative or accusative). Note that case-marking only applies to pronouns in Norwegian, as in English. We furthermore see four types of pronouns in the NDT: interrogative (`sp`), personal (`pers`), reflexive (`refl`) and reciprocal (`res`). One generally also considers possessives as pronouns, but these are tagged as determiners in the Norwegian Dependency Treebank, for reasons we consider in 3.3.4.

Reflexive pronouns are pronouns that refer to a preceding (pro)noun, its antecedent. In Norwegian, the accusative form of personal pronouns in first and second person can act as reflexive, e.g., *Jeg så meg om* 'I looked around', *jeg* being first person singular nominative and *meg* being (normally, in non-reflexive constructions) first person singular accusative. However, the reflexive pronoun *seg* is exclusively reflexive, used only in third person, cf. Example (3.4).

(3.4) *Israel bryr seg om dette*
Israel cares REFL.3 about this
'Israel cares about this'

The Penn Treebank tag set contains four tags for pronouns: `PRP` for personal pronouns, `PRP$` for possessive pronouns, `WP` for wh-pronouns and `WP$` for possessive wh-pronouns. SUC operates with the `PN` tag for pronouns, additionally separating possessive pronouns in the `PS` tag. For interrogative/relative pronouns and interrogative/relative possessive pronouns, they have the tags `HP` and `HS`, respectively. The UD tag set has a single tag for pronouns, namely `PRON`.

3.3.6 Adverbs

Adverbs are words that modify verbs, adjectives, determiners and more, and typically express manner, time, degree, etc.. It is sometimes also regarded as a catch-all category for words unfit for other categories. In the NDT, the `adv` 'adverb' category does not have any features and hence no further morphological information. Adverbs that are derived from adjectives by conversion (also known as zero derivation), i.e., adverbs that have the same form as their corresponding adjective and normally express the manner in which a verb is performed, are simply tagged as adjective, but given the feature `<adv>`. Furthermore, these adverbs are given the exact same morphological features as the adjective from which they are derived. Other adverbs, such as locative or temporal adverbs, are simply tagged as adverbs.

Penn Treebank operates with four different tags for adverbs: `RB` for adverbs, `RBR` for comparative adverbs, `RBS` for superlative adverbs and lastly `WRB` for wh-adverbs. SUC has a single tag for adverbs (`AB`), as does the UD tag set (`ADV`).

3.3.7 Other Categories

Several parts-of-speech in the treebank (mostly closed classes) do not have any available features, hence there are no features to use for tag set modifications. These categories are, apart from `adv`, `inf-merke` 'infinitive marker', `interj`

'interjection', `konj` 'conjunction', `prep` 'preposition', `sbu` 'subordinate conjunction' and `ukjent` 'unknown' (mostly foreign words).

Penn Treebank uses the `TO` tag for both uses of the word 'to', including the infinitive marker. SUC tags infinitive markers with the `IE` tag. In the UD tag set, infinitive markers are tagged as particles (`PART`). Interjections receive the `UH` tag in Penn Treebank, `IN` in SUC and `INTJ` in the UD tag set. Conjunctions are tagged with `CC` in Penn Treebank, `KN` in SUC and `CONJ` in the UD tag set. Penn Treebank groups both prepositions and subordinate conjunctions under the `IN` tag. SUC assigns `PP` to prepositions and `SN` to subordinate conjunctions, while the UD tag set uses `ADP` and `SCONJ`, respectively. For foreign words, Penn Treebank uses the `FW` tag, while SUC uses `UO`. The `X` tag in the UD tag set acts as a catch-all for foreign words as well as other categories, including abbreviations.

Chapter 4

Experimental Setup

In preparation to conducting our experiments with linguistically motivated tag set modifications, a concrete setup for the experiments needed to be established, which is presented in the following. This setup includes a proposed data set split (training/development/test) of the Norwegian Dependency Treebank, our initial tag sets and how we realize the tag set modifications by mapping the original tags to new, more fine-grained tags by appending selected sets of morphological features. We then detail the evaluation of tagging and parsing in our tag set experiments before discussing various ways of computing a baseline tagger accuracy to which we can compare the tagger performance. Finally, we present the tagger and parser used in our tag set experiments and how the sum of these components are conjoined in a pipeline under which the experiments are run.

4.1 Data Set Split

When working with data for machine learning, it is common practice to split the data into separate sets to be used for specific parts of the machine learning process. This usually involves splitting the data into three data sets: training data, which is used for training the machine learning algorithm; development data, used for evaluation of the system during development; and test data, to be used for the final evaluation of the system. It is crucial that we keep the test set separate from the other data and abstain from using it during the development as the final evaluation needs to be performed on new data for the most realistic evaluation of how the system will perform on data outside of the treebank.

A central point of concern in machine learning is that machine learning algorithms have a tendency to overfit. Overfitting occurs when the learning algorithm is excessively trained on the training data, leading to the algorithm being 'fitted' to the training data to such an extent that its performance on the training/development

Source	Genre	# Sentences	# Tokens
Aftenposten	Newspaper	4629	75832
Bergens Tidende	Newspaper	2587	38762
Dagbladet	Newspaper	3573	48640
Klassekampen	Newspaper	686	12446
Sunnmørsposten	Newspaper	847	13231
Verdens Gang	Newspaper	755	11091
Blogs	Blogs	806	12514
Government reports	Government reports	843	14797
Parliament transcripts	Parliament transcripts	970	17463
Total		15696	244776

Table 4.1: Overview of the training data set of NDT.

data is a false measure of its predictive performance on unseen data. We want the training data to be representative of the data we encounter elsewhere in the treebank (i.e., in the development and test sets), but not overly representative, as that would lead to the algorithm being geared towards the training data and highly sensitive to input data. The algorithm needs to be able to generalize from training data to unseen data.

4.1.1 Data Set Split of the Norwegian Dependency Treebank

Currently, there is no standard data set split of the Norwegian Dependency Treebank. We therefore propose a split of the treebank which we hope to establish as the new standard. A standard split is important for ease of use and reusability of the treebank, which we facilitate by splitting on files and listing what files comprise the different data sets (See Appendix A). Furthermore, we have collected the data for the data sets in separate files in CoNLL format, which will be distributed with the treebank. Our data set split was used in the Norwegian contribution to the Universal Dependencies project (Øvrelid & Hohle, 2016). See Tables 4.1, 4.2, 4.3 and 4.4 for overview of the split of the treebank.

Our split of the Norwegian Dependency Treebank splits the data into three data sets, viz., training, development and testing. 80% of the data is used in the training, 10% in the development and the final 10% resides in the held-out test data set, which is a commonly used data set split for machine learning in NLP. NDT consists of data from various sources in a range of genres, namely newspaper articles, blog posts, parliament transcripts and government reports. The data

Source	Genre	# Sentences	# Tokens
Aftenposten	Newspaper	659	11006
Bergens Tidende	Newspaper	400	5141
Dagbladet	Newspaper	448	7086
Klassekampen	Newspaper	119	2135
Sunnmørsposten	Newspaper	186	2923
Verdens Gang	Newspaper	98	1465
Blogs	Blogs	200	1444
Government reports	Government reports	100	2298
Parliament transcripts	Parliament transcripts	200	2969
Total		2410	36467

Table 4.2: Overview of the development data set of NDT.

Source	Genre	# Sentences	# Tokens
Aftenposten	Newspaper	578	9160
Bergens Tidende	Newspaper	348	5332
Dagbladet	Newspaper	415	5048
Klassekampen	Newspaper	114	1912
Sunnmørsposten	Newspaper	168	2983
Verdens Gang	Newspaper	15	277
Blogs	Blogs	144	2433
Government reports	Government reports	47	1165
Parliament transcripts	Parliament transcripts	110	1724
Total		1939	30034

Table 4.3: Overview of the test data set of NDT.

Source	Genre	# Sentences	# Tokens
Aftenposten	Newspaper	5866	95998
Bergens Tidende	Newspaper	3335	49235
Dagbladet	Newspaper	4436	60774
Klassekampen	Newspaper	919	16493
Sunnmørsposten	Newspaper	1201	19137
Verdens Gang	Newspaper	868	12833
Blogs	Blogs	1150	16391
Government reports	Government reports	990	18260
Parliament transcripts	Parliament transcripts	1280	22156
Total		20045	311277

Table 4.4: Overview of the full data set of NDT.

from these sources is organized in files, where each file contains a maximum of 100 documents. Here, a document denotes a coherent piece of text from a given source, i.e., an article, a blog post, a parliament transcript or a government report.

We want new words and sentences in the development and test data sets, while balancing the split in terms of genre. This is important because the language of different genres tends to differ in many ways, and we cannot expect a tagger that is trained exclusively on fashion blogs to perform well on the sports section of a newspaper, for instance.

The two approaches for splitting the treebank we considered was either to maintain contiguous sections within the sources (i.e., splitting on files) or to divide the corpus into units of ten sentences each, assigning the first eight sentences of each unit to the training data set and the two remaining sentences to development and testing, respectively. This is analogous to assigning every ninth sentence to development and every tenth sentence to testing, and simply assigning the remaining sentences to the training set.

We opted for contiguous sections within the sources of the treebank, where the first 80% of the files from a particular source is used for training, the next 10% is used for development and the final 10% is used for testing. The entirety of a given document is generally contained in one of the data sets, instead of having document fragments distributed over the sets. Fragments are problematic because they might lead to overfitting, as we would be testing on sentences from a document that has already been partially seen in the training. There are some overlap of documents, i.e., documents that start in one file and continue and end in the next, but this can occur only with the last document of a particular file, and

Tag set	# Tags
Original	19
Full	368

Table 4.5: Overview of the initial tag sets and their respective size.

is hence rather negligible.

4.2 Initial Tag Sets

In our experiments, we want to investigate how we can use the morphological features included in the treebank in the creation of new, more fine-grained tags. The original tag set of NDT contains 19 tags, 12 of which are morphosyntactic tags, the remaining 7 being for punctuation, symbols, etc.. In an initial experiment, we concatenated the tag of each token with its set of morphological features in order to map the original tag set to a new, more fine-grained tag set[1]. The result of this was a total of 368 tags, which is clearly very fine-grained. These two tag sets thus represent two extremes in terms of granularity, shown in Table 4.5.

As a consequence of the very fine granularity in the full tag set, we see tags in the development and test data that do not occur in the training data. As these tags are not part of the training data, the tagger has no way of learning or successfully assigning them. This is the case only for very infrequent tags, and is caused by the problem of sparse data. As there is no clear way of resolving this minor issue, we do not take any action to alleviate this.

4.3 Tag Set Mapping

In order to modify the tag set of the treebank, we need to be able to map the original tag set to a new tag set. This is done by specifying the features to be concatenated to the relevant existing tag(s) in a separate file and supplying it to the pipeline as an argument. For instance, to create new tags for nouns which include the grammatical gender, we would add *subst fem* 'noun feminine', *subst mask* 'noun masculine' and *subst nøyt* 'noun neuter' to the tag set file, thus creating the new tags `subst|fem`, `subst|mask` and `subst|nøyt`. We then replace the

[1] Hereafter referred to as the *full* tag set

ID	Token	Tag	Features	New Tag
1	*Lam*	subst	appell|nøyt|ub|ent	subst|nøyt
2	*og*	konj	_	
3	*piggvar*	subst	appell|mask|ub|ent	subst|mask
4	*på*	prep	_	
5	*bryllupsmenyen*	subst	appell|mask|be|ent|samset	subst|mask
6	|	clb	_	

Table 4.6: Example of tag set mapping, introducing gender for nouns.

original tag with the more fine-grained tag for all applicable tokens in the treebank, here corresponding to tokens tagged as subst, having either fem, mask or nøyt in the morphological features. See Table 4.6 for an example of the tag set mapping. This is one of the mappings carried out in the experiments with tag set modifications, the results of which can be seen in Section 5.3.1.

The tag set mapping needs to be deterministic and injective, i.e., for any given token, there can be at most one applicable new tag. This imposes restrictions on what tag set modifications we can carry out, which will be further discussed where relevant. We need to enforce determinism both locally, in regard to a specific feature, as well as across features when we combine features. This involves the various values for a given feature having to be mutually exclusive, ensuring local determinism. If any token in the treebank has more than one of these feature values, we would ultimately reach a dilemma in which we have to choose which feature to use. For instance, when experimenting with tense and voice for verbs, we found that many relevant tokens in the treebank had both tense and voice marked, meaning that there were two possible tags for these verbs, either appending the tense or the voice to the original tag. As this violates the determinism, we instead combined the feature values for the relevant tokens, resulting in verb|inf|pass 'verb infinitive passive' and verb|pres|pass 'verb present passive', as all passive verbs in Norwegian are either infinitive or present tense. Similar workarounds are performed when necessary and presented accordingly.

4.4 Evaluation

To evaluate and compare the performance of PoS taggers and syntactic dependency parsers, there are certain metrics regarded as 'de facto standards' in the

field. We adhere to these standards in order to facilitate a comparative analysis of the various systems that can be readily compared to previous work.

For a given class (e.g., PoS or dependency relation), the *true positives* are the instances correctly predicted as belonging to said class, while the *false positives* are the instances erroneously assigned to said class. The *false negatives* are the instances that should have been assigned to said class, but erroneously was not.

Precision measures the reliability of the system's predictions, i.e., the percentage of instances assigned to a given class by the system that actually belong in the class per the gold standard.

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives + false positives}}$$

Recall measures the robustness of the system, i.e., the percentage of instances in a given class per the gold standard that the system correctly assigned to said class.

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives + false negatives}}$$

F-score is a harmonic mean of precision and recall.

$$\text{F-score} = 2 * \frac{\text{precision * recall}}{\text{precision + recall}}$$

PoS Tagging *Accuracy* is defined as the percentage of tokens assigned the correct PoS tag.

$$\text{Accuracy} = \frac{\text{\# correctly tagged tokens}}{\text{\# tagged tokens}}$$

The TnT-included evaluation script `tnt-diff` is used to evaluate TnT as well as our MFT baseline tagger in our tag set experiments. Precision, recall and F-score are calculated by our tailor-made tagger error analysis script[2].

Syntactic Parsing *Unlabeled attachment score (UAS)* measures the percentage of tokens that are assigned the correct head by the system, while *labeled attachment score (LAS)* additionally takes the label into account.

$$\text{UAS} = \frac{\text{\# correctly attached tokens}}{\text{\# tokens}}$$

[2]See `https://github.com/petterhh/ndt-tools`

$$\text{LAS} = \frac{\text{\# correctly attached and labeled tokens}}{\text{\# tokens}}$$

Frequency-weighted difference of F-scores for a given dependency relation weights the difference in F-score between two samples by the relative frequency of the dependency relation.

$$\text{Weighted F-score diff} = \text{F-score diff} * \frac{\text{\# tokens assigned given label}}{\text{\# tokens}}$$

The parser accuracy scores, including LAS and UAS, were computed by the `eval.pl`[3] script used in the CoNLL shared tasks. It also reports the precision and recall of labeled and unlabeled attachment, which we use in our error analyses.

4.5 Baseline

It is common practice to compare the performance of PoS taggers to a pre-computed baseline for an initial point of comparison. For PoS tagging, a commonly used baseline is the Most Frequent Tag (MFT) baseline, which we use in our experiments. This involves labeling each word with the tag it was assigned most frequently in the training. For all unknown words, i.e., words not seen in the training data, there are two main approaches: assign it the most frequent tag overall in the training data, or the tag most frequently assigned to words seen only once in the training data. We adopted the latter approach for our baseline tagger, as we believe that unknown words may have much in common with words that occur only once in the training, more so than simply words of the most frequent part-of-speech. The reason for this is that unknown and infrequent words have in common that they rarely occur, and we might therefore expect them to have similar properties.

Norwegian is a synthetic language with a quite productive morphology, paving the way for many closed compound words, e.g., *trøffelhonningvinaigrette*, 'truffle honey vinaigrette'. The possibility for these kinds of derivations leads to the formation of many new words, and it is very likely that unseen data contains many closed compound words, as well as proper nouns, that were not seen in the training data. In the case of the initial tag sets, noun is the most frequent tag for words occurring only once in the training; `subst` 'noun' for the coarse-grained tag set and `subst|prop` 'proper noun' for the fine-grained tag set. If we instead were to use the most frequent tag overall as our baseline, `prep` 'preposition' would be assigned to previously unseen words when using the full tag set, mostly due to

[3] `http://ilk.uvt.nl/conll/software.html`

`prep` having no morphological features, hence not being affected by the inclusion of these. As prepositions is a closed class, and we expect to see all prepositions in the training data, this does not seem very promising, and serves to indicate that our approach for unknown words may be the better choice.

4.6 PoS Tagger

For our experiments with tag set modifications, we want a PoS tagger that is both fast and accurate. There is often a trade-off between the two, as the best taggers tend to suffer in terms of speed due to their complexity. However, a tagger that achieves both close to state-of-the-art accuracy as well as very high speed is TnT (Brants, 2000), which we introduced in Section 2.2.1. The fact that TnT was used for evaluating the universal tag set (Petrov et al., 2012), as described in Section 3.2.3, served as another good indication of TnT being appropriate for our task. The sum of these factors led to TnT being the tagger of choice for our experiments.

4.7 Syntactic Parser

In choosing a parser for our experiments, we considered previous work on dependency parsing of Norwegian, specifically that of the Norwegian Dependency Treebank, as presented in Solberg et al. (2014). They evaluated a range of dependency parsers, such as MaltParser (Nivre et al., 2007) and the Mate parser (Bohnet, 2010), which we briefly described in Section 2.3.2. They reported that the Mate parser proved best in parsing of NDT, where it achieved a labeled attachment score (LAS) of 90.41% and unlabeled attachment score (UAS) of 92.84% for Bokmål. In comparison, MaltParser with default settings reported 84.57% and 88.02% for LAS and UAS, respectively, while reaching an LAS of 89.61% and UAS of 91.96% after a round of optimization using the optimization tool MaltOptimizer (Ballesteros & Nivre, 2012). Mate was consequently chosen as the parser for our experiments with tag set modifications.

4.8 Tags & Features

As we seek to quantify the effects of PoS tagging in a realistic setting, we want to run the parser on automatically assigned PoS tags. For the training of the parser, however, we have two options: using either gold standard or automatically assigned tags. In order to settle on a configuration, we conducted experiments with gold standard and automatically assigned tags to see how they differ with respect to performance. The results of these experiments are shown in Table 4.7. They

Training	Testing	LAS	UAS
Gold	Gold	90.15%	92.51%
Gold	Auto	85.68%	88.98%
Auto	Auto	87.01%	90.19%

Table 4.7: Results from initial parsing experiments. *Gold* denotes gold standard tags, *Auto* denotes automatically assigned tags from TnT.

reveal that the combination of training and testing on automatic tags is superior to training on gold standard tags and testing on automatic tags. This is rather surprising, as one would expect gold standard tags to always be the preferred choice. This motivates us to use automatically assigned tags both for training and testing in our tag set experiments.

Note that it is absolutely crucial that the morphological features in the treebank (See column *Features* in Table 4.6) are removed when using automatic tags, as they are still gold standard. For instance, if a verb token is erroneously tagged as a noun, we could potentially have a noun token with verbal features such as tense, which markedly obfuscates the training and parsing. Another important factor is that we want to isolate the effect of PoS tags, necessitating the exclusion of morphological features.

A similar approach was employed in the dependency parser comparison of Choi et al. (2015), where they trained on automatically assigned PoS tags and excluded any morphological features from the input data. They found Mate to be the best parser for the English portion of the OntoNotes 5 corpus, beating a wide range of contemporary state-of-the-art parsers.

4.9 Pipeline

With all the components in place for our experimental setup, the pipeline under which we run each experiment is presented in Figure 4.1. First, we perform the mapping of the relevant tags in the data sets. We then train TnT on the mapped training data and use the resulting model to tag the mapped development data. Subsequently, we tag the training data with TnT and use the resulting data to train Mate, which is then used to parse the tagged development data.

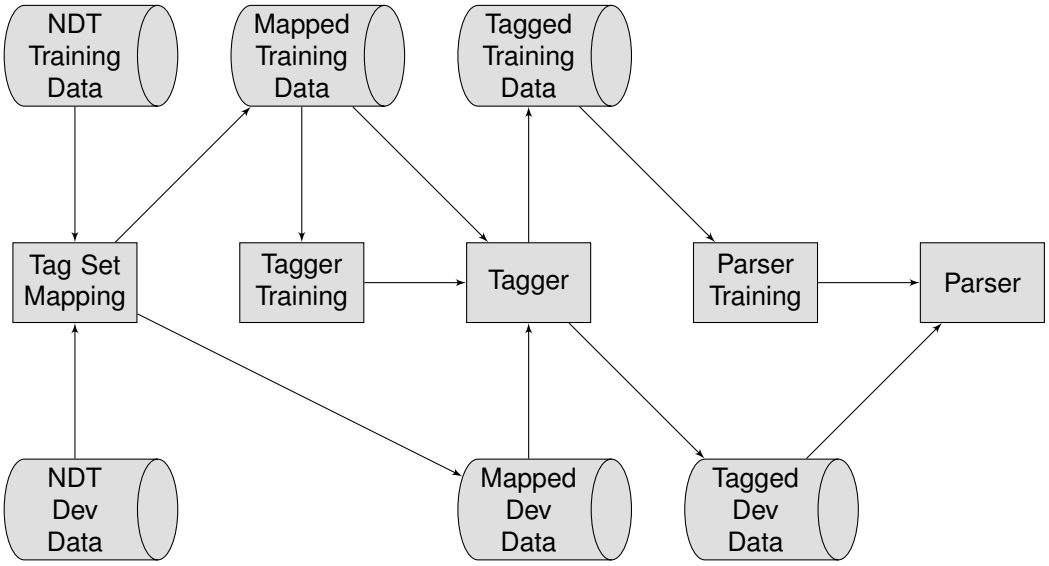

Figure 4.1: Architecture of the experimental pipeline.

Chapter 5

Tag Set Optimization

It is now time to turn to our experiments with tag set modifications. In these experiments, we will modify the tag set of the Norwegian Dependency Treebank in various ways by taking advantage of the morphological features in the treebank. They will be used in the creation of more fine-grained tags to quantify the effects of PoS tag set granularity on parsing and identify an optimized tag set for NDT, complemented by in-depth error analysis of tagging and parsing. The experiments will be carried out for the five parts-of-speech for which there is a range of morphological features in the treebank, namely determiners (`det`), verbs (`verb`), nouns (`subst`), pronouns (`pron`) and adjectives (`adj`). In the conclusion of this chapter, we will present an optimized tag set for NDT along with results from parsing with said tag set.

5.1 Motivation

As we discussed in Chapter 3, introducing more fine-grained distinctions in a tag set may improve the tagger performance as well as the performance of downstream applications such as syntactic parsers, which is the focus of this work. With more fine-grained linguistically motivated distinctions, we increase the linguistic information represented in the tags, which may assist the tagger in disambiguating ambiguous and unknown words, which in turn may aid the parser in recognizing and generalizing syntactic patterns. However, the addition of more linguistic information to the tags and thus a more fine-grained tag set will most likely lead to a drop in tagger accuracy, due to the increase in complexity. The best tagging does not necessarily lead to the best parse, and it is therefore interesting to investigate how the tag set modifications may affect the interplay between tagging and parsing. We seek to identify the most beneficial and informative morphological features for syntactic parsing of NDT and append these to the existing coarse-

Tag Set	# Tags
Original	19
Full	368

Table 5.1: Overview of the initial tag sets and their respective size.

Tag Set	MFT	Accuracy	LAS	UAS
Original	**94.14%**	**97.47%**	87.01%	90.19%
Full	85.15%	93.48%	**87.15%**	**90.39%**

Table 5.2: Evaluation of tagging and parsing the development data with the two initial tag sets.

grained tags to create a new tag set tailored for dependency parsing of Norwegian.

As we want to investigate how we can increase the linguistic quality of a tag set, we will not introduce tag set modifications which are not linguistically motivated. We will only consider distinctions we deem linguistically sensible, even if we expect them to impair the computational tractability; investigating the interplay between these two considerations is ultimately our goal.

5.2 Baseline Experiments

In an initial round of experiments, we trained and evaluated TnT and Mate on the training and development data, respectively, using the two initial tag sets described in Section 4.2 (repeated in Table 5.1 for convenience) to see how the tag set granularity would affect the tagging and parsing performance. The *original* tag set is the existing tag set in NDT, while the *full* tag set is created by concatenating the PoS tag of each token with its set of morphological features. In Table 5.2, we report the results of these experiments. We see that the tagger accuracy drastically drops when going from the original to the full tag set. The MFT baseline for the original tag set is 94.14%, while it drops by almost 9 percentage points to 85.15% for the full tag set. TnT reports an accuracy of 97.47% on the original tag set, which is reduced to 93.48% for the full tag set. These results confirm our hypothesis that the very high linguistic quality in the full tag set comes at the expense of computational tractability in terms of tagger performance.

However, the additional linguistic information provided by the full tag set improves the parser performance. With the original tag set, Mate reports an LAS of

Tag	Freq	Precision	Recall	F-score
adj	3144	95.89%	94.97%	95.43%
adv	1337	96.00%	96.93%	96.46%
det	2408	96.06%	95.14%	95.60%
inf-merke	531	99.62%	99.81%	99.72%
interj	35	92.00%	65.71%	76.67%
konj	1307	99.62%	99.16%	99.39%
prep	4878	98.37%	97.62%	97.99%
pron	2369	96.08%	97.34%	96.71%
sbu	1074	90.49%	93.95%	92.19%
subst	8944	97.69%	98.29%	97.99%
ukjent	51	67.80%	78.43%	72.73%
verb	5932	97.79%	97.08%	97.44%

Table 5.3: Tagger performance with the original tag set.

87.01% and a UAS of 90.19%, which increases to 87.15% and 90.39%, respectively, when using the full tag set. As we are looking for fine-grained distinctions that improve the syntactic parsing, these results are promising and serve to indicate that additional morphological information assists the syntactic parsers, which will be further explored later in this chapter.

To assess what parts-of-speech are challenging for the tagger, we performed error analysis of tagging with the original coarse tag set, presented in Table 5.3. The frequency of the various PoS tags and dependency relations in the following is reported per their frequency in the gold standard development data. We find that ukjent 'unknown' is the most challenging tag for the tagger, with a reported F-score of 72.73% on the development data. As this constitutes a highly disparate class containing mostly foreign words, and is the second least frequent tag in the data, this is not surprising. We see a similar trend for interjections, the least frequent tag. On the other hand, the tagger obtains an F-score of more than 97% for infinitive markers, conjunctions, prepositions, nouns and verbs. Infinitive markers and conjunctions are the two categories with an F-score exceeding 99%. Both these classes contain almost exclusively unambiguous tokens, which makes it easy for the tagger to recognize them and successfully assign them the correct tag.

In Table 5.4, we present the error analysis of parsing with the original tag set. We observe similar patterns as for tagging, where the most frequent dependency relations generally achieve the highest F-scores. DET (determiner), FINV

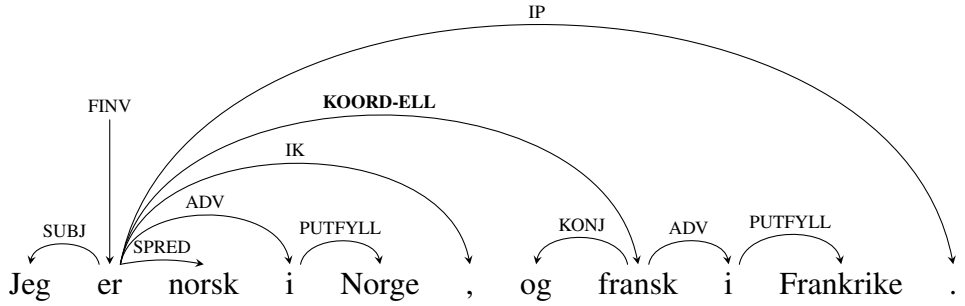

Figure 5.1: Example of sentence with dependency relation KOORD-ELL (coordination with verbal ellipsis).

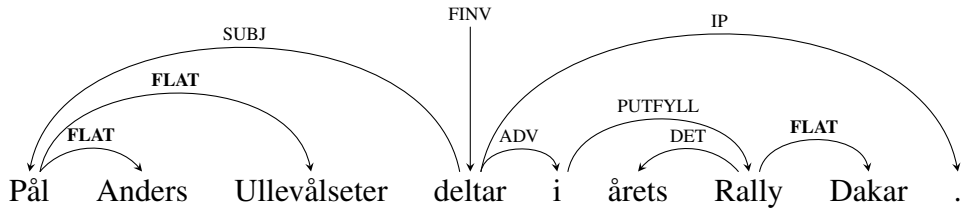

Figure 5.2: Example of sentence with dependency relation FLAT (flat structure).

(finite verb), INFV (nonfinite verb), IP (sentence-separating punctuation), KONJ (conjunction), PUTFYLL (prepositional complement) and SBU (complementizer) obtain an F-score exceeding 90%. Coordination (KOORD) and adverbial (ADV), albeit frequent, constitute challenging constructions, with reported F-scores of 76.54% and 80.06%, respectively. They are known to be notoriously difficult to parse due to the structural ambiguity they often exhibit, which we will see more clearly in our later experiments. An even more difficult type of coordination is found in coordination with verbal ellipsis (KOORD-ELL), which obtains an F-score of mere 12.90%. In coordination with verbal ellipsis, the verb in the first conjunct is elided in the second conjunct, exemplified in Figure 5.1, where *er* 'am' is implicitly the verb for the second conjunct. Other challenging constructions include apposition (APP), flat structure (FLAT), indirect object (IOBJ) and superfluous word (UKJENT). FLAT is assigned to constructions to which it is not appropriate or possible to give a hierarchical, syntactic structure, such as foreign quotes, proper nouns and other multi-words units; see Example in Figure 5.2.

Deprel	Description	Freq	Precision	Recall	F-score
ADV	Adverbial	5101	79.40%	80.73%	80.06%
APP	Apposition	285	46.32%	49.07%	47.66%
ATR	Attribute	4010	83.82%	82.86%	83.34%
DET	Determiner	2435	93.14%	88.94%	90.99%
DOBJ	Direct object	1982	87.64%	85.44%	86.53%
FINV	Finite verb	2097	96.90%	96.76%	96.83%
FLAT	Flat structure	691	76.85%	76.40%	76.62%
FOBJ	Formal object	10	40.00%	66.67%	50.00%
FOPRED	Free object predicative	7	28.57%	50.00%	36.36%
FRAG	Fragment	301	84.05%	84.05%	84.05%
FSPRED	Free subject predicative	49	32.65%	40.00%	35.95%
FSUBJ	Formal subject	360	85.56%	75.86%	80.42%
IK	Sentence-internal punctuation	5	20.00%	100.00%	33.33%
INFV	Nonfinite verb	1761	95.29%	95.50%	95.39%
INTERJ	Interjection	34	55.88%	70.37%	62.29%
IOBJ	Indirect object	74	63.51%	83.93%	72.31%
IP	Sentence-separating punctuation	98	95.92%	95.92%	95.92%
KONJ	Conjunction	1297	93.29%	93.36%	93.32%
KOORD	Coordination	1344	76.71%	76.37%	76.54%
KOORD-ELL	Coordination with verbal ellipsis	33	12.12%	13.79%	12.90%
OPRED	Object predicative	87	44.83%	58.21%	50.65%
PAR	Parenthetical expression	188	72.34%	86.62%	78.84%
POBJ	Potential object	7	42.86%	100.00%	60.00%
PSUBJ	Potential subject	205	65.85%	82.32%	73.17%
PUTFULL	Prepositional complement	4433	94.90%	94.39%	94.64%
SBU	Complementizer	1071	95.99%	95.19%	95.59%
SPRED	Subject predicative	1025	84.88%	84.06%	84.47%
SUBJ	Subject	3102	88.36%	89.94%	89.09%
UKJENT	Superfluous words	15	0.00%	0.00%	0.00%

Table 5.4: Parser performance in terms of labeled attachment with the original tag set.

5.3 Tag Set Experiments

We will modify the tags for nouns, verbs, adjectives, determiners and pronouns in NDT by appending selected sets of morphological features to each tag in order to increase the linguistic information expressed by the tags. For each of these categories, we will provide a brief recap of the linguistic considerations as discussed in Section 3.3 and present the set of available morphological features before going into how these features were used in the tag set modifications and the results of tagging and parsing with these modifications. Finally, we present error analysis of tagging and parsing with the most promising tag set modification in terms of parser performance.

For each tag, we first experiment with each of the features in isolation before employing various combinations of them. We base our choices of combinations on how promising the features are and what we deem worth investigating in terms of linguistic utility, in order to see how the features might interact.

To gain a better understanding of what constructions constitute challenges for the parser, and similarly the most challenging categories for the tagger, we perform error analysis of the most promising tag set modification for a particular category in terms of parser performance (specifically, labeled attachment score). Precision and recall for labeled attachment are reported by the aforementioned `eval.pl` script (as used in the CoNLL shared tasks), from which we calculate the F-score, which serves as the harmonic mean of precision and recall. The corresponding precision, recall and F-score for tagging are computed by our tailor-made tagger error analysis script[1].

We do not perform statistical significance testing of the differences in parser accuracy scores for all tag set modifications. Instead, we test for statistical significance for the most successful tag set modification for each respective category in Section 5.4, where we combine the best tag set modification for each category to a final, optimized tag set.

For the parser error analysis, we use frequency-weighted differences in F-scores, where the difference in F-score from the baseline (i.e., the original tag set) for a particular dependency relation is weighted by its relative frequency in the gold standard development set. Weighting the differences by relative frequency is crucial because improvements from baseline are more significant for more frequent relations; the more frequent a relation is, the greater the effect of said improvement. We report the five most improved dependency relations.

The complete overview of the tag set modifications used in the experiments are presented in Appendix B.

[1] See `https://github.com/petterhh/ndt-tools`

Feature	Values	Description
Definiteness	`be, ub`	Definite, indefinite
Gender	`fem, mask, nøyt`	Feminine, masculine, neuter
Number	`ent, fl`	Singular, plural
Type	`appell, prop`	Common, proper
Case	`gen`	Genitive

Table 5.5: Overview of available morphological features for nouns.

5.3.1 Nouns

In Norwegian, there is agreement in gender, definiteness and number between nouns and their modifiers (adjectives and determiners), motivating us to investigate how these properties interact in syntactic parsing. In addition to gender, definiteness and number, nouns are marked with type[2] and case in NDT. The features and their respective set of values are presented in Table 5.5.

The results from tagging and parsing with modifications to nouns are reported in Table 5.6. As nouns constitute the largest class in the treebank by far (as shown in Table 5.3), the effects are greatest for noun tokens in terms of overall change in performance. Apart from case, none of the tag set modifications improves the tagging. However, they all give rise to increases in parser accuracy scores. Genitive case marks possession, hence nouns marked with genitive case are quite different from other nouns, taking a noun phrase as complement. Distinguishing on type is useful and informative, as evident by the presence of separate tags for proper and common nouns in many tag sets, such as those of PTB, SUC and UD (described in Section 3.2). When introducing the distinction of type, we see a large increase in parser accuracy scores, with an LAS of 88.07% and UAS of 91.11%, both exceeding the baseline by more than a percentage point. Definiteness is the most informative feature for parsing, achieving LAS of 88.27% and UAS of 91.42%

We then combined the most promising tag set modifications to investigate how they might interact and assist each other. The most successful combinations in terms of LAS are case and definiteness (88.39%), type and case (88.46%), and

[2]Note that we throughout our experiments with all five categories only consider main types when experimenting with the type feature. Secondary types, i.e., types that come in addition to the main type and serve as less linguistically informative, e.g., `fork` (abbreviation) and `høflig` (polite), are discarded. Norwegian, like many other Germanic languages, has separate forms for polite/formal possessives, indicated with capital first letter, e.g., *Deres* 'your(s)'. They are, however, practically nonexistent in current Norwegian, with only two occurrences in NDT.

Feature(s)	MFT	Accuracy	LAS	UAS
—	**94.14%**	97.47%	87.01%	90.19%
Case	93.77%	**97.48%**	87.63%	90.72%
Definiteness	89.67%	97.00%	88.27%	91.42%
Gender	89.54%	96.09%	87.21%	90.36%
Number	90.04%	96.37%	87.97%	91.00%
Type	91.90%	96.92%	88.07%	91.11%
Case & definiteness	89.65%	97.03%	88.39%	91.44%
Type & case	91.73%	96.92%	88.46%	91.51%
Type & definiteness	89.65%	96.99%	88.44%	91.48%
Type & number	90.02%	96.37%	87.95%	90.95%
Type, case & definiteness	89.61%	97.05%	**88.81%**	**91.73%**
Type, definiteness & number	88.39%	96.46%	88.06%	91.12%

Table 5.6: Results of experiments with modified PoS tags for nouns.

type and definiteness (88.44%). This led us to combine all of them in a final experiment, in which we reach an LAS of 88.81%, improving upon the baseline by 1.8 percentage points, while also achieving the second highest tagger accuracy out of all the experiments.

The tagger error analysis for the combination of type, case and definiteness is shown in Table 5.7. We observe that common nouns without marked definiteness are the most challenging for the tagger, while definite common nouns and indefinite common nouns achieve the highest F-scores, both close to 97%, coincidently also being the most frequent tags. Turning to the parser error analysis in Table 5.8, we find that the dependency relation DET (determiner), occurring 2435 times in the gold standard development data, benefits most from the tag set modification, with an increase in F-score of more than 4 percentage points. FLAT (flat structure), the second most improved dependency relation, is mostly assigned to multi-words units such as proper nouns, and benefits greatly from additional information about nouns, especially knowing whether a noun is common or proper, reflected in the large improvement in parser accuracy scores.

5.3.2 Verbs

Verbs in NDT may take on six different feature values, shown in Table 5.9. Note that both voice and mood have only a single value, `pass` (passive) and `imp` (imperative), respectively. Verbs without `pass` are implicitly active, and verbs which

Features	Tag	Freq	Precision	Recall	F-score
Baseline	subst	8944	97.69%	98.29%	97.99%
Type, case & definiteness	subst|appell|ub	4185	96.73%	96.80%	96.76%
	subst|appell|be	2193	94.94%	98.36%	96.62%
	subst|prop	2022	94.40%	92.58%	93.48%
	subst|prop|gen	154	88.55%	95.45%	91.87%
	subst|appell|be|gen	148	95.89%	94.59%	95.24%
	subst|appell	128	92.92%	82.03%	87.14%
	subst	89	96.39%	89.89%	93.02%
	subst|appell|ub|gen	25	95.65%	88.00%	91.67%

Table 5.7: Tagger performance with the most promising tag set modification for nouns, namely type, case and definiteness.

Features	Deprel	Freq	Baseline	W/ feat	Diff
Type, case & definiteness	DET	2435	90.99%	95.18%	0.3176
	FLAT	691	76.62%	89.00%	0.2664
	PUTFYLL	4433	94.64%	96.04%	0.1938
	SUBJ	3102	89.09%	91.08%	0.1927
	ATR	4010	83.34%	84.83%	0.1855

Table 5.8: The five most improved dependency relations in terms of F-score, ranked by their weighted difference, for the most promising tag set modification for nouns, namely type, case and definiteness.

Feature	Values	Description
Mood	`imp`	Imperative
Tense	`inf, pres, pret, perf-part`	Infinitive, present, preterite, past perfect (participle)
Voice	`pass`	Passive

Table 5.9: Overview of available morphological features for verbs.

are not imperative, are implicitly indicative.

Initially, we modified the tag set by introducing each of the features in separate experiments, before trying out various combinations of these. For combinations with voice and tense, we included both voice and tense in the tag, as the tags need to be injective and all tokens with voice marked also have tense marked.

In an additional experiment, we mapped the verb tenses (mood, in the case of imperative) to finiteness. Finite verbs are verbs that can serve as root in an independent clause, i.e., verbs in imperative, present or preterite, which nonfinite verbs, i.e., infinitive and past perfect (participle), cannot. This distinction is syntactically grounded, and we might therefore expect it to positively impact the syntactic parsing, as finite verbs and nonfinite verbs appear in completely different syntactic constructions. All verbs have finiteness, hence this distinction has broad coverage.

As noted in Section 3.3.2, NDT does not have a separate tag for auxiliary verbs. As these act quite differently from main verbs, which they often precede, this distinction would be quite useful and interesting to investigate in our experiments. However, NDT does not have features for auxiliary verbs, so we would have to make use of the syntactic structure of the verbs to implement this distinction. This is beyond the scope of this thesis, hence not implemented. In the conversion of NDT to UD (Øvrelid & Hohle, 2016), the tagger error analysis revealed that the `AUX` (auxiliary verb) tag of the UD PoS tag set obtains an F-score of 91%, indicating that the distinction between auxiliary and main verbs is difficult for the tagger to make. As all auxiliary verbs in Norwegian may also take the form of a main verb (from which they are grammaticalized), this drop in accuracy is expected.

As Table 5.10 shows, the introduction of more fine-grained distinctions rarely lead to an increase in tagger accuracy. In fact, none of them improve upon the baseline; the original tag set achieves the highest accuracy overall. This serves to indicate that more fine-grained distinctions for verbs might not be beneficial for tagging Norwegian. However, more fine-grained linguistically motivated dis-

Feature(s)	MFT	Accuracy	LAS	UAS
—	**94.14%**	**97.47%**	87.01%	90.19%
Mood	94.12%	97.43%	87.04%	90.19%
Tense	93.74%	97.30%	86.97%	90.18%
Voice	94.13%	97.45%	86.96%	90.09%
Mood & tense	93.74%	97.31%	87.12%	90.31%
Voice & tense	93.74%	97.28%	86.99%	90.15%
Mood, tense & voice	93.74%	97.27%	86.83%	90.05%
Finiteness	93.72%	97.35%	**87.30%**	**90.43%**

Table 5.10: Results of experiments with modified PoS tags for verbs.

tinctions yield better parses in many cases. In three of the seven altered tag sets, namely for mood, mood and tense, and the finiteness mapping, we see a rise in LAS and UAS.

Imperative clauses are fundamentally different from indicative clauses, as they lack an overt subject; they are implicitly addressed at the reader(s)/listener(s). This is illustrated by the increase (albeit marginal) in LAS when introducing the distinction of mood, from 87.01% to 87.04%, even if mood (`imp`) accounts for only 23 tokens in the development data. The combination of mood and tense leads to LAS of 87.12% and UAS of 90.31%, both of which outperform the baseline. However, when combining voice, mood and tense together, we get a drop in LAS and UAS, to 86.83% and 90.05%, respectively.

The mapping to finiteness proved to greatly improve the parsing, as we saw the overall largest parser accuracy scores, with 87.30% for LAS and 90.43% for UAS, 0.29 and 0.24 percentage points higher than the baseline, respectively. This coincides with the observations seen for Swedish in Øvrelid (2008), where finiteness was found to be a very beneficial linguistic feature for parsing. Looking at the parser error analysis in Table 5.12, we see that FINV 'finite verb' and INFV 'nonfinite verb' are among the five most improved dependency relations, the others being KOORD 'coordination', SPRED 'subject predicative' and KONJ 'conjunction'. The 'law' of coordination of likes states that two elements can be coordinated only if they are of the same syntactic category. This is reflected in the improvement for finite and infinite verbs, as we generally coordinate verbs with the same finiteness. The error analysis from tagging with marked finiteness in Table 5.11 shows that nonfinite verbs are more challenging than finite verbs for the tagger, which could be caused by them being less than half as frequent.

Feature	Tag	Freq	Precision	Recall	F-score
Baseline	verb	5932	97.79%	97.08%	97.44%
Finiteness	verb|fin	3999	97.93%	97.10%	97.51%
	verb|infin	1933	93.05%	94.88%	93.95%

Table 5.11: Tagger performance with the most promising tag set modification for verbs, namely finiteness.

Feature	Deprel	Freq	Baseline	W/ feat	Diff
Finiteness	KOORD	1344	76.54%	78.51%	0.0825
	SPRED	1025	84.47%	85.56%	0.0348
	FINV	2097	96.83%	97.22%	0.0257
	INFV	1761	95.39%	95.80%	0.0223
	KONJ	1297	93.32%	93.83%	0.0206

Table 5.12: The five most improved dependency relations in terms of F-score, ranked by their weighted difference, for the most promising tag set modification for verbs, namely finiteness.

5.3.3 Adjectives

Adjectives are words that describe or modify other words, mostly nouns, by assigning them certain attributes. In Norwegian, adjectives agree with the noun they modify in gender, number and definiteness. In NDT, there are five different features for adjectives, presented in Table 5.13. Four of the five types are bracket tags, which do not concern the morphological form directly, with the last value being fork 'abbreviation', none of which serve as particularly linguistically informative.

The tagging and parsing results in Table 5.14 show that none of the tag set modifications lead to improvements in tagger accuracy. Degree and type are the most promising features in terms for tagger performance, with reported accuracy of 97.41% and 97.40%, respectively. All but two tag set modifications outperform the baseline parser accuracy scores, the most successful being degree, with a reported LAS of 87.29% and UAS of 90.44%, improvements of 0.28 and 0.25 percentage points, respectively, from the baseline. Other promising features include definiteness and type.

Turning to combinations of features, definiteness and number achieve the best results, very close to that of degree, with 0.02 percentage points lower LAS and

Feature	Values	Description
Definiteness	`be, ub`	Definite, indefinite
Degree	`komp, pos, sup`	Comparative, positive, superlative
Gender	`m/f, nøyt`	Masculine/feminine, neuter
Number	`ent, fl`	Singular, plural
Type	`<adv>, <ordenstall>, <perf-part>, <pres-part>, fork`	Adverb, ordinal number, past participle, present participle, abbreviation

Table 5.13: Overview of available morphological features for adjectives.

Feature(s)	MFT	Accuracy	LAS	UAS
—	**94.14%**	**97.47%**	87.01%	90.19%
Definiteness	93.45%	96.84%	87.14%	90.29%
Degree	94.13%	97.41%	**87.29%**	**90.44%**
Gender	93.56%	96.89%	87.10%	90.25%
Number	93.51%	96.71%	86.99%	90.10%
Type	94.12%	97.40%	87.11%	90.25%
Definiteness & degree	93.45%	96.81%	87.23%	90.39%
Definiteness & gender	92.94%	96.31%	87.18%	90.39%
Definiteness & number	93.48%	96.78%	87.27%	**90.44%**
Degree & gender	93.56%	96.87%	87.00%	90.16%
Degree & number	93.49%	96.76%	87.13%	90.26%
Definiteness, degree & number	93.47%	96.81%	87.14%	90.30%

Table 5.14: Results of experiments with modified PoS tags for adjectives.

Feature	Tag	Freq	Precision	Recall	F-score
Baseline	adj	3144	95.89%	94.97%	95.43%
Degree	adj|pos	2485	96.44%	94.93%	95.68%
	adj|komp	273	98.88%	96.70%	97.78%
	adj|sup	153	99.34%	98.69%	99.02%
	adj	233	72.53%	72.53%	72.53%

Table 5.15: Tagger performance with the most promising tag set modification for adjectives, namely degree.

Feature	Deprel	Freq	Baseline	W/ feat	Diff
Degree	ADV	5101	80.06%	80.82%	0.1208
	FLAT	691	76.62%	78.62%	0.0430
	DOBJ	1982	86.53%	87.19%	0.0409
	KOORD	1344	76.54%	77.25%	0.0299
	INFV	1761	95.39%	95.78%	0.0214

Table 5.16: The five most improved dependency relations in terms of F-score, ranked by their weighted difference, for the most promising tag set modification for adjectives, namely degree.

identical UAS. Adjectives agree with their head noun and determiner in definiteness and number, making this an expected improvement. The combination of definiteness and degree is also quite promising, obtaining LAS of 87.23% and UAS of 90.39%. It is interesting that none of the combinations surpass the experiment with degree alone, which indicates that degree does not interact with the other features in any syntactically significant way.

The tagger evaluation in Table 5.15 reveals that the 233 tokens without marked degree constitute a challenging group for the tagger, as the tagger obtains an accuracy of mere 72.53%. For tokens with degree marked, superlative is the least challenging, with accuracy exceeding 99%. Comparative comes in second with 97.78%, while the most frequent degree, positive, reaches an accuracy of 95.68%. The parser error analysis in Table 5.16 shows that adverbials, for which there are 5101 occurrences in the gold standard development data, gain most from the distinction of degree, with an increase in F-score from 80.06% to 80.82%. As noted in Section 5.2, adverbials are often difficult to parse, mainly due to structural ambiguity arising with the problem of PP attachment.

Feature	Values	Description
Definiteness	`be, ub`	Definite, indefinite
Gender	`fem, mask, nøyt`	Feminine, masculine, neuter
Number	`ent, fl`	Singular, plural
Type	`dem, forst, kvant, poss, sp`	Demonstrative, amplifier, quantifier, possessive, interrogative

Table 5.17: Overview of available morphological features for determiners.

5.3.4 Determiners

Determiners occur with a noun or noun phrase, of which they determine or specify the reference. In Norwegian, determiners agree with their head noun in gender, number and definiteness. NDT operates with four features for determiners, viz., type, gender, definiteness and number. In Table 5.17, these features and their respective set of available values are presented.

For a number of reasons, we do not report results from combinations of features for determiners, as the features could not be combined in any meaningful way. For instance, determiners in plural never have marked definiteness, hence this combination would only apply to determiners in singular and roughly correspond to the distinction on definiteness alone. Moreover, determiners in plural are not distinguished in terms of gender (thus ruling out the combination of number and gender), neither are definite determiners. Another aspect taken into consideration is that the most promising distinction, definiteness, applies to such a small number of tokens that more fine-grained distinctions would be overly sparse.

The results from the experiments with determiners are shown in Table 5.18. Introducing the type led to an increase in tagger accuracy by 0.14 percentage points to 97.61%, while marginally impacting the parsing, with LAS of 87.00%, 0.01 percentage points below the baseline, and UAS of 90.11%, 0.08 percentage points below the baseline. The fact that the baseline remains the same is especially worth noting, as increased tag set granularity almost inevitably leads to a drop in the MFT baseline. The increase in tagger accuracy when introducing the distinction of type is noteworthy, as we expected the finer granularity to lead to a decrease in accuracy. This serves to indicate that more fine-grained distinctions for determiners, which is a quite disparate category in the treebank, may be quite useful for tagging. However, as it has negative impact on the syntactic parsing, we can conclude that the type of a determiner does not assist in generalizing syntactic patterns, as most determiners, regardless of type, appear in the same syntactic

Feature	MFT	Accuracy	LAS	UAS
—	**94.14%**	97.47%	87.01%	90.19%
Definiteness	94.13%	97.49%	**87.30%**	**90.42%**
Gender	93.86%	97.28%	87.09%	90.31%
Number	94.06%	97.49%	87.04%	90.18%
Type	94.14%	**97.61%**	87.00%	90.11%

Table 5.18: Results of experiments with modified PoS tags for determiners.

constructions (i.e., before a noun or noun phrase).

Gender, on the other hand, improved the parsing, but complicated the tagging, as the various genders are often difficult to differentiate, especially so in the case of masculine and feminine, which share many of the same determiners. The number of a determiner, i.e., singular or plural, led to a small increase in tagger accuracy and LAS, while marginally lower UAS, 90.18%, 0.01 percentage points lower than that of the original tag set. Almost all determiners have number, and the introduction of this distinction led to small increases in tagger accuracy and LAS, but marginally lower UAS. The introduction of definiteness to the determiners led to the best parsing results, LAS of 87.30% and UAS of 90.42%, while also increasing the tagger accuracy slightly. The increase in LAS and UAS is rather interesting, as Table 5.19 shows that this change applies to only 121 tokens. As this accounts for a very small number of tokens, coupled with the previously noted considerations, we did not consider further fine-grained modifications with definiteness. This goes to show that tokens with overt definiteness have noticeable impact on the syntactic parsing, and that distinguishing on definiteness is very beneficial.

When we look at the tagger performance for definiteness in Table 5.19, we see that indefinite determiners achieve an F-score of 100%, while definite determiners reach 96.08% and the remaining determiners without marked definiteness obtain an F-score of 95.56%. The tagger perfectly tags the indefinite determiners, which constitutes a quite closed class, with only four tokens, viz., *egen* (amplifier 'own', feminine/masculine), *eget* (amplifier 'own', neuter), *annen* (demonstrative 'other', feminine/masculine) and *annet* (demonstrative 'other', neuter), all singular. Comparing the results to the baseline, we see a decrease of 0.04 percentage points for the 'base' tag `det`.

The error analysis of parsing with definiteness for determiners is shown in Table 5.20, and it is evident that adverbials (ADV) benefit the most from the distinction of definiteness, together with DOBJ 'direct object', KOORD 'coordination',

Feature	Tag	Freq	Precision	Recall	F-score
Baseline	det	2408	96.06%	95.14%	95.60%
Definiteness	det	2287	95.99%	95.15%	95.56%
	det|ub	72	100.00%	100.00%	100.00%
	det|be	49	92.45%	100.00%	96.08%

Table 5.19: Tagger performance with the most promising tag set modification for determiners, namely definiteness.

Feature	Deprel	Freq	Baseline	W/ feat	Diff
Definiteness	ADV	1597	80.06%	80.71%	0.1034
	DOBJ	1982	86.53%	87.32%	0.0485
	KOORD	1344	76.54%	77.60%	0.0443
	KONJ	1297	93.32%	93.90%	0.0236
	SUBJ	3102	89.09%	89.33%	0.0230

Table 5.20: The five most improved dependency relations in terms of F-score, ranked by their weighted difference, for the most promising tag set modification for determiners, namely definiteness.

KONJ 'conjunction' and SUBJ 'subject'. Definiteness is known to be a distinguishing property for syntactic arguments, as noted by Croft (2003), who found a cross-lingual tendency for subjects to be definite and objects to be indefinite.

5.3.5 Pronouns

Pronouns include personal, reflexive, reciprocal and interrogative pronouns in NDT. Pronouns can be assigned five features, i.e., case, gender, number, person and type, with their respective values shown in Table 5.21. As possessive pronouns are assigned to the det class, they are not discussed here.

The results in Table 5.22 show that number, person and type are the most informative features for parsing, with LAS of 87.21%, 87.22% and 87.19%, respectively. However, when combining number and person, we observe a drop by more than 0.2 percentage points, indicating that these features do not interact in any syntactically distinctive way. The most interesting observation is that all experiments exceed the baseline tagger accuracy, the most improved being the most fine-grained distinction, namely type, case and number combined, obtaining a

Feature	Values	Description
Case	`akk, nom`	Accusative, nominative
Gender	`fem, fem|mask, mask, nøyt`	Feminine, feminine/masculine, masculine, neuter
Number	`ent, fl`	Singular, plural
Person	`1, 2, 3`	1st, 2nd, 3rd
Type	`pers, refl, res, sp`	Personal, reflexive, reciprocal, interrogative

Table 5.21: Overview of available morphological features for pronouns.

tagger accuracy of 97.52%. This shows that the introduction of more fine-grained distinctions for pronouns is beneficial and aids the PoS tagger in disambiguating ambiguous words. While case alone yields an LAS of 87.08%, we found that the combination of type and case, which is the most successful experiment in terms of parser performance, yields the second highest tagging accuracy of 97.51%. The reason for this is that pronouns of different type and personal pronouns of different case exhibit quite different properties and appear in different constructions. Pronouns in nominative case (i.e., subjects) primarily occur before the main verb, while pronouns in accusative case (i.e., objects) occur after the main verb, as Norwegian exhibits so-called V2 word order, requiring that the finite verb of a declarative clause appears in the second position, hence its name. The combination of type and number comes in close to the performance of type and case, with LAS of 87.27% and UAS identical to that of type and case.

Gender did not have notable impact on the parsing, which is not surprising, seeing as the various genders have roughly equal properties. There is no agreement between pronouns and verbs in neither gender, number nor person in Norwegian, unlike English, for instance, where there is agreement in number and person (e.g., *I am* vs. *You are* vs. *He is*). This agreement is the reason for Penn Treebank having separate tags for verbs in third person singular present (`VBZ`), non-third person singular present (`VBP`), etc., as we saw in Chapter 3.

Turning to the tagger error analysis in Table 5.23, we see that reflexive and reciprocal pronouns are tagged with an F-score of 100%. The reason for this is that *seg* is unambiguously the only reflexive pronoun in Norwegian, and *hverandre* 'each other' is unambiguously the only reciprocal pronoun, which simplifies the tagging, even though they are infrequent. Interrogative pronouns (`pron|sp`) receive an F-score of 99.19%, while personal pronouns without marked case is the most challenging class for the tagger, with a reported F-score of 92.13%. Personal

Feature(s)	MFT	Accuracy	LAS	UAS
—	**94.14%**	97.47%	87.01%	90.19%
Case	94.12%	97.50%	87.08%	90.21%
Gender	94.13%	97.48%	87.06%	90.23%
Number	94.13%	97.49%	87.21%	90.33%
Person	94.14%	97.49%	87.22%	90.32%
Type	94.14%	97.48%	87.19%	90.40%
Number & person	94.13%	97.49%	96.98%	90.16%
Type & case	94.12%	97.51%	**87.30%**	**90.41%**
Type & number	94.13%	97.49%	87.27%	**90.41%**
Type & person	94.14%	97.49%	87.00%	90.14%
Type, case & number	94.12%	**97.52%**	87.11%	90.36%

Table 5.22: Results of experiments with modified PoS tags for pronouns.

pronouns in nominative or accusative reach an F-score exceeding 98%.

The parser error analysis in Table 5.24 shows that adverbial (ADV), coordination (KOORD), subject (SUBJ), attribute (ATR) and prepositional complement (PUTFYLL) are the dependency relations benefiting the most from the distinction of type and case. Adverbials and coordination are the most improved, just as for determiners and adjectives. As they are notoriously challenging for parsers, as seen in Section 5.2, the additional linguistic information greatly assists the parser in parsing these constructions.

5.4 Optimized Tag Set

The most successful tag set modification for each category and their respective results are presented in Table 5.25. Nouns benefit by far the most from the introduction of more fine-grained linguistically motivated distinctions, with an LAS of 88.81% and UAS of 91.73%. We observe that the most promising tag set modifications for verbs, adjectives, determiners and pronouns all reach LAS of ~87.30% and UAS of ~90.40%. To investigate the overall effect of these tag set modifications, we tested each of the improvements in parser accuracy scores from baseline for statistical significance using Dan Bikel's randomized parsing evaluation comparator script[3], as used in the CoNLL shared tasks. For the most successful tag set modification for each of the categories seen in Table 5.25, the difference in

[3]Available as `compare.pl` at `http://ilk.uvt.nl/conll/software.html`

Features	Tag	Freq	Precision	Recall	F-score
Baseline	pron	2369	96.08%	97.34%	96.71%
Type & case	pron|pers|nom	1121	97.38%	99.55%	98.46%
	pron|pers	825	91.31%	92.97%	92.13%
	pron|pers|akk	203	100.00%	97.54%	98.75%
	pron|refl	143	100.00%	100.00%	100.00%
	pron|sp	62	100.00%	98.39%	99.19%
	pron	9	100.00%	88.89%	94.12%
	pron|res	6	100.00%	100.00%	100.00%

Table 5.23: Tagger performance with the most promising tag set modification for pronouns, namely type and case.

Features	Deprel	Freq	Baseline	W/ feat	Diff
Type & case	ADV	5101	80.05%	80.46%	0.0632
	KOORD	1344	76.54%	77.70%	0.0485
	SUBJ	3102	89.09%	89.47%	0.0369
	ATR	4010	83.34%	83.59%	0.0318
	PUTFYLL	4433	94.64%	94.81%	0.0241

Table 5.24: The five most improved dependency relations in terms of F-score, ranked by their weighted difference, for the most promising tag set modification for pronouns, namely type and case.

Category	Feature(s)	MFT	Accuracy	LAS	UAS
Baseline	—	94.14%	97.47%	87.01%	90.19%
Noun	Type, case & definiteness	89.61%	97.05%	88.81%	91.73%
Verb	Finiteness	93.72%	97.35%	87.30%	90.43%
Adjective	Degree	94.13%	97.41%	87.29%	90.44%
Determiner	Definiteness	94.13%	97.49%	87.30%	90.42%
Pronoun	Type & case	94.12%	97.51%	87.30%	90.41%

Table 5.25: Results of tagging and parsing with the most successful tag set modification for each category.

LAS from the original tag set is statistically significant at significance level 0.05 (p-value < 0.05), as are all differences in UAS, except for verbs with finiteness (p-value 0.15) and pronouns with type and case (p-value 0.06).

An overview of the final, optimized tag set can be found in Table 5.26, comprising three new tags for adjectives, two for determiners, six for pronouns, seven for nouns and two for verbs, totaling 20 tags. Appending these to the original tag set comprising 19 tags, we reach a total of 39 tags for NDT.

The results from tagging and parsing with the optimized tag set are reported in Table 5.27, compared to the initial tag sets. The parser achieves an LAS of 88.87% and UAS of 91.78%, which constitutes substantial increases from the baseline, by 1.86 and 1.59 percentage points, respectively. The increase from type, case and definiteness for nouns alone is no more than 0.06 percentage points and not statistically significant, but as all the other tag set modifications are far behind in terms of parser accuracy scores, this does not serve as particularly shocking.

In the next chapter, we will evaluate various taggers and parsers on NDT using this optimized tag set in order to identify the optimal pipeline for syntactic parsing of Norwegian based on NDT.

Tag	Description
adj|komp	Comparative adjective
adj|pos	Positive adjective
adj|sup	Superlative adjective
det|be	Definite determiner
det|ub	Indefinite determiner
pron|pers	Personal pronoun
pron|pers|akk	Personal pronoun, accusative
pron|pers|nom	Personal pronoun, nominative
pron|refl	Reflexive pronoun
pron|res	Reciprocal pronoun
pron|sp	Interrogative pronoun
subst|appell	Common noun
subst|appell|be	Common noun, definite
subst|appell|be|gen	Common noun, definite, genitive
subst|appell|ub	Common noun, indefinite
subst|appell|ub|gen	Common noun, indefinite, genitive
subst|prop	Proper noun
subst|prop|gen	Proper noun, genitive
verb|fin	Finite verb
verb|infin	Nonfinite verb

Table 5.26: The optimized tag set.

Tag set	MFT	Accuracy	LAS	UAS
Original	**94.14%**	**97.47%**	87.01%	90.19%
Full	85.12%	93.46%	87.13%	90.32%
Optimized	89.20%	96.85%	**88.87%**	**91.78%**

Table 5.27: Results of tagging and parsing with the optimized tag set, compared to the initial tag sets.

Chapter 6

Optimized Pipeline for Norwegian Dependency Parsing

In Chapter 5, we optimized a tag set for syntactic parsing of the Norwegian Dependency Treebank. We will now train and evaluate the state-of-the-art PoS taggers and dependency parsers we introduced in Chapter 2 on the Norwegian Dependency Treebank using our optimized tag set. Additionally, the automatically assigned tags from the most successful tagger will be used to train and evaluate the various parsers to assess the effects of automatically assigned tags on parsing. We will finally run the most successful tagger and parser on the held-out test data with the optimized tag set and contrast these results to those obtained using the original tag set.

6.1 PoS Tagger

There is a plethora of available PoS taggers, but we will consider only at a select few publicly available state-of-the-art taggers representing a range of contemporary approaches, viz., the aforementioned TnT v2.2[1] (Brants, 2000), HunPos v1.0[2] (Halácsy et al., 2007), the Stanford tagger v3.6.0[3] (Toutanova et al., 2003) and SVMTool v1.3.1[4] (Giménez & Màrquez, 2004). TnT is an HMM-based trigram tagger, while HunPos is a re-implementation of TnT, using tag bigrams instead of unigrams to estimate the emission probability. The Stanford tagger is based on maximum entropy, also known as logistic regression, and cyclic dependency networks. Lastly, SVMTool is an implementation of Support Vector Machines

[1] http://www.coli.uni-saarland.de/~thorsten/tnt/
[2] https://code.google.com/p/hunpos/
[3] http://nlp.stanford.edu/software/tagger.shtml
[4] http://www.cs.upc.edu/~nlp/SVMTool/

(SVM). See Section 2.2.1 for further details on these taggers. The taggers were trained on the training set and evaluated on the development data set of the Norwegian Dependency Treebank, using the original and the optimized tag set. The run times of training and tagging were computed using the UNIX command `time`. The taggers were all run on the High Performance Computing (HPC) cluster at the Department of Informatics, University of Oslo.

TnT and HunPos were run with default settings and evaluated using the TnT-included `tnt-diff`, which calculates the accuracy.

The Stanford tagger comes with a variety of included architectures, determining what features are used to build the model. We use the included *generic*, described as language-independent in the documentation, and *bidirectional5words*, reported to be the most accurate in the manual.

SVMTool is highly flexible and tunable, but we limited our runs to experimenting with the option of strategy, employing three of the available seven. Apart from this, SVMTool was run with default settings. The strategies make use of a set of five predefined models, detailed in the documentation of the tagger. The training time represents the time spent learning the model(s) required for the strategy in question. Strategy 0 is the default strategy, making use of model 0 and tagging the data in a greedy on-line fashion in a single pass. With strategy 1, the data is tagged in two passes. In the first pass, the unseen morphological context remains ambiguous, while in the second pass, the tag predicted in the first pass is known and used as a feature. Strategy 2 tags the data in a single pass, but uses both Model 0 and Model 2, choosing the optimal one in a given case; if all the words in the unseen context is already known, it uses Model 0. Otherwise, it uses Model 2.

6.1.1 Results on Original and Optimized Tag Set

In a realistic setting, both speed and accuracy are important factors to consider for PoS taggers and syntactic parsers, especially considering the trade-off between the two, as very fast systems tend to suffer in terms of accuracy, while the most accurate systems often lag behind in terms of speed. In the following, we will focus on accuracy, as we are optimizing in terms of performance. However, for some applications, high speed may be more critical than high performance, and we therefore include the run times of training and running the systems, providing insights that can hopefully aid other potential users in choosing a tagger and/or parser that best meets their needs.

We report the time spent training and tagging with the various taggers on the optimized tag set in Table 6.1. SVMTool with strategy 0 is trained in 5 minutes, while the two other strategies spend roughly twice that. Strategies 0 and 2 tag the data in 14 and 17 seconds, respectively, while strategy 1 spends 29 seconds. The Stanford tagger employing the *generic* architecture spends roughly 5 minutes

Tagger	Setup	Training Time	Tagging Time
HunPos	Default	1.6s	0.6s
Stanford	Generic	5m 22s	7.1s
Stanford	Bidirectional5words	7m 47s	39.1s
SVMTool	Strategy 0	5m 16s	14.2s
SVMTool	Strategy 1	10m 51s	29.4s
SVMTool	Strategy 2	10m 21s	17.1s
TnT	Default	0.5s	0.4s

Table 6.1: Duration of training and testing the various taggers on the training and development data, respectively, using the optimized tag set.

		Accuracy	
Tagger	**Setup**	**Original**	**Optimized**
Baseline	MFT	94.14%	89.20%
HunPos	Default	97.73%	96.92%
Stanford	Generic	96.82%	94.24%
Stanford	Bidirectional5words	97.01%	94.43%
SVMTool	Strategy 0	97.70%	97.19%
SVMTool	Strategy 1	**97.90%**	**97.40%**
SVMTool	Strategy 2	97.72%	97.04%
TnT	Default	97.47%	96.85%

Table 6.2: Results of tagging the development data with the original tag set and the optimized tag set.

training, while training the *bidirectional5words* variant takes almost 8 minutes. Stanford with the *bidirectional5words* architecture, which is the slowest tagger, tags the development data in just under 40 seconds, while the *generic* variant spends 7 seconds. TnT was able to learn from the 244k tokens in the training data in under 1 second, while also spending less than a second tagging the development data. HunPos is not far behind, spending 1.6 seconds training, while tagging the development data in 0.9 seconds.

In Table 6.2, we report the performance of the taggers on the development data set with the original and the optimized tag set. It reveals that tagging with the original tag set yields better tagger accuracy than the optimized tag set, which is expected due to the increase in complexity with the 20 additional tags in the optimized tag set. We see that SVMTool employing strategy 1 achieves the highest

accuracy of 97.40% with the optimized tag set, followed by strategy 0 (97.19%) and strategy 2 (97.04%). Stanford tagger with the *generic* architecture obtains the lowest accuracy of 94.24% with the optimized tag set, followed by the *bidirectional5words* variant, which obtains an accuracy of 94.43%. TnT reaches the third lowest accuracy using the optimized tag set of 96.85%, while HunPos comes in fourth with 96.92%.

As these results show that SVMTool employing strategy 1 is the best tagger for NDT, we will use its assigned tags to investigate the effects of automatically assigned PoS tags on syntactic parsing.

6.2 Syntactic Parser

In similar fashion to the PoS taggers, the parsers were run on the High Performance Computing (HPC) cluster at the Department of Informatics, University of Oslo. They were trained on the training set and subsequently tested on the development set of NDT, the run times of which were computed using the UNIX command `time`.

MaltParser v1.8.1[5] (Nivre et al., 2007) is a transition-based dependency parser based on a deterministic parser strategy and treebank-induced classifiers; Mate v3.61[6] (Bohnet, 2010) is graph-based and uses maximum spanning trees with third-order features; RBG v1.1[7] (Lei et al., 2014) is a graph-based dependency parser employing tensor decomposition and a low-rank factorization method; TurboParser v2.3.0[8] (Martins et al., 2013) is a graph-based third-order non-projective dependency parser. These parsers are further detailed in Section 2.3.2. The parser accuracy scores, namely LAS and UAS, were computed by the `eval.pl` script used in the CoNLL shared tasks.

RBGParser was run both with default settings, employing third-order features, and 'basic' settings, employing first-order features, in order to investigate whether we could get a considerable decrease in training and parsing time while maintaining high accuracy. MaltParser was first run with default settings and then after a round of optimization with the optimization tool MaltOptimizer[9] (Ballesteros & Nivre, 2012), the duration of which was added to the training time. Mate and TurboParser were run out-of-the-box with default settings.

As morphological features generally require separate morphological analysis and hence are not included in non-annotated data, these were removed from the

[5] `http://www.maltparser.org`
[6] `http://code.google.com/p/mate-tools`
[7] `http://www.github.com/taolei87/RBGParser`
[8] `http://www.cs.cmu.edu/~ark/TurboParser/`
[9] `http://nil.fdi.ucm.es/maltoptimizer/`

input data for the most realistic comparison. Furthermore, as parsers differ with respect to the use of morphological features, a direct comparison between parsers that to varying degree make use of morphological features would be 'unfair' and misleading. A parser employing these features may report a higher accuracy than one that excludes them, while not necessarily implying superiority. We want to focus on how the PoS tags may affect the parsing and isolate the task of PoS tagging in order to investigate the effect of part-of-speech information.

We furthermore want to investigate the interplay between gold standard and automatically assigned tags and how the choice of tag configuration affects the parser results. We therefore ran the parsers with three configurations: (i) train and test on gold standard tags; (ii) train on gold standard tags and test on automatically assigned tags from SVMTool; (iii) train and test on automatically assigned tags from SVMTool.

6.2.1 Results on Optimized Tag Set

We previously noted the trade-off between speed and performance and the fact that even though we optimize in terms of performance, the speed of the various parsers are of interest. In Table 6.3, we present the run times of the various syntactic parsers, trained and tested on gold standard tags using the optimized tag set. MaltParser with default settings is by far the fastest, spending only 41 seconds training on the training data. MaltParser optimized, RBG with default settings and TurboParser, on the other hand, spend 1 hour and 15 minutes, 1 hour and 46 minutes, and 1 hour and 44 minutes, respectively. Mate is trained in 17 minutes and TurboParser in roughly 15 minutes. All parsers are able to parse the development data in less than a minute. The two variants of MaltParser parses the development data in less than 10 seconds, while Mate spends 34 seconds and RBG with basic settings spends 17 seconds. TurboParser and RBG with default settings parse the data in just under a minute.

Turning to the results in Table 6.4, it is clear that Mate, with the third fastest training and the fourth fastest parsing, is the best parser, with an LAS of 91.83% when trained and tested using gold standard tags, only behind RBG with default settings in terms of UAS when training on gold standard tags and testing on automatically assigned tags (91.46% vs. 91.60%). It is worth noting that the authors of RBG, Lei et al. (2014), report UAS as their evaluation metric and optimize their parser in terms of UAS. MaltParser with default settings obtains the lowest LAS and UAS in all experiments (86.22% and 89.53% for LAS and UAS, respectively, when using gold standard tags for both training and testing), followed by its optimized counterpart (similarly 89.93% and 92.17%). RBG with basic settings is consistently the fourth best parser and obtains an LAS of 90.58% when trained and tested on gold standard tags, while running it with default settings yields the third

Parser	Training Time	Parsing Time
MaltParser	41s	4s
MaltParser *optimized*	1h 15m 42s	7s
Mate	17m 48s	34s
RBG *default*	1h 46m 36s	59s
RBG *basic*	14m 57s	17s
TurboParser	1h 43m 6s	56s

Table 6.3: Duration of training and testing the various parsers on the training and development data, respectively, trained and tested on gold standard tags using the optimized tag set.

best LAS in all configurations (91.52% when trained and tested on gold standard tags) and UAS very close to or better than that of Mate. TurboParser consistently obtains the second best LAS (91.67% when using gold standard tags for both training and testing) and the third best UAS. Using automatically assigned tags instead of gold standard tags for training and testing leads to an expected drop in LAS of more than 3 percentage points for all parsers.

We observe an expected trend for all parsers when going from training and testing on gold standard tags to training on gold standard tags and testing on automatically assigned tags, as we see a substantial drop in parser accuracy scores. What strikes us as very interesting is that the parsers (except for MaltParser optimized) parse the development data with automatic tags more accurately when trained on automatic tags rather than on gold standard tags. This coincides with the results we found when parsing with Mate on automatically assigned tags from TnT in Section 4.8. We expected that gold standard tags would always be the best choice, as automatically assigned PoS tags contain errors which could obfuscate the training. These results indicate that this is not the case. Using automatically assigned tags for both training and testing is statistically significantly better at a significance level of 0.05 than training on gold standard tags and testing on automatically assigned tags for Mate, RBG *basic* and Turbo in terms of both LAS and UAS. For RBG with default settings, the difference is statistically significant only in terms of LAS.

	LAS			UAS		
Parser	**G-G**	**G-A**	**A-A**	**G-G**	**G-A**	**A-A**
MaltParser	86.22%	82.88%	82.96%	89.53%	87.10%	87.16%
MaltParser *optimized*	89.93%	86.38%	86.32%	92.17%	89.53%	89.51%
Mate	**91.83%**	**88.39%**	**88.80%**	**94.03%**	91.46%	**91.84%**
RBG *default*	91.52%	88.17%	88.35%	94.02%	**91.60%**	91.72%
RBG *basic*	90.58%	87.26%	87.55%	92.96%	90.56%	90.79%
TurboParser	91.67%	88.25%	88.43%	93.88%	91.41%	91.56%

Table 6.4: Results of parsing the development data using the optimized tag set. *G* denotes gold standard tags, *A* denotes automatically assigned tags from SVMTool. For instance, *G-G* denotes training and testing on gold standard tags. It is clear that training and testing on automatically assigned tags is superior to training on gold standard tags and testing on automatically assigned tags across the board. Mate is the best parser overall, only being surpassed by RBG with default settings in terms of UAS when training on gold standard tags and testing on automatic tags.

6.3 Final Evaluation

With SVMTool (employing strategy 1) established as the best tagger and Mate as the best parser on our optimized tag set, we can finally evaluate on the held-out test data set and compare our findings to the original tag set to contrast the tag sets and make clear the effects of our optimization. As in our previous runs, we experiment with both gold standard tags and automatically assigned tags: (i) train on gold standard tags and test on gold standard tags; (ii) train on automatically assigned tags and test on automatically assigned tags. We exclude training on gold standard tags and testing on automatically assigned tags as that combination was found to be consistently inferior to training and testing on automatically assigned tags in our previous experiments.

Table 6.5 presents the results of parsing the development data set and the held-out test data set with the original tag set and the optimized tag set using either gold standard tags or automatically assigned tags from SVMTool. Parsing the development data set with our optimized tag set yields very promising results, with improvements ranging from 1.52 percentage points (UAS, training and testing on gold standard tags) to 2.07 percentage points (LAS, training and testing on automatically assigned tags) compared to the original tag set, all of which are statistically significant at a significance level of 0.05. Turning to the parser results on the

		LAS		UAS	
Data Set	**Configuration**	**Original**	**Optimized**	**Original**	**Optimized**
Dev	Gold–Gold	90.15%	91.83%	92.51%	94.03%
	Auto–Auto	86.73%	88.80%	89.99%	91.84%
Test	Gold–Gold	90.55%	92.12%	92.97%	94.20%
	Auto–Auto	86.76%	88.28%	90.13%	91.22%

Table 6.5: Results of parsing the development data set and test data set with the setup we found most successful, i.e., tagging with SVMTool and parsing with Mate, compared to the results obtained with the original tag set. *Gold–Gold* denotes training and testing on gold standard tags, *Auto–Auto* denotes training and testing on automatically assigned tags from SVMTool.

held-out test data set, we see increases in parser accuracy scores of more than 1 percentage point in all experiments when using our optimized tag set, all of which are statistically significant at a significance level of 0.05. The largest increase is seen for LAS when training and testing on gold standard tags, from 90.55% with the original tag set to 92.12% with the optimized tag set[10], constituting 1.57 percentage points. The improvement seen for LAS when using automatically assigned tags is very close, from 86.76% to 88.28% (1.52 percentage points). These results indicate that the additional linguistic information in the tags of our optimized tag set greatly aids syntactic parsing and that optimizing an existing PoS tag set for a downstream application can be useful and beneficial.

[10]It is worth comparing our parser results with the first published results on parsing the Norwegian Dependency Treebank (Solberg et al., 2014) using the original tag set and gold standard tags. Parsing their held-out test data set with Mate, they reported an LAS of 90.41% and UAS of 92.84%. However, as they used a different data set split, their results are not strictly comparable to our results and should perhaps be taken with a grain of salt.

Chapter 7

Conclusion & Future Work

In this thesis, we have developed an optimized PoS tag set for syntactic dependency parsing of Norwegian. The optimized tag set is based on the tag set of the Norwegian Dependency Treebank (Solberg et al., 2014), the first treebank of its kind for Norwegian. The tag set was optimized by augmenting the original, coarse-grained tag set containing 12 morphosyntactic tags with additional linguistic information represented in the various morphological features assigned to tokens in the treebank. The improvements in parser performance with our optimized tag set indicate that a more fine-grained PoS tag set may assist syntactic parsers in recognizing and generalizing syntactic patterns, while potentially compromising the performance of PoS taggers. Our optimized tag set was attained through experiments with the range of morphological features of the parts-of-speech available in the treebank, namely noun, verb, adjective, determiner and pronoun. These experiments also served as an assessment of how and to what extent the various morphological features aid syntactic parsing.

The linguistic and computational considerations for designing PoS tag sets and how these might often conflict was discussed in Chapter 3, where we also presented a qualitative comparison of the PoS tag set of the Norwegian Dependency Treebank and those of other comparable treebanks, such as Penn Treebank and Stockholm-Umeå Corpus. Furthermore, we provided an in-depth survey of the parts-of-speech in Norwegian and their respective properties, before detailing how these are represented in the Norwegian Dependency Treebank. The morphological properties of the PoS tags in the Norwegian Dependency Treebank served as basis for our experiments with tag set modifications, where we identified the most informative morphological features for syntactic parsing of Norwegian.

In preparation to conducting our experiments with linguistically motivated tag set modifications, we established a concrete setup for the experiments, outlined in Chapter 4. As a standardized data set split (training/development/test) of the Norwegian Dependency Treebank was not yet established, we introduced a data set

split that will be distributed with the treebank and proposed as the new standard, annotated with tags from our optimized tag set. Our data set split was used in the conversion of the Norwegian Dependency Treebank to annotations adhering to the Universal Dependencies scheme (Øvrelid & Hohle, 2016), where much of the same experimental setup was used and the results of evaluating the converted treebank were compared to our results obtained on the original treebank. We created a mapping for carrying out the tag set modifications that maps the relevant existing tags to new, more fine-grained tags including more relevant morphological features. We then presented the evaluation metrics used to evaluate the performance of PoS taggers and syntactic parsers in our tag set experiments and how we altered the treebank to simulate a realistic setting. As it is common practice to compare the accuracy of PoS taggers to a pre-computed baseline, we looked at various ways of computing the commonly used Most Frequent Tag (MFT) baseline, which provided the baseline accuracy for our experiments. Finally, the sum of these components were combined into a pipeline which we employed in each of our experiments with tag set modifications.

The experimental setup established in Chapter 4 was put to use in Chapter 5, where we conducted our experiments with tag set modifications and assessed the results of these experiments to optimize the tag set of the Norwegian Dependency Treebank. In order to establish initial figures to which we could compare the results of our experiments, we conducted baseline experiments with our two initial tag sets, i.e., the original tag set comprising a total of 19 tags, and the full tag set, which was created by concatenating the coarse PoS tag of each token with its set of morphological features, yielding a total of 368 tags. The results of these experiments showed that the morphological features were informative and assisted syntactic parsing of Norwegian. We then turned to our main experiments with modified PoS tags, where we for each part-of-speech experimented with a range of morphological features to find the most syntactically informative ones, i.e., those that led to the best parser accuracy scores. In addition to utilizing each of the features in isolation, we combined the most promising features to see if the features interacted in a syntactically informative way, which often proved to be the case. For nouns, the most informative features are type, case and definiteness, and similarly type and case for pronouns. Finiteness is the most informative verbal feature, and for determiners, information about definiteness yields the best parser accuracy scores, while degree is the most syntactically informative feature for adjectives. We performed error analysis of both tagging and parsing with the most promising tag set modification for each part-of-speech to further analyze how they assisted the tagging and parsing. Finally, the most promising tag set modifications for each respective category were combined to a final, optimized tag set which proved to give a large rise in parser accuracy scores from our initial experiments.

With the optimized tag set established, we evaluated a range of state-of-the-art

PoS taggers and syntactic dependency parsers on the task of tagging and parsing the Norwegian Dependency Treebank in Chapter 6. We found SVMTool to be the best tagger and Mate the best parser, which we finally used to parse the held-out test data. Our results revealed an increase in parser accuracy scores from the original tag set of more than 1 percentage point across the board. Furthermore, we found that the combination of training and testing on automatically assigned tags is superior to training on gold standard tags and testing on automatically assigned tags. The interplay between these tag configurations is to our knowledge severely underinvestigated, and, contrary to our empirical results, the general assumption seems to be that gold standard tags are always the preferred choice, even when testing with automatically assigned tags.

This work has presented a systematic, empirical investigation of how an existing PoS tag set can be modified and optimized for the task of syntactic dependency parsing, complemented by evaluation of a range of state-of-the-art PoS taggers and syntactic parsers applied to Norwegian. This has resulted in concrete contributions to the Norwegian NLP community: (i) a data set split (training/development/test) of the Norwegian Dependency Treebank; (ii) a PoS tag set optimized for syntactic dependency parsing of Norwegian; (iii) a PoS tagger model (based on SVMTool) trained on the treebank; and (iv) a syntactic parser model (based on Mate) trained on the treebank. These resources are made publicly available[1] with the hope that they can be found useful by researchers and others interested in applying and advancing NLP applications for Norwegian.

7.1 Future Work

There are several aspects of this thesis that can be further explored in future work, including extrinsic evaluation of the effects of PoS tag sets on other downstream NLP applications besides parsing, such as sentiment analysis and named entity recognition. These applications often require tagged data, but are markedly different from syntactic parsing, hence the evaluation would involve investigating an entirely different aspect of the effects of tag set granularity.

Tokenization is the piece left of the puzzle that is a complete pipeline from raw text to parsed data for Norwegian. An off-the-shelf tokenizer for Norwegian is thus an important tool required for the facilitation of research and employment of NLP applications for Norwegian, as the tokenizer included in the Oslo-Bergen Tagger cannot be run separately and the tagger cannot reproduce the annotation choices of the Norwegian Dependency Treebank.

We have focused on Norwegian Bokmål in our work, which is the most preva-

[1] See `https://www.github.com/petterhh/ndt-tools`

lent written standard of Norwegian. However, the Norwegian Dependency Treebank also contains data in Nynorsk, the other official written standard of Norwegian, which can be used in similar fashion to the Bokmål portion of the treebank for training and evaluating various NLP applications. In the future, we would be interested in investigating how and to what extent the two varieties of Norwegian differ with respect to various morphosyntactic aspects, as they are very similar in terms of both morphology and syntax.

Appendix A

Data Set Split

The data set split (training/development/test) of the Norwegian Dependency Treebank and the files comprising each data set are described in the following. Each file contains a maximum of 100 sentences.

A.1 Training Data

The training data set of NDT consists of 15696 sentences distributed over 180 files, shown in Table A.1.

A.2 Development Data

The development data set of NDT consists of 2410 sentences distributed over 26 files, shown in Table A.2.

A.3 Test Data

The test data set of NDT consists of 1939 sentences distributed over 26 files, shown in Table A.3.

Source	File Interval	# Files
Aftenposten	ap001_0000 – ap012_0002	53
Bergens Tidende	bt001_0000 – bt005_0001	28
Dagbladet	db001a_0000 – db013_0004	42
Klassekampen	kk001_0000 – kk006_0000	10
Sunnmørsposten	sp-bm001_0000 – sp-bm001_0008	9
Verdens Gang	vg001_0000 – vg002_0003	8
Blogs	blogg-bm001_0000 – blogg-bm003_0000	9
Government reports	nou001_0000 – nou004_0000	10
Parliament transcripts	st001_0000 – st005_0000	11
Total		180

Table A.1: Overview of the files comprising the training data set of NDT.

Source	File Interval	# Files
Aftenposten	ap012_0003 – ap014_0002	7
Bergens Tidende	bt005_0002 – bt005_0005	4
Dagbladet	db013_0005 – db014_0002	5
Klassekampen	kk006_0001 – kk007_0000	2
Sunnmørsposten	sp-bm002_0000 – sp-bm002_0001	2
Verdens Gang	vg002_0004	1
Blogs	blogg-bm003_0001 – blogg-bm003_0002	2
Government reports	nou004_0001	1
Parliament transcripts	st005_0001 – st005_0002	2
Total		26

Table A.2: Overview of the files comprising the development data set of NDT.

Source	File Interval	# Files
Aftenposten	ap014_0003 – ap015_0002	7
Bergens Tidende	bt005_0006 – bt006_0001	4
Dagbladet	db014_0003 – db014_0007	5
Klassekampen	kk007_0001 – kk008_0000	2
Sunnmørsposten	sp-bm003_0000 – sp-bm003_0001	2
Verdens Gang	vg002_0005	1
Blogs	blogg-bm003_0003 – blogg-bm003_0004	2
Government reports	nou004_0002	1
Parliament transcripts	st005_0003 – st005_0004	2
Total		26

Table A.3: Overview of the files comprising the test data set of NDT.

Appendix B

Tag Sets

The following provides an overview of the tag set modifications, i.e., the additional tags, used in our tag set experiments, as described in Section 5.3.

B.1 Noun

B.2 Verb

B.3 Adjective

B.4 Determiner

B.5 Pronoun

Feature(s)	Tag(s)
Case	`subst|gen`
Definiteness	`subst|be` `subst|ub`

Gender	`subst|fem`
	`subst|mask`
	`subst|nøyt`

Number	`subst|ent` `subst|fl`
Type	`subst|appell` `subst|prop`

Case & definiteness	`subst|be`
	`subst|be|gen`
	`subst|ub`
	`subst|ub|gen`

Type & case	`subst|appell`
	`subst|appell|gen`
	`subst|prop`
	`subst|prop|gen`

Type & definiteness	`subst|appell|be`
	`subst|appell|ub`
	`subst|prop`

Type & number	`subst|appell`
	`subst|appell|ent`
	`subst|appell|fl`
	`subst|prop`

Type, case & definiteness	`subst|appell`
	`subst|appell|be`
	`subst|appell|be|gen`
	`subst|appell|ub`
	`subst|appell|ub|gen`
	`subst|prop`
	`subst|prop|gen`

Type, definiteness & number	`subst|appell|be|ent`
	`subst|appell|be|fl`
	`subst|appell|ub|ent`
	`subst|appell|ub|fl`
	`subst|prop`

Table B.1: Tag set modifications for nouns (`subst`).

Feature(s)	**Tag(s)**
Mood	`verb|imp`

Tense	`verb|inf`
	`verb|perf-part`
	`verb|pres`
	`verb|pret`

Voice	`verb|pass`

Mood & tense	`verb|imp`
	`verb|inf`
	`verb|perf-part`
	`verb|pres`
	`verb|pret`

Voice & tense	`verb|inf`
	`verb|perf-part`
	`verb|pres`
	`verb|pret`
	`verb|inf|pass`
	`verb|pres|pass`

Mood, tense & voice	`verb|imp`
	`verb|inf`
	`verb|perf-part`
	`verb|pres`
	`verb|pret`
	`verb|inf|pass`
	`verb|pres|pass`

Finiteness	`verb|fin` `verb|infin`

Table B.2: Tag set modifications for verbs (`verb`).

Feature(s)	Tag(s)
Definiteness	adj|be adj|ub
Degree	adj|komp adj|pos adj|sup
Gender	adj|m/f adj|nøyt
Number	adj|ent adj|fl
Type	adj|<adv> adj|<ordenstall> adj|<perf-part> adj|<pres-part> adj|fork
Definiteness & degree	adj|be|pos adj|be|sup adj|komp adj|pos adj|ub|pos adj|ub|sup
Definiteness & gender	adj|be adj|ub adj|ub|m/f adj|ub|nøyt
Definiteness & number	adj|be|ent adj|ub|ent
Degree & gender	adj|komp adj|pos adj|pos|m/f adj|pos|nøyt adj|sup
Degree & number	adj|komp adj|pos adj|pos|ent adj|pos|fl adj|sup
Definiteness, degree & number	adj|be|pos|ent adj|ub|pos|ent adj|komp adj|pos adj|sup

Table B.3: Tag set modifications for adjectives (adj).

Featur	Tag(s)
Definiteness	`det|be` `det|ub`

Gender	`det|fem`
	`det|mask`
	`det|nøyt`

Number	`det|ent` `det|fl`

Type	`det|dem`
	`det|forst`
	`det|kvant`
	`det|poss`
	`det|sp`

Table B.4: Tag set modifications for determiners (`det`).

Feature(s)	Tag(s)
Case	pron|akk pron|nom
Gender	pron|fem pron|fem|mask pron|mask pron|nøyt
Number	pron|ent pron|fl
Person	pron|1 pron|2 pron|3
Type	pron|pers pron|refl pron|res pron|sp
Number & person	pron|ent|1 pron|ent|2 pron|ent|3 pron|fl|1 pron|fl|2 pron|fl|3
Type & case	pron|pers pron|pers|akk pron|pers|nom pron|refl pron|res pron|sp
Type & number	pron|pers|ent pron|pers|fl pron|refl pron|res pron|sp
Type & person	pron|pers|1 pron|pers|2 pron|pers|3 pron|refl pron|res pron|sp
Type, case & number	pron|pers pron|pers|akk|ent pron|pers|akk|fl pron|pers|nom|ent pron|pers|nom|fl pron|refl pron|res pron|sp

Table B.5: Tag set modifications for pronouns (pron).

References

Ballesteros, M., & Nivre, J. (2012). MaltOptimizer: An Optimization Tool for MaltParser. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics* (pp. 58–62). Avignon, France.

Bohnet, B. (2010). Very High Accuracy and Fast Dependency Parsing is not a Contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics* (pp. 89–97). Beijing, China.

Brants, T. (2000). TnT - A Statistical Part-of-Speech Tagger. In *Proceedings of the Sixth Applied Natural Language Processing Conference.* Seattle, WA, USA.

Brill, E. (1994). Some Advances in Rule-Based Part of Speech Tagging. In *Proceedings of the 12th National Conference on Artificial Intelligence.* Seattle, WA, USA.

Choi, J. D., Tetreault, J., & Stent, A. (2015). It Depends: Dependency Parser Comparison Using A Web-Based Evaluation Tool. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics* (pp. 387–396). Beijing, China.

Croft, W. (2003). *Typology and Universals* (2nd ed.). Cambridge, England: Cambridge University Press.

Daelemans, W., Zavrel, J., Berck, P., & Gillis, S. (1996). MBT: A Memory-Based Part of Speech Tagger-Generator. In *Proceedings of the Fourth Workshop on Very Large Corpora* (pp. 14–27). Lisbon, Portugal.

Ejerhed, E., Källgren, G., Wennstedt, O., & Åström, M. (1992). *The Linguistic Annotation System of the Stockholm-Umeå Corpus Project* (Tech. Rep.). Umeå, Sweden: University of Umeå: Department of Linguistics.

Faarlund, J. T., Lie, S., & Vannebo, K. I. (1997). *Norsk referansegrammatikk.* Oslo, Norway: Universitetsforlaget.

Francis, W. N., & Kučera, H. (1979). *Manual of Information to Accompany A Standard Corpus of Present Day Edited American English, for Use with Digital Computers.* Providence, RI, USA.

Giménez, J., & Màrquez, L. (2004). SVMTool: A General POS Tagger Generator

Based on Support Vector Machines. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation* (pp. 43–46). Lisbon, Portugal.

Guevara, E. (2010). NoWaC: A Large Web-Based Corpus for Norwegian. In *Proceedings of the Sixth Web as Corpus Workshop* (pp. 1–7). Los Angeles, CA, USA.

Gustafson-Capková, S., & Hartmann, B. (2006). *Manual of the Stockholm Umeå Corpus version 2.0.* Stockholm, Sweden.

Hagen, K., Johannessen, J. B., & Nøklestad, A. (2000). A Constraint-Based Tagger for Norwegian. In *Proceedings of the 17th Scandinavian Conference of Linguistics* (pp. 31–48). Odense, Denmark.

Halácsy, P., Kornai, A., & Oravecz, C. (2007). HunPos - An Open Source Trigram Tagger. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics on Interactive Poster and Demonstration Sessions* (pp. 209–212). Prague, the Czech Republic.

Johannessen, J. B., Hagen, K., Nøklestad, A., & Lynum, A. (2011). OBT+Stat: Evaluation of a Combined CG and Statistical Tagger. In *Proceedings of the 18th Nordic Conference of Computational Linguistics* (pp. 26–34). Riga, Latvia.

Jurafsky, D., & Martin, J. H. (2009). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition* (2nd ed.). Upper Saddle River, NJ, USA: Prentice Hall.

Karlsson, F. (1992). SWETWOL: A Comprehensive Morphological Analyser for Swedish. *Nordic Journal of Linguistics*, *15*(1), 1–45.

Kinn, K., Solberg, P. E., & Eriksen, P. K. (2013). *Retningslinjer for morfologisk og syntaktisk annotasjon i Språkbankens gullkorpus* (Tech. Rep.). Oslo, Norway: National Library of Norway.

Leech, G. (1997). Grammatical Tagging. In *Corpus Annotation: Linguistic Information from Computer Text Corpora* (pp. 19–33). New York, NY, USA: Addison Wesley Longman Limited.

Lei, T., Xin, Y., Zhang, Y., Barzilay, R., & Jaakkola, T. (2014). Low-Rank Tensors for Scoring Dependency Structures. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics* (pp. 1013–1024). Baltimore, MD, USA.

MacKinlay, A. (2005). *The Effects of Part-of-Speech Tagsets on Tagger Performance* (Bachelor's Thesis). University of Melbourne, Melbourne, Australia.

Marco, C. S. (2014). An Open Source Part-of-Speech Tagger for Norwegian: Building on Existing Language Resources. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation* (pp. 4111–

4117). Reykjavik, Iceland.

Marcus, M., Santorino, B., & Marcinkiewicz, M. A. (1993). *Building A Large Annotated Corpus of English: The Penn Treebank* (Tech. Rep.). Philadelphia, PA, USA: University of Philadelphia.

Martins, A. F. T., Almeida, M. B., & Smith, N. A. (2013). Turning on the Turbo: Fast Third-Order Non-Projective Turbo Parsers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Lingustics* (pp. 617–622). Sofia, Bulgaria.

McDonald, R., & Nivre, J. (2011). Analyzing and Integrating Dependency Parsers. *Computational Linguistics*, *37*(1), 197–230.

Megyesi, B. (2001). Comparing Data-Driven Learning Algorithms for PoS Tagging of Swedish. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing* (pp. 151–158). Pittsburgh, PA, USA.

Megyesi, B. (2002). *Data-Driven Syntactic Analysis: Methods and Applications for Swedish* (Unpublished doctoral dissertation). Royal Institute of Technology, Stockholm, Sweden.

Megyesi, B. (2009). The Open Source Tagger HunPoS for Swedish. In *Proceedings of the 17th Nordic Conference on Computational Linguistics* (pp. 239–241). Odense, Denmark.

Nivre, J. (2005). *Dependency Grammar and Dependency Parsing* (MSI Report No. 05133). Växjö, Sweden: Växjö University.

Nivre, J. (2015). Towards a Universal Grammar for Natural Language Processing. In *Computational Linguistics and Intelligent Text Processing* (Vol. 9041, pp. 3–16). Springer International Publishing.

Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryiğit, G., Kübler, S., ... Marsi, E. (2007). MaltParser: A Language-Independent System for Data-Driven Dependency Parsing. *Natural Language Engineering*, *13*(2), 95–135.

Petrov, S., Das, D., & McDonald, R. (2012). A Universal Part-of-Speech Tagset. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation* (pp. 2089–2096). Istanbul, Turkey.

Ratnaparkhi, A. (1996). A Maximum Entropy Model for Part-Of-Speech Tagging. In *Proceedings of the Conference on Empirial Methods in Natural Language Processing* (pp. 133–142). Philadelphia, PA, USA.

Santorino, B. (1990). *Part-of-Speech Tagging Guidelines for the Penn Treebank Project* (Tech. Rep.). Philadelphia, PA, USA: University of Pennsylvania.

Solberg, P. E. (2013). Building Gold-Standard Treebanks for Norwegian. In *Proceedings of the 19th Nordic Conference of Computational Linguistics* (pp. 459–464). Oslo, Norway.

Solberg, P. E., Skjærholt, A., Øvrelid, L., Hagen, K., & Johannessen, J. B. (2014). The Norwegian Dependency Treebank. In *Proceedings of the Ninth Interna-*

tional Conference on Language Resources and Evaluation (pp. 789–795). Reykjavik, Iceland.

Toutanova, K., Klein, D., Manning, C. D., & Singer, Y. (2003). Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics* (pp. 173–180). Edmonton, Canada.

Östling, R. (2013). Stagger: An Open-Source Part of Speech Tagger for Swedish. *Northern European Journal of Language Technology*, *3*(1), 1–18.

Øvrelid, L. (2008). Finite Matters: Verbal Features in Data-Driven Parsing of Swedish. In *Proceedings of the Sixth International Conference on Natural Language Processing.* Gothenburg, Sweden.

Øvrelid, L., & Hohle, P. (2016). Universal Dependencies for Norwegian. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation.* Portorož, Slovenia.