Data-Driven Bitext Dependency Parsing and Alignment

Haulrich, Martin Wittorff

Document Version
Final published version

Publication date:
2012

COPENHAGEN BUSINESS SCHOOL
HANDELSHØJSKOLEN
SOLBJERG PLADS 3
DK-2000 FREDERIKSBERG
DANMARK

www.cbs.dk

ISSN 0906-6934

Print ISBN: 978-87-92842-30-5
Online ISBN: 978-87-92842-31-2

Data-Driven Bitext Dependency Parsing and Alignment

PhD Series 2-2012

Copenhagen
Business School
HANDELSHØJSKOLEN

Data-Driven Bitext Dependency Parsing and Alignment

Martin Haulrich

LIMAC PhD School
Programme in Language and Culture

PhD Series 2-2012

Data-Driven Bitext Dependency Parsing and Alignment

Martin Haulrich

PhD Thesis

Supervisor: Daniel Hardt

Submitted October 2011

LIMAC

Copenhagen
Business School
HANDELSHØJSKOLEN

Martin Haulrich
Data-Driven Bitext Dependency Parsing and Alignment

1st edition 2012
PhD Series 2.2012

ISSN 0906-6934

Print ISBN: 978-87-92842-30-5
Online ISBN: 978-87-92842-31-2

LIMAC PhD School is a cross disciplinary PhD School connected to research communities within the areas of Languages, Law, Informatics, Operations Management, Accounting, Communication and Cultural Studies.

Summary

Parallel treebanks have received increasing attention in the past few years, primarily due to their potential use in statistical machine translation. Creating parallel treebanks manually is a time-consuming and expensive task and for this reason there is considerable interest in creating treebanks automatically. This task can be solved using standard tools such as parsers and aligners. However, because parallel treebanks are based on parallel corpora, we are in a special situation where the same meaning is represented in two different ways. This thesis is about how we can exploit this information to create better parallel treebanks than we can by using standard tools.

We will work with bilingual parallel treebanks with pairs of *closely related languages*. This differs from most work in the field where the languages differ more. This presents a different challenge since it is exactly the differences in structure that are the basis of the success of methods that exploit the bilingual information available.

We will present three data-driven approaches that exploit bilingual information.

We will describe and analyze *bilingually informed* parsing. Bilingually informed parsing is monolingual parsing that is informed by the syntactic structures of sentences parallel to those being parsed. We argue why this should also work with the language pairs we use and analyze both the data we use, and the errors the bilingually informed parsers make. This approach consistently gives improvement over a baseline parser.

Building on bilingually informed parsing, we present an *iterative* approach that rests on the assumption that the better the structures that guide the parsing, the better the output of the parser. Although we see several in-

dications that this assumption is correct, we do not see consistent improvements over the bilingually informed parsing with this approach.

Finally, we test a classic *reranking* approach where monolingual parses are reranked, based on bilingual features. This approach leads to consistent improvements over the baseline.

For all approaches we test how the size of the data that the models are based on affects the effectiveness of the approach. For bilingually informed parsing and the iterative approach, we see that the increase in quality is bigger when smaller data sets are used.

We show that all the presented methods are efficient enough to process large-scale data.

Resumé

I de seneste år har der været øget fokus på parallelle træbanker. Primært på grund af deres potentielle anvendelse i statistisk maskinoversættelse. Da det er meget tidskrævende og dyrt at producere parallelle træbanker manuelt, har der været en øget interesse i at gøre dette automatisk. Denne opgave kan løses med eksisterende værktøjer som parsere og alignere. Men da parallelle træbanker er baserede på parallelle korpora, foreligger der en særlig situation, hvor den samme betydning er repræsenteret på to forskellige måder. Denne afhandling handler om, hvordan vi kan udnytte denne information til at skabe bedre parallelle træbanker, end dem vi kan skabe med standard værktøjer.

Vi arbejder med bilingvale parallelle træbanker, hvor de to sprog er nært beslægtede. Det meste arbejde der tidligere er blevet lavet på området, har været med sprog med større indbyrdes forskelle. Dette betyder at vi står overfor en anden udfordring eftersom det ofte er netop forskellen på sprogene, der bliver betragtet som grunden til, at metoder der anvender bilingval information virker.

Vi præsenterer tre data-drevne metoder, der forsøger at udnytte den bilingvale information.

Vi beskriver og analyserer *bilingval informeret* parsing. Dette er monolingval parsing, som er informeret af de syntaktiske strukturer fra sætninger, der er parallelle med dem der parses. Vi argumenterer for, at bilingval informeret parsing også virker med nært beslægtede sprog, og vi analyserer både de data vi bruger, og de fejl de bilingvalt informerede parsere laver. Denne metode giver konsekvent bedre resultater end standard parsing.

Vi bygger videre på denne metode og præsenterer en *iterativ* metode,

som bygger på den antagelse, at jo bedre de strukturer, der informerer parseren er, jo bedre vil resultatet af denne parser blive. På trods af at vi observerer flere indikationer på at antagelsen er korrekt, giver denne metode ikke konsekvent bedre resultater end bilingval informeret parsing.

Den tredje og sidste metode vi afprøver er en *reranking* metode, hvor analyser fra standard monolingvale parsere bliver rerankede på baggrund af bilingval information. Denne metode giver konsekvent bedre resultater end standard parsing.

For alle metoder afprøver vi, hvordan størrelsen af det data modellerne er baserede på, påvirker resultatet af metoden. For bilingval informeret parsing og den iterative metoder ser vi, at jo mindre data, der bliver brugt, jo bedre virker metoderne.

Vi viser, at alle de præsenterede metoder er effektive nok til at håndtere store mængder data.

Acknowledgments

First of all I would like to thank my two advisors. Matthias Buch-Kromann for getting me started and Dan Hardt for getting me to finish. I would also like to thank Joakim Nivre and Keith Hall. Not only for agreeing to be on my thesis committee but also for having helped me understand lots of different things during my years as a PhD student.

I would like to thank Anders for being much more optimistic with respect to the research we do than me. Jakob has helped with the thesis in several ways and has made coming to work much more fun.

Thanks to Niels for correcting many errors in things I have written and a huge thanks to Karin for drastically reducing the number of errors in this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The task we address in this work is the creation of parallel treebanks. The basis for a parallel treebank is a parallel corpus. A parallel corpus consists of parallel texts in two (or more) languages. The notion of parallel is not strictly defined. In most cases there will be one original text and the other text will be a translation of this, but there are also parallel corpora where the two texts represent the same meaning without either of them being a direct translation of the other. We will also use the term *bitext* for a parallel corpus.

In order to turn a parallel corpus or bitext into a parallel treebank, syntactic trees are added to the sentences on both sides. Some parallel treebanks also include alignments between words and/or nodes in the syntactic trees (Buch-Kromann, Wedekind, and Elming, 2007; Volk et al., 2010), and some do not (Čmejrek et al., 2004). Here, we are interested in the first kind, i.e. we also want the alignments.

Treebanks can be based on different syntactic theories which result in different syntactic structures. In this thesis, we will focus only on dependency structures. Figure 1.1[1] shows an example of the kind of structure we are interested in, i.e. a bitext (Danish-English) with a dependency structure for both sentences and an alignment between them. This kind of structure is the main focus of this thesis. We see that the structure consists of three independent structures, namely two syntactic trees and an alignment. The

[1]The sentence seems flawed as is says ”protein can be found in protein”, but this is how it appears in the corpus.

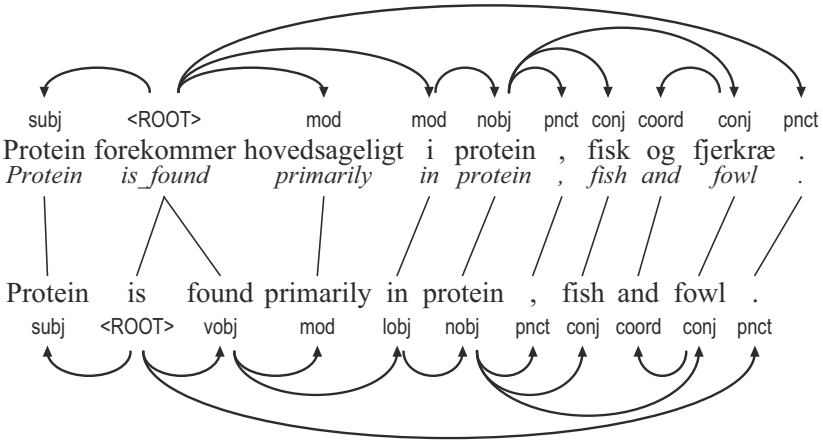

Figure 1.1: Example of the kind of structure this thesis focuses on, i.e. a structure consisting of two dependency analyses and an alignment. The example is from the Copenhagen Dependency Treebank.

problem we are trying to solve is how to create these structures in the best possible way. Although the structures in themselves are independent, it is not necessarily a good idea to create them independently. In the next sections we will describe in a little more detail the different tasks involved in creating this kind of structure.

The structure in figure 1.1 is created by a human annotator. Manual treebank annotation is in no way a trivial task, but it is not one we will address here. Instead, we are interested in creating the structures and thereby the parallel treebank automatically. This process might, as we will discuss later, require an initial treebank created by manual annotation.

1.1 Parsing

Parsing is the task of creating structures like the two syntactic trees in figure 1.1. Parsing can be done manually (by humans) or automatically (by computers) - we are interested in the latter.

In parsing, there is often a distinction between grammar-based and data-driven (Kübler, McDonald, and Nivre, 2009). In data-driven parsing the approach is to try and learn how to parse new sentences from existing linguistic data. Grammar-driven approaches rely on formal grammars. The two approaches are not mutually exclusive, but the approach taken in

the work here falls entirely in the category data-driven parsing.

1.2 Alignment

Another task that is necessary to solve when creating parallel treebanks, or doing bitext parsing and alignment, is producing the alignment. The entities being aligned can be different things. In parallel corpora the sentences and/or the words can be aligned. This is called sentence alignment and word alignment. In parallel treebanks the nodes in trees can also be aligned. This is called sub-tree alignment.

If dependency structures are used, a word alignment will actually be a sub-tree alignment and vice versa, because the only nodes in the syntactic trees are the words-nodes. This means that solving the word alignment problem also solves the sub-tree alignment problem. We expect that the trees contain some information that will be helpful to the alignment process, and for this reason we will not restrict ourselves to word alignment.

1.3 Combining Parsing and Alignment

A lot of work has been done in alignment and dependency parsing separately, and these areas are well understood. However, when combining them new issues arise. In principle the two dependency structures and the alignment structure are independent of each other (unless the theory behind the treebank has some criteria of well-formedness where the structures depend on each other), but a common assumption is that we can learn to create better structures by letting them influence each other. This is the basic assumption for all work presented here: we can achieve better results by letting the structures depend on each other when creating them, than we can by creating them independently. Apart from trying to show this empirically by actually creating better structures, we will also address the question of why this is the case.

1.4 Data Driven

In all work presented here we use so-called data-driven methods. This means that we try to create models for parsing and alignment on the basis of existing linguistic data. The fact that we will only use data-driven methods implies that we will not at any point try to hand-craft any rules.

That we use data-driven methods implies that machine learning will play an important role, as machine learning is a way of creating models on the basis of existing data. Although we use machine learning methods that are common in Natural Language Processing (NLP), we will describe these in some detail because the choice of machine learning method has a huge impact on the results one can achieve.

We do not hand-craft any rules but this does not mean that we will not look at the linguistic data. The task of feature engineering is extremely important in order to achieve good results, and it also leads to a better understanding of the data. As mentioned above, we will try to analyze the basic assumption that letting the structures affect each other increases the quality of the structures. Hopefully, this analysis will give us some insight into the data which will allow us to design better features.

1.4.1 Supervised

In machine learning one typically distinguishes between supervised and unsupervised learning (and semi-supervised which is a combination of the two).

In supervised learning the learning algorithm receives input examples together with the correct output/label for these examples, and tries to learn a model that can predict the correct output from the input. For instance, a parser that is trained in a supervised fashion will be given a treebank, and the learning algorithm will try to learn to map from the sentences to the correct trees.

In unsupervised learning the learning algorithm is given only the input. It will then try to find some structure in this without having the correct output to look at. This is commonly used in, for instance, word alignment, where the word alignment learning algorithm is given parallel texts without word alignments. The learner then tries to optimize some given

objective on these data without having any 'correct' answers to look at.

Whereas unsupervised methods are widely used in word alignment they are less common in parsing. The reason for this (apart from giving less good results) is that the structures learned and outputted by these methods do not necessarily match the linguistically motivated structures found in treebanks. As our main goal is the extension of linguistically motivated treebanks, unsupervised parsing is not well-suited. For this reason, we focus only on supervised methods[2].

1.5 Why Create Parallel Treebanks?

The task we are addressing is the creation of parallel treebanks - more specifically how to create these automatically. There are at least two reasons why this is an interesting task. The first is that solving the task can help in the creation of hand-annotated treebanks. The second is that parallel treebanks can be used to improve machine translation. There are also other uses for parallel treebank, but we will not address these.

1.5.1 Hand-Aligned Parallel Treebanks

Parallel treebanks are valuable tools from a linguistic point of view as they allow a contrastive view on linguistics. In contrastive linguistics, automatically created treebanks may not be useful because they will contain errors that may make the linguistic conclusions drawn from the treebanks invalid. However, automatically created treebanks can help in the creation of manually annotated treebanks which are better suited for linguistic investigations. By using an automatically created treebank as the basis for manual annotation, a lot of time (and money) can be saved since a lot of the more trivial annotation has already been done. This of course requires a certain level of quality of the automatically created treebank in order to avoid that the annotators will end up spending more time correcting errors from this than they would have spent annotating the text from scratch.

This use for automatically created treebanks is the main focus of this thesis.

[2] As described later we use output from unsupervised word aligners to improve the output of our supervised aligner, but we do not directly use unsupervised methods.

1.5.2 Machine Translation

Standard phrase-based statistical machine translation (SMT) systems do not directly include any linguistic information. This makes it difficult for these systems to produce the correct translation in some cases. In recent years, there has been a lot of interest in statistical machine translation models that include some kind of linguistic information. There are several different approaches to how parallel treebanks can be used in this context. Some existing and proposed systems rely directly on parallel treebanks (Hearne and Way, 2006; Buch-Kromann, 2007). Another possibility is that the joint modeling sometimes used in the creation of parallel treebanks can lead to at least one of the three structures becoming better and then this can be used to achieve better translation quality. For instance, Burkett and Klein (2008) get better alignments by doing joint modeling, compared to individual modeling, and these improved alignments lead to better translations.

SMT systems that use parallel treebanks will often require that these are large, and the quality of the translations from the system will generally depend on the quality and size of the treebanks being used. Because such large amounts of data are required for statistical MT systems, it is not feasible to annotate these treebanks by hand, and therefore automatic annotation is used. This is probably the main motivation for most work in the creation of parallel treebanks.

We will focus less on the use of automatically created treebanks in SMT. With respect to SMT, our focus will mainly be that the methods presented are efficient enough to process large amounts of text.

1.5.3 Evaluation

It is important to consider that the evaluation criteria might be different for the two different uses for parallel treebanks we described above, i.e. manually annotated treebanks and for use in machine translation. In the first, the best evaluation criteria would be some measurement of how much time annotators will use on completing the annotation. This will seldom be a realistic way of testing a system. More realistically, one could use some held-out hand-annotated data and measure for instance edit distance, or

even simpler, count the number of errors in the treebank.

For machine translation the goal is good translations, and the treebanks should be evaluated with respect to this. This may not necessarily correspond with the standard evaluation metrics for parsing and alignment.

We will later describe in more detail how we evaluate the treebanks.

1.6 Thesis Outline

The thesis is structured as follows:

Chapter 2 describes the background of the work we do. This includes machine learning, more specifically linear classification, dependency parsing and alignment. We also discuss work in different areas that we believe is related to our work.

Chapter 3 introduces the data we use, how to evaluate our approaches, and the basic tools we use in the experiments.

Chapter 4 introduces bilingually informed parsing. We argue why this approach should work and analyze the data. The errors of this approach are analyzed and new features are introduced as a result of the analysis.

Chapter 5 presents two approaches that model the different structures in the treebank jointly. We introduce an iterative approach that is based on the approach from the previous chapter and a reranking approach.

Chapter 6 describes further experiments. We evaluate the approaches from chapters 4 and 5 on another language pair, and discuss how the approaches can be used in SMT.

Chapter 7 presents results on evaluation data, discussion of the approaches, and future directions of the work on creating parallel treebanks.

Chapter 2

Background

2.1 Machine Learning

In this section we will give an overview of the machine learning methods used in the different experiments described in this thesis.

The focus will be on discriminative learning. The only place non-discriminative learning, i.e. generative, is used, is in the experiments where GIZA++ is used, and as this is used as an off-the-shelf tool, we will not discuss this.

2.1.1 Linear Classification

All of the methods used here are instances of linear classification. Linear classification is often used in NLP because the Zipfian distribution found in linguistic data makes the number of features needed to get good results very high. This makes it necessary to use methods that are fast to learn and fast to apply, and here linear classification fits in nicely. Examples of non-linear methods are decision trees, nearest neighbor algorithms, artificial neural networks (with at least one hidden layer).

Notation

To describe linear classification we first need to introduce some notation[1] .

[1] The notation and description used here is heavily inspired by slides by Ryan McDonald (2009)

We will use $\mathbf{x} \in X$ to denote the input to the classifier and $\mathbf{y} \in Y$ to denote the output. The input could for instance be a document with some words $w_1 \dots w_n$ such that $\mathbf{x} = w_1 \dots w_n$. The output could be some label describing the type of document or some structured output such as a tree or a sequence.

We will assume that the input and possible output is always mapped using an existing mapping into a (high-dimensional) feature vector:

$$\mathbf{f}(\mathbf{x}, \mathbf{y}) : X \times Y \to \mathbb{R}^m$$

In binary classification we can map from the input only into the feature space:

$$\mathbf{f}(\mathbf{x}, \mathbf{y}) : X \to \mathbb{R}^m$$

We often use multi-class classification and one way of handling this is to include the label into the feature vector using so-called block notation.

If for instance we have two features f_1 and f_2, and three labels A, B, C and the following training data:

	f_1	f_2	label
$\mathbf{x_1}$	0	1	A
$\mathbf{x_2}$	1	1	B
$\mathbf{x_3}$	1	0	C

This will lead to the following three feature vectors (the first row shows the combination leading to the feature value):

	A:f_1	A:f_2	B:f_1	B:f_2	C:f_1	C:f_2
$\mathbf{x_1}$	0	1	0	0	0	0
$\mathbf{x_2}$	0	0	1	1	0	0
$\mathbf{x_3}$	0	0	0	0	1	0

Classification

We can now turn to the actual classification, using linear classifiers.

The score of a classification is given by a linear combination of feature values and their weights. If we let $\mathbf{w} \in \mathbb{R}^m$ be a given weight vector, then

for multi-class classification where $Y = \{0, 1 \ldots N\}$ the output of a linear classifier is found as follows:

$$\mathbf{y} = \arg\max_{\mathbf{y}} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y})$$

In binary linear classification the weight vector represents (is the norm of) a

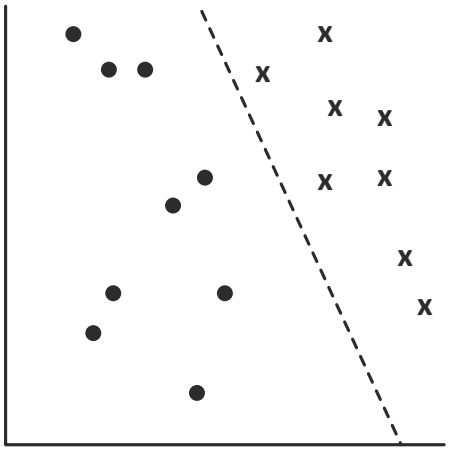

Figure 2.1: Example of separating hyperplane (line) that discriminates between the two classes.

hyperplane that discriminates between the two classes. Data points that are located on one side of the plane belong to one class, data points on the other side belong to another class. This is also called a separating hyperplane. Figure 2.1 shows an example of a separating hyperplane (which in two dimensions is a line).

Learning

We will now turn to how the weights in the weight vector $\mathbf{w}$ can be learned.

As we are considering supervised methods only, we assume training instances:

$$T = \{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1}^{|T|}$$

We also need a feature representation $\mathbf{f}$ that maps the input into feature vectors.

The task of the learning algorithm is then to output the optimal weight vector. How *optimal* is defined depends on, which learning algorithm is used. Here we consider algorithms that minimize error on the training data, however algorithms that instead maximize the likelihood of the data are also commonly used (e.g. Logistic regression, Naive Bayes).

2.1.2 Learning Algorithms

In this section we will describe a number of learning algorithms. All of them are algorithms used in the following chapters or algorithms that are considered important in order to understand the ones used.

Perceptron

The perceptron algorithm is probably the simplest and fastest linear classification learning algorithm. The perceptron algorithm tries to find a weight vector that minimizes error on the training data. If the data is linearly separable the perceptron guarantees convergence to a separating hyperplane.

The algorithm is an online algorithm. This means that it treats one input example at a time. Online algorithms have the advantage that it is not necessary to keep all of the input data in memory at once. Typically, online algorithms are trained by iterating over the training data more than once, which makes the training time trivially linear if a predefined number of iterations is used.

Algorithm 1 shows the perceptron learning algorithm. For each training example it updates the weight vector, if the example is classified wrongly. The change to the weight vector is the difference between the feature vector representing the correct label and the feature vector representing the incorrect output. In this way, weights for features present in the correct example, but not in the incorrect, will be increased, and weights for features present in the incorrect example, but not in the correct, will be decreased.

Algorithm 1: Perceptron learning

Data: N,$T = \{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1}^{|T|}$
Result: $\mathbf{w}^{(i)}$
$\mathbf{w}^{(0)} \leftarrow 0$; $i \leftarrow 0$;
for $n \leftarrow 0$ **to** N **do**
  for $t \leftarrow t$ **to** T **do**
    $\mathbf{y}' \leftarrow \arg\max_{y'} \mathbf{w}^{(i)} \cdot \mathbf{f}(\mathbf{x}_t, \mathbf{y}')$;
    if $\mathbf{y}' \neq \mathbf{y}_t$ **then**
      $\mathbf{w}^{(i+1)} \leftarrow \mathbf{w}^{(i)} + \mathbf{f}(\mathbf{x}_t, \mathbf{y}_t) - \mathbf{f}(\mathbf{x}_t, \mathbf{y}')$;
      $i \leftarrow i + 1$
    end
  end
end

Averaged Perceptron

The perceptron algorithm has a big risk of overfitting - especially to the last examples in the training data because the last updates are based on these. The voted perceptron is a variant of the perceptron that addresses this. It works by keeping a copy of the weight vector after each considered training example. When applied, all these vectors 'vote' on the correct classification of the input example. This method reduces the risk of overfitting and has certain margin guarantees (Freund and Schapire, 1999). The problem with the voted perceptron is that it requires that all the weights vectors are stored.

An approximation to the voted perceptron is the averaged perceptron (Collins, 2002). Here the final weight vector is found by averaging the weight vectors obtained after each considered training example. In this way all the weight vectors will have an influence on the final weight vector and the risk of overfitting is reduced.

Large Margin Classification

Perceptron is a very simple and efficient algorithm but there are learning algorithms that both in theory and practice provide better results. The prob-

lem with perceptron learning is, that although it is guaranteed to find a separating hyperplane (if it exists), there is no guarantee that it will find the best or even a good separating hyperplane.

Figure 2.2 shows two different linear separations of the same data. Intuitively, the line at the right is a better separator. The reason for this is that the distance between the data points and the separating line is larger than in the left case. The distance between the data points and the separating hyperplane is called the margin and large margin classifiers are classifiers that try to maximize this. Both in theory and practice large margins classifiers provide better results than the perceptron.

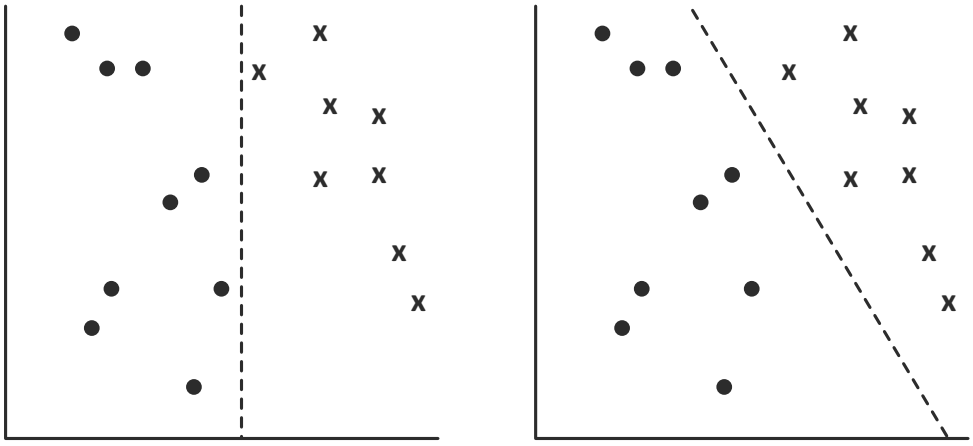

Figure 2.2: Two different separating lines for the same data set. The line on the right has a larger margin than the one on the left.

The most commonly used large margin classifiers are Support Vector Machines (Cortes and Vapnik, 1995). A Support Vector Machine is a batch learning algorithm that finds a separating hyperplane with the maximum possible margin.

As mentioned above, online algorithms are often faster[2] and require less memory than batch algorithms. This makes them especially interesting in NLP where large data sets with a huge amount of features are often used. Therefore we will now look at an online large margin algorithm.

[2]Linear SVMs can be trained in linear time (Joachims, 2006), so in theory we cannot have algorithms faster than this.

MIRA/PA

Crammer and Singer (2003) present a learning algorithm for online large margin multi-class classification called MIRA (Margin Infused Relaxed Algorithm). In the binary case this algorithm is the same as the simplest version of the PA (Passive-Aggressive) learning algorithm (Crammer et al., 2006). The difference between the simple PA algorithm and the more complicated PA-I and PA-II is the ability to deal with non-separability. Because we focus on the algorithm designed for the separable case we will use the term MIRA even though we initially focus on binary classification.

MIRA is a large margin algorithm because it tries to maintain a margin of a least $\frac{1}{2}$. Actually, the algorithm enforces this, so that after each update, if the example was classified incorrectly, there is a margin of at least $\frac{1}{2}$. This is what makes the algorithm aggressive - no matter how much the weight vector needs to be changed to achieve this margin, it is done. If the example was classified correctly and the margin is $\frac{1}{2}$ or more no update is made - it is passive. Simply changing the weight vector so that there is a margin of $\frac{1}{2}$ would result in a weight vector that changes a lot after each update, and would guarantee only good performance on the latest input. Therefore the algorithm changes the margin as little as possible to achieve a margin of $\frac{1}{2}$. The quadratic optimization problem in the algorithm can be solved using standard methods.

Algorithm 2 shows the MIRA learning algorithm. We define $\overline{Y}_t = Y \setminus \{\mathbf{y}_t\}$, i.e. the set of incorrect predictions. In the binary case the size of this set is always 1. $L(\mathbf{y}, \mathbf{y}')$ is a loss-function that defines the penalty for an incorrect prediction. In classification a $0/1$ loss is often used - i.e. there is no penalty if the label is correct and the penalty is 1 if it is incorrect. We see that the overall structure is the same as the perceptron algorithm. The only difference is the update. The constraint requires that the distance between the two data points when projected onto the weight vector should be more than the loss. If a $0/1$-loss is used, this means that the distance should be at least 1 if the example is classified incorrectly. This is equivalent to a margin of $\frac{1}{2}$.

Algorithm 2: MIRA learning

Data: N,$T = \{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1}^{|T|}$
Result: $\mathbf{w}^{(i)}$
$\mathbf{w}^{(0)} \leftarrow 0$; $i \leftarrow 0$;
for $n \leftarrow 0$ **to** N **do**
  for $t \leftarrow t$ **to** T **do**
    $\mathbf{w}^{(i+1)} \leftarrow \arg\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}^{(i+1)} - \mathbf{w}^{(i)}||^2$
      s.t. $\mathbf{w} \cdot \mathbf{f}(\mathbf{x}_t, \mathbf{y}_t) - \mathbf{w} \cdot \mathbf{f}(\mathbf{x}_t, \mathbf{y}') > L(\mathbf{y}, \mathbf{y}') \quad \forall \mathbf{y}' \in \overline{Y}_t$;
    $i \leftarrow i + 1$
  end
end

Confidence-Weighted Classification

Dredze, Crammer, and Pereira (2008) introduce confidence-weighted (CW) linear classifiers, which are online classifiers that maintain a confidence parameter for each weight and use this to control how to change the weights in each update. A problem with online algorithms is that because they have no memory of previously seen examples, they do not know if a given weight has been updated many times or few times. If a weight has been updated many times, the current estimation of the weight is probably relatively good and therefore should not be changed too much. On the other hand if it has never been updated before, the estimation is probably very poor. CW classification deals with this by having a confidence-parameter for each weight, modeled by a Gaussian distribution, and this parameter is used to make more aggressive updates on weights with lower confidence (Dredze, Crammer, and Pereira, 2008). The classifiers also use Passive-Aggressive updates (Crammer et al., 2006) to try to maximize the margin between positive and negative training instances.

CW classifiers are online algorithms and are therefore fast to train, and it is not necessary to keep all training examples in memory. Despite this they perform as well or better than SVMs (Dredze, Crammer, and Pereira, 2008). Crammer, Dredze, and Kulesza (2009) extend the approach to multi-class classification and show that also in this setting the classifiers often

outperform SVMs. They show that updating only the weights of the best of the wrongly classified classes yields the best results. We also use this approach, called top-1, here.

Crammer, Dredze, and Pereira (2008) present different update-rules for CW classification and show that the ones based on standard deviation rather than variance yield the best results. Our experiments have confirmed this, so in all experiments the update-rule from equation 10 (Crammer, Dredze, and Pereira, 2008) is used.

2.1.3 Structured Prediction

Above we looked at (binary) classification. However, for both syntactic parsing and alignment, the output from a system is a structured variable. Given the input, one sentence in parsing and two sentences in alignment, we want to predict either a syntactic structure or an alignment that contains some internal structure. In both cases there will only be a finite number of structures possible. This means that this could in principle be treated as multi-class classification, but in practice the number of possible output makes this impossible.

We will now look at two different ways of dealing with structured prediction. Both have been used in both parsing and alignment.

Factorization

In this section we will look at how it is possible to use some of the classification methods described above for structured prediction. We will assume that a structured hypothesis can be represented by a feature vector like in standard classification.

If we look at the perceptron algorithm, Algorithm 1, the following line is the one that is problematic with respect to structured prediction:

$$\mathbf{y}' \leftarrow \arg\max_{\mathbf{y}'} \mathbf{w}^{(i)} \cdot \mathbf{f}(\mathbf{x}_t, \mathbf{y}')$$

Given the feature representation we have chosen, and the current weight vector, we need to find the best scoring hypothesis. In standard classification we can simply enumerate the possible hypotheses, calculate the scores

and pick the best one. In structured classification this is not feasible. Instead we need to pick a feature representation that makes it possible to solve the $\arg\max$ without having to enumerate all the solutions. This typically requires some kind of factorization, meaning that the features must be defined in a way that makes the score of some sub-structure independent of the rest of the structure. In parsing and alignment we often represent the structure as a graph and use edge-factored models where the score of one edge in the graph is independent of the rest of the graph. In both parsing and alignment algorithms exist that can solve the $\arg\max$ problem when appropriate factorization is used, and we will describe these in more detail later. The conclusion with respect to the perceptron is, that if we can find the best scoring hypothesis, we can also use the perceptron algorithm for structured prediction.

We will now look at the MIRA algorithm. In the presentation above we focused on binary classification. However we can use the same formulation for multi-class and structured classification. The challenging part of the algorithm is the following line.:

$$\begin{aligned}\mathbf{w}^{(i+1)} \leftarrow \arg\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}^{(i+1)} - \mathbf{w}^{(i)}||^2 \\ \text{s.t. } \mathbf{w} \cdot \mathbf{f}(\mathbf{x}_t, \mathbf{y}_t) - \mathbf{w} \cdot \mathbf{f}(\mathbf{x}_t, \mathbf{y}') > L(\mathbf{y}, \mathbf{y}') \qquad \forall \mathbf{y}' \in \overline{Y}_t\end{aligned}$$

Again, we cannot enumerate the possible hypotheses.

There are two possible solutions to the problem. The first is to use factorization to split the constraint into a number of constraints - this is called *factored* MIRA. If we for instance do this for each edge in dependency parsing, there will be one constraint per possible edge, i.e. n^2 constraints. Using this makes the optimization problem feasible.

Another more widely used approach is to reduce the problem to multi-class classification by looking only at the k-best scoring hypotheses - this is called k-best MIRA. This requires an algorithm that can find the k-best solutions. Often $k = 1$ is used and then the requirement is the same as we saw for the perceptron algorithm - an algorithm (and factorization) that solves the $\arg\max$ problem.

Structured Prediction as Classification

Instead of trying to predict the entire structure in one step, we can split the prediction task into a number of smaller tasks, which are less complicated. For instance if we look at dependency parsing there are a number of possible relations. We can iterate through these and train, and later apply, a classifier that outputs whether or not there should be a relation between two tokens. This reduces the structured prediction task involved in dependency parsing to binary classification, which is easier to deal with than true structured prediction, where all of the structure is predicted in one step.

One of the advantages of this approach is that the problem of factorization is no longer present. At any stage in the classification process we can use whatever information is available. The disadvantage is that because every decision is basically local there is no guarantee that the optimal structure, given the model, is found. This problem can be reduced by using beam-search strategies, but when rich feature models are used the problem will always persist.

This approach is often used without too much consideration of the theoretical implications of doing structured prediction this way, but there is work investigating these. For instance Daume (2006) presents SEARN which is a more systematic approach for reducing structured prediction to classification.

This approach to structured prediction has been used in both parsing and alignment, and we will return to this later to describe exactly how.

The tools we use in this thesis primarily use graph-based methods so we will not describe the classification approach in further detail.

2.1.4 Reranking

Reranking is an approach often used in NLP and many different methods for doing this have been suggested. We will briefly discuss one of these methods for reranking, a method based on linear classification.

We follow Joachims (2002) in the following description of the (re)ranking task.

Given a list of input examples $x_1 \dots x_n$ the task of ranking consists of finding an ordering r of these. r is a binary relation over $X \times X$, where

$X = x_1, x_2, \ldots x_n$ so that if x_i is ranked higher than x_j then $(x_i, x_j) \in r$. We assume a strict ordering so either $(x_i, x_j) \in r$ or $(x_j, x_i) \in r$.

In reranking $x_1, x_2, \ldots x_n$ will be different solutions to the same problem, for instance a list of possible parses for a given input, and the task will be to rank these possible parses.

If r^* is the correct ordering, the learning task consist in finding an ordering r_f that is as similar as possible to r^*. The similarity measure used by Joachims (2002) is Kendall's τ, which can be defined as:

$$\tau(r_a, r_b) = \frac{P - Q}{P + Q} = 1 - \frac{2Q}{\binom{m}{2}}$$

where P is the number of concordant pairs, and Q is disconcordant pairs. If $(x_i, x_j) \in r_a$ and $(x_i, x_j) \notin r_b$ then the pair is disconcordant.

This means that given a training set S with m training examples containing an input list $\mathbf{x}$ and the target ranking r^*:

$$(\mathbf{x}_1, r_1^*), (\mathbf{x}_2, r_2^*), \ldots (\mathbf{x}_m, r_m^*)$$

the task of the learner is to find a ranking function f that maximizes the empirical τ

$$\tau_s(f) = \frac{1}{m} \sum_{i=1}^{m} \tau(r_{f(x_i)}, r_i^*)$$

on the training set.

Ranking can be solved using linear classifiers. Given a mapping ϕ from an input example to a feature vector, the following are the class of linear ranking functions:

$$(x_i, x_j) \in f_{\mathbf{w}}(x) \Leftrightarrow \mathbf{w}\phi(x_i) > \mathbf{w}\phi(x_j)$$

Given the description above, Joachims (2002) formulates an optimization problem that makes is possible to solve the ranking problem as a SVM-optimization problem.

Joachims (2006) presents a more efficient formulation of the problem.

2.2 Dependency Parsing

Natural language parsing is the task of automatically producing syntactic analyses for natural language sentences. Here we focus on dependency

parsing, which means that we will be producing dependency grammar analyses. We will not go into the subtleties of different kinds of dependency grammars, which are well described several places (Nivre, 2006; Kübler, McDonald, and Nivre, 2009). In the following section we will formally define the dependency analyses that we will work with.

2.2.1 Formal Definition

Here we will formally define dependency graphs and dependency parsing. This will also allow us to define exactly the structures we consider in this work. The description and definition of the problem is adapted from (Kübler, McDonald, and Nivre, 2009).

Dependency Trees

First we define a sentence S.

Definition 1. $S = w_0 w_1 \dots w_n$

We assume that the tokenization is done before the parsing step, and we will not discuss tokenization. w_0 is an artificial root-token inserted in the beginning of every sentence.

Definition 2. *Let $R = \{r_1, \dots, r_n\}$ be a set of possible* dependency relation types *that can hold between any two words in a sentence. A relation type $r \in R$ is additionally called an* arc label.

Now we can define dependency graphs

Definition 3. *A* dependency graph $G = (V, A)$ *is a labeled directed graph in the standard graph-theoretic sense and consists of nodes, V, and arcs, A, such that for sentence $S = w_0 w_1 \dots w_n$ and label set R the following holds:*

1. $V \subseteq \{w_0, w_1, \dots, w_n\}$
2. $A \subseteq V \times R \times V$
3. *if* $(w_i, r, w_j) \in A$ *then* $(w_i, r', w_j) \notin A$ *for all* $r' \neq r$

This definition allows cycles in the analysis and that a token can have more than one head (a vertex having more than one incoming edge). Cycles and more than one head for a token are allowed in some dependency formalisms. For instance figure 2.3 shows an analysis from the Copenhagen Danish-English Dependency Treebank (Buch-Kromann, Wedekind, and Elming, 2007). Here the token "you" has more than one head. Although

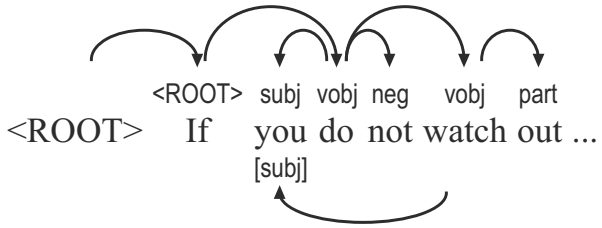

Figure 2.3: Analysis from CDT that uses secondary dependencies, which can introduce cycles into the structure.

parsing methods that can construct parses like this exists (McDonald and Pereira, 2006; Sagae and Tsujii, 2008) we will limit our focus to dependency *trees*.

Definition 4. *A* well-formed dependency graph $G = (V, A)$ *for an input sentence* S *and dependency relation set* R *is any dependency graph that is a* directed tree originating out of node w_0 *and has the spanning node set* $V = V_s$*. We call such dependency graphs* dependency trees*.*

Kübler, McDonald, and Nivre (2009) show that a dependency tree satisfies both the *single-head property*, excluding the analysis in example 2.3 from the set of well-formed dependency graphs and also the *acyclicity property*. Another property of dependency graphs is whether or not they are projective. Again we need some definitions.

Definition 5. *An arc* $(w_i, r, w_j) \in A$ *in a dependency tree* $G = (V, A)$ *is* projective *if and only if* $w_i \rightarrow^* w_k$ *for all* $i < k < j$ *when* $i < j$*, or* $j < k < i$ *when* $j < i$*.*

where $w_i \rightarrow^* w_k$ indicates the reflexive transitive closure of the dependency relation.

Definition 6. *A dependency tree is a projective dependency tree if it is a dependency tree and all* $(w_i, r, w_j) \in A$ *are projective.*

Definition 7. *A dependency tree is non-projective if it is a dependency tree and it is not projective.*

Figure 2.4 shows an example of a non-projective tree.

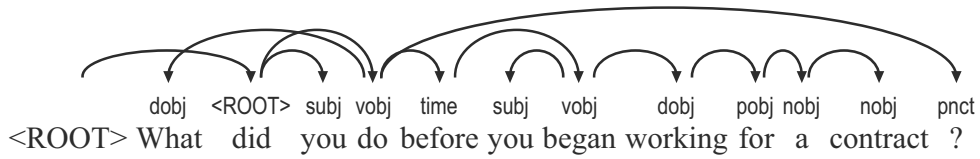

Figure 2.4: Non-projective structure from CDT.

Dependency Parsing

Dependency parsing is often divided into two classes, called *data-driven* parsing and *grammar-based* parsing. Kübler, McDonald, and Nivre (2009) describe data-driven approaches as being approaches that make essential use of *machine learning* from linguistic data to parse new sentences. Grammar-based approaches on the other hand rely on *formal grammars*. They also note that the two approaches are not mutually exclusive. A parser can be based on a formal grammar *and* use machine learning. In the work presented here we use only purely data-driven methods. Furthermore we focus solely on *supervised* methods. This means that we always have some annotated data available to learn from. We will only learn from this data, i.e. we will not use semi-supervised methods.

Now we turn to a formal definition of dependency parsing. Again we follow Kübler, McDonald, and Nivre (2009).

Definition 8. *A* dependency parsing model *consists of a set of constraints* Γ *that define the space of permissible dependency structures for a given sentence, a set of parameters* θ*, and a fixed parsing algorithm* h*. A model is denoted by* $M = (\Gamma, \theta, h)$*.*

In data-driven parsing there are two phases. A *learning* phase where the parameters are learned from annotated data. What exactly these parameters are depends on the learning method and parsing method used. The other phase is the *parsing* phase where the learned model is applied to new

sentences. Here the objective is to find the most likely dependency tree, given the constraints and the parameters.

2.2.2 Graph-Based

We have seen that a dependency tree is a directed tree that spans all the tokens of a sentence. This implies that spanning tree algorithms from graph theory research can be used as parsing algorithms. A lot of work has been done on this type of parsing. Here we will focus primarily on the work presented in (McDonald, Crammer, and Pereira, 2005; McDonald et al., 2005; McDonald and Pereira, 2006; McDonald, Lerman, and Pereira, 2006). We call the parser presented in these the *MSTParser*[3]. In graph-based parsing scores over possibly trees are defined, and the job of the parsing algorithm is to find the tree with the best score, given the model. These scores can be defined in different ways, and here we will primarily discuss so-called *arc-factored* models.

Arc-Factored

McDonald et al. (2005) define the score of a dependency tree as the sum of the scores of the individual arcs in the tree. In graph-theoretic terms the arcs would be called edges. Furthermore the score is defined as the dot-product between a high-dimensional feature representation of the edge and a weight vector. This means that the score s of an arc (w_i, r, w_j) is given by

$$s(w_i, r, w_j) = \mathbf{w} \cdot \mathbf{f}(w_i, r, w_j)$$

if we do unlabeled parsing the r is just left out of this score. The important thing is that the feature function is arc-factored. The means that it is defined so that the score of one edge is not dependent on the the rest of the structure.

The score of a dependency tree $G = (V, A)$ for a sentence S is:

$$s(G, S) = \sum_{(w_i, r, w_j) \in A} s(w_i, r, w_j) = \sum_{(w_i, r, w_j) \in A} \mathbf{w} \cdot \mathbf{f}(w_i, r, w_j)$$

[3]There are differences in the parsers from the different papers, but for now we will ignore these and treat them as one parser.

Parsing

Above we described parsing with graph-based methods as the task of finding the tree with the highest score given the model. In the setting described above, the model is given by the weight vector. So, given the weight vector, we need to find the highest scoring tree. As we have seen, a well-formed dependency graph is a spanning tree, and therefore an algorithm that finds the spanning tree that has the highest score solves the parsing problem. This problem can be solved using the Chu-Liu-Edmonds algorithm (McDonald et al., 2005; Chu and Liu, 1965; Edmonds, 1967; Georgiadis, 2003). We will not describe this algorithm, but just note that it has the nice property that for dense graphs an implementation with run-time $O(n^2)$ exists (Tarjan, 1977). The Chi-Liu-Edmonds algorithm will output the highest scoring tree, which means that there are no restrictions on this tree. Therefore this algorithm is not optimal if only projective trees are allowed. In this case the Eisner-algorithm can be used instead (Eisner, 1996). The Eisner-algorithm outputs the highest scoring projective spanning tree and has a run-time of $O(n^3)$.

Learning

The MSTParser uses MIRA for learning. We have described MIRA for structured prediction in sections 2.1.2 and 2.1.3 so here we will focus on how it is integrated with the parser.

During training, one training example is parsed as described above. The output of the parsing algorithm is then compared to the gold-standard training example, and the weight vector is updated (unless the parse is correct). Then the next example is parsed with the new weight vector. This continues for all training examples for a given number of iterations. At the end the weights are averaged as described in section 2.1.2.

The loss used in the MSTParser is the number of dependents that either gets the wrong head or the wrong label (if we are performing labeled parsing). We will return to the question of how to evaluate dependency parsing, but for now we will note that this loss corresponds to Labeled Attachment Score, so the parser optimizes the measure commonly used to evaluate dependency parsers.

Two-Stage Parsing

In the setting described above, the parser produces the structure of the parse and the labels of the arcs at the same time[4]. The graph representing the input to the inference algorithm is a multi-graph as there are multiple edges (arcs) between each vertex (word).

McDonald, Lerman, and Pereira (2006) present a two-stage dependency parser. This parser initially performs unlabeled parsing. It then uses a second stage to label the arcs produced by the first stage. One advantage of this is that non-arc-factored features can be used in the second stage.

Second-Order

From a linguistic point of view edge-factored models seem very implausible. The factorization means that the score for a given edge is independent from the rest of the syntactic structure produced. For instance the likelihood of something being a subject in the sentence is independent of whether another subject will also be constructed.

To remedy this problem one can use models where information about other edges can be used in determining the score of an edge. McDonald and Pereira (2006) show how to perform exact projective second-order MST-parsing. Exact second-order non-projective parsing is shown to be NP-hard so an approximate approach to this is introduced. First a projective tree is constructed and following this, changes that increase the overall score of the tree is searched for and applied in a hill-climbing fashion. As the projective tree is guaranteed to be the best projective tree, any changes to this that increase the score, will lead to a non-projective tree. McDonald and Pereira (2006) argue that this approach is reasonable because even in languages with many non-projective arcs, the trees are mainly projective.

Higher-Order

It is possible to go beyond second-order parsing. Koo and Collins (2010) present exact third-order projective parsing, that runs in $O(n^4)$. Higher-order exact non-projective parsing is, as mentioned, NP-hard, but approx-

[4]This is also how the open-source implementation of the MSTParser works.

imate methods that yields good results exist (Smith and Eisner, 2008; Koo et al., 2010). We will not use these methods in the work presented here.

Reranking

Another approach to including higher-order features is to produce a k-best list of parses using a lower-order model, and then rerank these using higher-order features. By doing this, features over all of the structure can be used.

Hall, Havelka, and Smith (2007) introduce this approach for non-projective dependency parsing[5]. To find the k-best maximum spanning trees an algorithm proposed by Camerini, Fratta, and Maffioli (1980) is considered. Using the right data-structures this algorithm runs in $O(kn^2)$ for dependency parsing (dense graphs). The output from this algorithm is then reranked, using higher-order features.

2.2.3 Structured Prediction as Classification

Another approach to dependency parsing is called transition-based parsing. In this approach the structured prediction task is reduced to classification. We will primarily use graph-based parsing but we will shortly describe transition-based parsing as this is used in a few experiments.

There are different variants of transition-based parsing, with different transitions, different search strategies and different learning algorithms. We restrict ourselves to describing the variant of parsers described by Nivre (2008).

Transition-based parsing builds on the idea that parsing can be viewed as a sequence of transitions between states. A transition-based parser (deterministic classifier-based parser) consists of three essential components (Nivre, 2008):

1. A parsing algorithm
2. A feature model
3. A classifier

[5]For projective parsing the Eisner algorithm can be used.

We will not describe the feature model further, but instead look at the other two parts.

The parsing algorithm consists of two components, a *transition system* and an *oracle*. Nivre (2008) defines a transition system $S = (C, T, c_s, C_t)$ in the following way:

1. C is a set of configurations, each of which contains a buffer β of (remaining) nodes and a set A of dependency arcs,
2. T is a set of transitions, each of which is a partial function $t : C \rightarrow C$,
3. c_s is a initialization function mapping a sentence $x = (w_0, w_1, \ldots, w_n)$ to a configuration with $\beta = [1, \ldots, n]$,
4. C_t is a set of terminal configurations.

A *transition sequence* for a sentence x in S is a sequence $C_{0,m} = (c_0, c_1 \ldots, c_m)$ of configurations, such that

1. $c_0 = c_s(x)$,
2. $c_m \in C_t$,
3. for every i $(1 \leq i \leq m) c_i = t(c_{i-1})$ for some $t \in T$

The oracle is used during training to determine a transition sequence that leads to the correct parse. The job of the classifier is to 'imitate' the oracle, i.e. to try to always pick the transitions that lead to the correct parse. The information given to the classifier is the current configuration. Therefore the training data for the classifier consists of a number of configurations and the transitions the oracle chose with these configurations.

Here we focus on stack-based parsing algorithms. A stack-based configuration for a sentence $x = (w_0, w_1, \ldots, w_n)$ is a triple $c = (\sigma, \beta, A)$, where

1. σ is a stack of tokens $i \leq k$ (for some $k \leq n$),
2. β is a buffer of tokens $j > k$,
3. A is a set of dependency arcs such that $G = (0, 1, \ldots, n, A)$ is a dependency graph for x. (Nivre, 2008)

In the work presented here we use the NivreEager algorithm which has four transitions:

Shift Push the token at the head of the buffer onto the stack.

Reduce Pop the token on the top of the stack.

Left-Arc$_l$ Add to the analysis an arc with label l from the token at the head of the buffer to the token on the top of the stack, and push the buffer-token onto the stack.

Right-Arc$_l$ Add to the analysis an arc with label l from the token on the top of the stack to the token at the head of the buffer, and pop the stack.

Learning

The third part of the system is a classifier, and this is because transition-based dependency parsing reduces parsing to consecutive multi-class classification. From each configuration one amongst some predefined number of transitions has to be chosen. This means that any classifier can be plugged into the system. The training instances are created by the oracle, so the training is offline. This implies that any classification algorithm can be used.

2.3 Alignment

In NLP *alignment* is the task of aligning different linguistic entities - typically in two different languages. The two most common tasks are sentence alignment, where sentences with the same meaning in two or more languages are aligned, and word alignment where words with the same meaning are aligned. Another type of alignment is *sub-tree* alignment where the sub-trees in a syntactic tree in two (or more) languages are aligned.

Here we will only describe word alignment and sub-tree alignment. And as it turns out, the two are actually the same when we consider dependency structures. The nodes in a dependency tree are the words in the sentence, so aligning the sub-trees rooted in these nodes is equivalent to aligning the words.

2.3.1 Formal Definition

We will here formally define what we mean by word alignment in this work. We use the same definition of a sentence as we did in section 2.2.1, except we leave out the artificial root token.

Definition 9. $S = w_1, w_2 \dots w_n$

In alignment we have two sentences so we super-scribe both the S and the w, so that $S^a = w_1^a, w_2^a \dots w_n^b$. Now we can define a word alignment:

Definition 10. *A word alignment of two sentences S^a and S^b is an undirected bipartite graph $G = (V, A)$ consisting of the two disjoint set of nodes V^a and V^b such that the following holds:*

1. $V^a \subseteq \{w_1^a, w_2^a, \dots, w_n^a\}$
2. $V^b \subseteq \{w_1^b, w_2^b, \dots, w_m^b\}$
3. $V = V^a \cup V^b$
4. $A \subseteq V \times V$

The bipartite condition implies that $A \subseteq V_a \times V_b$, i.e. all edges are from vertices representing words from one sentence to vertices representing words from the other sentence.

Sometimes we will distinguish between *sure* and *possible* links. In this case we will instead need to define a *labeled* graph with the label set $R = \{\text{sure}, \text{possible}\}$ and then $A \subseteq V \times R \times V$

2.3.2 Graph Based

One way to try and create alignments is viewing the problem from a graph-theoretic point of view and using algorithms from this field to solve the problem. If we assume that each word or node in a sentence is represented by a vertex and links between words are edges, the graph will be bipartite, i.e. all edges connects a vertex from one sentence to a vertex from the other sentence. If all possible links are assigned a score (a weight in graph-theoretic terms) then the task of finding a maximum scoring alignment can be solved using different graph-algorithms, for instance a maximum matching algorithm. Figure 2.5 shows how such a graph would look for two sentences with three words each. The edges are possible links.

One approach using this method is presented by Taskar, Lacoste-Julien, and Klein (2005) who present a discriminative word aligner using a maximum matching approach. In the aligner they use linear programming to find the maximum matching as this combines well with their learning algorithm, but combinatorial algorithms, as they note, can also be used.

They use Max-Margin Markov Networks (Taskar, Guestrin, and Koller, 2003) for learning and achieve very good results. Especially when using the output from GIZA++ they get very low alignment error rates, and this underlines one of the advantages of discriminative learning - that features like this can easily be used.

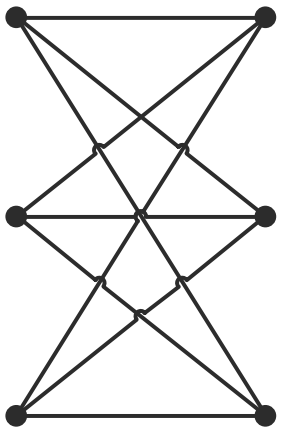

Figure 2.5: Graph representing a word alignment task. By doing maximum matching the highest scoring alignment can be found.

This approach requires edge-factored features. I.e. the score of one edge cannot be dependent on the rest of the structure. If the features are not edge-factored, the maximum matching algorithms cannot be used. A drawback of the approach is that it only allows 1-1, 0-1, 1-0 links.

Lacoste-Julien et al. (2006) present a word aligner that deals with this. The problem can be solved by viewing the alignment problem as a minimum cost maximum flow problem. In this, the problem is to find the maximum flow with the minimum cost through a network represented by a directed graph. The maximum matching problem can also be viewed this way by adding a so-called *source* and *sink* vertex. To allow fertility more

than 1, extra edges from the source to the source word vertices and extra edges from the target word vertices to the sink are added. Figure 2.6 shows how the initial graph would look with three words in both sentences and a maximum fertility of 3. It should be noted that compared to the graph shown in (Lacoste-Julien et al., 2006) our graph contains some extra intermediary fertility nodes. This is to keep to a simple graph, i.e. maximum one edge from one vertex to another.

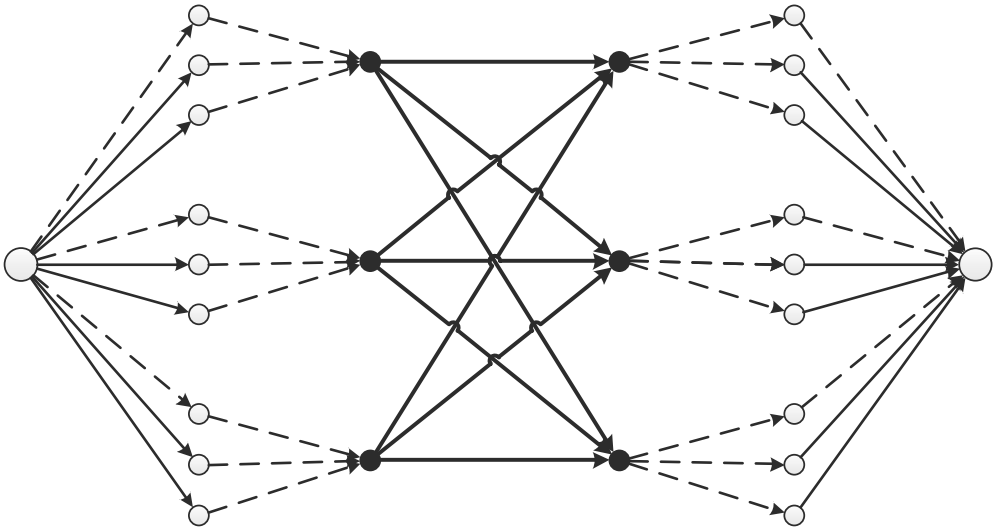

Figure 2.6: Directed graph representing a word alignment task with fertility higher than 1 (3 in this example). Solid dots represent the words. Dashed lines have no cost. The leftmost vertex is called the *source* and the rightmost the *sink*. The problem of finding a maximum scoring alignment can be solved using a minimum cost maximum flow algorithm.

As with the maximum matching problem, efficient algorithms exist to solve this problem. Apart from allowing fertility more than one, Lacoste-Julien et al. (2006) introduce second-order features. They solve the inference problem as a quadratic assignment problem and achieve very low alignment error rates on the Hansard corpus.

The results with this aligner are very good but the approach has one major drawback which is the training time. Solving the inference problem is quite slow but especially the learning algorithm is not feasible for larger

data sets (Daume, 2006). In the work mentioned training sets of 100 and 200 sentences are used.

2.4 Related Work

In this section we will discuss previous work related to bitext dependency parsing and alignment. We split the work into the following categories: projection, methods that use trees in alignment, methods that use alignments to improve parsing, methods that combine the two and try to solve the task jointly and grammar-based methods. We will discuss work from all categories starting with projection.

2.4.1 Projection

Projection is a task closely related to the task we are addressing here - i.e. using bilingual information to create better parallel treebanks. In projection an analysis exists on one language and this analysis is projected to another language. Often this is seen as a solution to help create resources on languages with few resources, by using resources from a language with many resources. If, for instance, analyses could be projected from English to another language with a certain quality, it would be much easier to create large-scale resources on this language. Our work is related to this because we try to use structure from one language to enrich the structure of another. We use two languages that have structures already, but many of the assumptions and problems we face are the same as in projection. In some sense we project the analysis from the target language to the source language and use this projected structure to inform the source side parsing.

The literature on projection is vast and we will only consider a very small part of this.

Hwa et al. (2002) present work on projection from English to Chinese. They address an assumption which to some degree underlies all work on projection and bilingually informed parsing. They call this the Direct Correspondence Assumption:

> **Direct Correspondence Assumption (DCA):**
> Given a pair of sentences E and F that are (literal) translations of each other with syntactic structures $Tree_E$ and $Tree_F$, if nodes x_E and y_E of $Tree_E$ are aligned with nodes x_F and y_F of $Tree_F$, respectively, and if syntactic relationship $R(x_E; y_E)$ holds in $TreeE$, then $R(x_F; y_F)$ holds in $TreeF$.

Hwa et al. (2002) show that using an algorithm for doing direct projection they achieve a F_1-score of only 38.1 when projecting from English to Chinese. With relatively simple hand-crafted rules they are able to increase this number to 67.3. This shows the need for working with more complex correspondence patterns than simple direct correspondence.

The experiments were performed on fully hand-aligned data, i.e. with hand-aligned alignments and hand-annotated syntax, although the Chinese tree were obtained by automatically converting phrase-structure trees to dependency trees.

We will work with related languages which makes the work of Zeman and Resnik (2008) interesting. Zeman and Resnik (2008) investigate parser projection between two closely related languages, Danish and Swedish. To test their approach they need to normalize the Danish treebank[6] they use and the Swedish treebank they use because of the difference in annotation, tag-sets and so on. Furthermore, because they use phrase-structure parsers, they convert the dependency trees to phrase-structure trees. Using an automatically aligned corpus for Danish and Swedish, the most probable translation of each word in isolation is found. When parsing a Swedish sentence, the sentence is 'translated' into Danish using these and then a parser trained on Danish is used to parse the sentence. In this way parses are projected from Danish to Swedish. Another approach tested is to delexicalize the sentences.

The best result achieved is a F_1-score of 66.40. Given the closely related languages this is surprisingly low compared to the results we saw for Chinese and English, using a simpler approach. We believe that one of the reasons for this is the amount of mapping going on. Both with the normalization of the treebanks and with the dependency to phrase-structure

[6]Which is the same treebank we use.

conversion.

In recent work, McDonald, Petrov, and Hall (2011) present a method for projecting from multiple languages. The method essentially relies on delexicalization in the same way as the work of Zeman and Resnik (2008). A universal tag-set (Petrov, Das, and McDonald, 2011) projected from English data (Das and Petrov, 2011) is used, which makes it very simple to use delexicalized parsers trained on one language on other languages. Multiple languages can be used as the source of the projection by simply concatenating the data on these languages and then training the parsers on this. Unlike other work (Søgaard, 2011) this concatenation is found to improve accuracy.

This simple approach does not rely on alignment at all. McDonald, Petrov, and Hall (2011) also present a more advanced approach that does. First parses are projected using the delixicalization approach. These parses are then used as gold-standard parses to train a target-side monolingual parser. The model from this parser is used to seed a parser that is trained using alignment information. This parser is trained using the framework presented by Hall et al. (2011). When training this parser an external metric of 'good' parses is used. This is obtained by creating a k-best list of parses for each example. The parse in this list that is most parallel (based on fixed alignments) with the parse of an English parallel sentence is considered the best and used as the gold-standard in the external metric.

The approach provides very good results and has the advantage that it does not rely on gold-standard PoS-tags.

2.4.2 Unsupervised Sub-Tree Alignment

To create parallel treebanks several researchers has suggested doing subtree alignment on existing bitext treebanks using the output from an unsupervised word aligner. Samuelsson and Volk (2007) does this in the creation of the SMULTRON parallel treebank (Volk et al., 2010). They do word alignment and create phrase-tables using existing tools for this (GIZA++, THOT, PHARAOH). Then they search for phrases that are consistent with the trees on the two languages and use these as phrase alignments. They report a $F_{0.5}$-score of up to 65% on Swedish-English with this approach.

Zhechev and Way (2008) also present work that departs in the unsupervised word alignments created with tools like GIZA++. They present methods for both creating parallel treebanks from bitexts and from parsed bitexts. We will focus on the latter.

The translation probabilities from the word aligner are used to score a given phrase alignment. For two trees S and T spanning $1 \dots m$ and $1 \dots n$ the phrase alignment $\langle s, t \rangle$ spanning $\langle s_i \dots s_{ix} \rangle$ and $\langle t_j \dots t_{jy} \rangle$ the score is given as the product of an inside score and an outside score in each translation direction. *Inside* are the tokens inside the phrases being aligned and *outside* are the tokens outside as described in (1).

(1)

	inside	*outside*
	$s_l = \langle s_i \dots s_{ix} \rangle$	$\bar{s}_l = \langle S_1 \dots s_{i-1} s_{ix+1} \dots S_m \rangle$
	$t_l = \langle t_j \dots t_{jy} \rangle$	$\bar{t}_l = \langle T_1 \dots t_{j-1} t_{jy+1} \dots T_n \rangle$

(2) $$\gamma(\langle s, t \rangle) = \alpha(s_l|t_l) \cdot \alpha(t_l|s_l) \cdot \alpha(\bar{s}_l|\bar{t}_l) \cdot \alpha(\bar{t}_l|\bar{s}_l)$$

(3) $$\alpha(x|y) = \prod_i^{|x|} \sum_j^{|y|} \frac{P(x_i|y_j)}{|y|}$$

The scores are defined in (2) and (3) and assume that translation probabilities $P(x|y)$ exist.

With these scores in place, the alignment is a matter of search. Because exhaustive search is prohibitly slow, Zhechev and Way (2008) suggest a number of greedy search algorithms. The algorithms conform to some restrictions which are that a node may only be aligned to one other node and that descendant/ancestors of a source linked node may only be linked to descendants/ancestors of its linked counterpart. In intrinsic evaluation the method yields precision just above 60% and recall just above 80% when evaluated on the HomeCentre treebank (Hearne and Way, 2006). An extrinsic evaluation is done, where the Data Oriented Translation system (Hearne and Way, 2006) is trained on the treebanks and translation quality is measured with different standard machine translation metrics. We will not go into these in detail, but note one interesting fact, namely that the quality of the translations does not match the scores in the intrinsic evaluation. Actually, the automatically created treebanks lead to better translations than the hand-aligned. This leads the authors to conclude, that the improvements

of the aligner should not be aimed at increasing recall and precision compared to the manual alignment, but instead directly at improving translation quality.

2.4.3 Supervised Sub-Tree Alignment

In the section above we described work that uses word alignment tools to induce alignments in an unsupervised fashion. Now we turn to work that learns from existing alignments of parsed text. The advantage of this approach is that higher accuracy can be expected as supervised learning methods can be used. The disadvantage is that an existing parallel treebank is necessary.

Tiedemann and Kotzé (2009a; 2009b) describe an approach for supervised sub-tree alignment. Compared to the approaches to structured prediction we looked at earlier they initially separate the learning part and the inference part. For each possible link between sub-trees a number of features are extracted. A maximum entropy classifier is then used to score each of these links. Using a discriminative learner allows the inclusion of arbitrary features. For instance Tiedemann and Kotzé (2009a) use the output from GIZA++ as features. They also use inside-outside features but use the maximum translation probability of a span instead of the averaged. This means changing (3) from above to:

$$\text{(3b)} \quad \alpha(x|y) = \prod_i^{|x|} max_j P(x_i|y_j)$$

Additional features are used but we will not describe these in detail here.

At test time the search for the best alignment is performed greedily with a bottom-up approach. This allows the inclusion of non-factored features because decisions can be made on the basis of previously made decisions. Tiedemann and Kotzé (2009a) test on the SMULTRON corpus (Volk et al., 2010) and reports F-scores significantly above what they achieve with the approach described by Zhechev and Way (2008). On the English-Swedish part of the treebank they get $F_{0.5}$-scores of around 78.

Tiedemann (2010) presents LinguaAlign which is a publicly available version of the aligner described above. In LinguaAlign there are some additional search strategies available. One is doing a maximum-matching as

described in section 2.3.2 using the scores from the classifier as weights in the graph. Another addition is to introduce a SEARN-like learning scheme. None of these approaches improves accuracy compared to the standard greedy search strategy described by Tiedemann and Kotzé (2009a).

2.4.4 Bilingually Informed Monolingual Parsing

Huang, Jiang, and Liu (2009) describe an approach for monolingual dependency parsing where information from the same sentence in another language (a translation) is used to guide the parsing. The motivation is that doing actual bitext parsing either is prohibitively slow (see section 2.4.7) or requires heavy use of approximation (see section for instance 2.4.5 and 2.4.6) to deal with the huge search space involved treating two trees (and possible alignments) at once.

Instead Huang, Jiang, and Liu (2009) use a standard monolingual transition-based parser and add features extracted by automatically aligning the sentences with parallel sentences.

Huang, Jiang, and Liu (2009) argue and empirically show that most errors in transition-based parsing are caused by so-called shift-reduce conflicts. These are situations where the parser chooses a shift transition where a reduce transition would be correct or the other way around. Only three features templates (and combinations of these) are used, and it is interesting that they are linguistically motivated in that specific situations where a shift transition should be used instead of a reduce and vice versa are identified.

The run-time of a standard transition-based parser is linear, and if a beam of size k is used $O(kn)$. The introduction of the extra features increases this to $O(kn^2)$ but in practice it is only slightly slower than the monolingual parser. On English-Chinese data an absolute improve of around 0.5 in accuracy[7] is achieved.

Zhao et al. (2009) also use bilingually based features to improve monolingual parsing. The sentence to be parsed is translated into another language using a word-to-word model based on a dictionary. The word pairs

[7]The paper does not state whether this is unlabeled or labelled, but we assume that it is unlabelled as nothing else is stated and the referenced results (McDonald, Crammer, and Pereira, 2005) are unlabeled.

corresponding to potential dependency relations in the new language are looked up in a list of word pairs extracted from a treebank in that language, and features are extracted using this list.

Chen, Kazama, and Torisawa (2010) present a method for what they call "Bitext Dependency Parsing". We describe this as "bilingually informed monolingual parsing" because although they do parse both languages, the two models are independent of each other, although dependent on the other language. The method works by parsing a large amount of text on the target language. From this automatically parsed treebank a list of existing sub-trees are extracted and saved in a way that makes it fast to retrieve them. When parsing the source language, automatically created alignments are used together with learned mapping rules to map potential sub-trees (2 or 3 tokens) on the source side to the parallel text. The tree formed on the target side is then looked up in the list of sub-trees extracted from the automatically parsed corpus. If the tree exists this is indicative of a probable sub-tree. The same (without the mapping) is actually done for the source side using a parsed large corpus on the source side as described by Chen et al. (2009).

The method provides a more than 2.5 increase in F_1-score on both English and Chinese and has the advantage compared to the approaches we describe in sections 2.4.5 and 2.4.6, that a parallel treebank is not necessary. It is only necessary to have text parallel with the treebank used for training and a parser for the target language.

Chen et al. (2011) show that the same approach works if treebanks where the parallel sentences are obtained by using machine translation instead of human translation.

Smith and Eisner (2006) introduce *quasi-synchronous grammars*, which are a kind of parallel grammar that allows 'sloppy' alignment between the syntactic trees of the two sentences, i.e. they do not require the trees to be isomorphic. Smith and Eisner (2009) show that these grammars can be used for parser adaptation and parser projection. Parser adaptation is the task of adapting a parser to a new annotation-style. This task and parser projection is basically the same as one sentence is parsed with the analysis of a parallel sentence as input. In principal, the only difference is that alignment in parser adaptation is trivial.

When some data is available, this approach falls under what we call bilingually informed monolingual parsing (Smith and Eisner (2009) call it *supervised cross-lingual projection*). Smith and Eisner (2009) evaluate this on the LDC English-Chinese Parallel Treebank, and find that a bilingually informed parser trained on n sentences has roughly the same accuracy as a standard parser trained on $2n$ sentences.

2.4.5 Joint Parsing by Reranking

Burkett and Klein (2008) present a reranking approach for parsing bitext and creating word alignments. The reranked sentences come from two independently trained state-of-the art monolingual phrase-structure parsers.

Burkett and Klein (2008) use a maximum entropy model to model the joint probability of the trees on both languages and of the sub-tree alignment. As no gold-standard alignments exist, the alignments are treated as latent variables in the model. For different reasons, primarily the size of the search space, training the model directly is not feasible. Instead, an iterative approach is adopted. First an initial set of weights on alignment features are used to create a possible alignment of the trees. Then this alignment is fixed and used to optimize the parameters of the parsing features. Then the trees are fixed and the alignment features optimized and so forth.

Burkett and Klein (2008) use the English-Chinese translation treebank (Bies et al., 2007) and achieves a 2.5 F_1-score improvement on English trees and 1.8 F_1-score improvement on Chinese trees compared to the monolingual parsers. Furthermore they show a 2.4 BLEU improvement in a downstream MT evaluation using a syntactic MT system.

We will now shortly describe some of the features used in this work.

For the alignment, features similar to the inside-outside features described earlier are used. In addition hard versions of these are used, where the count of alignments from an external word aligner between two spans are used instead of the product (or max) of the probabilities from the word aligner. The inside-outside features are also scaled using the geometric means of span lengths.

For parsing the only monolingual features used are the posterior probabilities from the monolingual parsers. The difference in span length between the two spans dominated by the alignments is used as a feature.

Another feature indicated the number of children of the nodes in a possible link. Finally, a feature encoding the occurrence of label pairs of children of the nodes in the possible link are used. All features are also combined with the labels of the two nodes.

It is worth noticing that all alignment features pertain only to the two nodes in question, i.e. they are edge-factored, which allows the use of a maximum-matching algorithm to find an optimal alignment.

2.4.6 Joint Parsing and Alignment

Burkett, Blitzer, and Klein (2010) present an approach for joint parsing and alignment. The motivation for this is that grammar-based approaches such as SCFG (Shieber and Schabes, 1990) does not allow divergence between the analyses. For this reason they suggest using *weak synchronization* instead. The idea behind this is to use synchronization when possible and leave the pieces of the sentences that cannot be synchronized to the monolingual parsers. The monolingual analyses are still required to be well-formed under monolingual CFGs. The space of alignments is restricted to those that can be generated by ITGs. This excludes some of the alignments in the gold-standard but very few (Haghighi et al., 2009).

The two CFG models and the ITG models are independent of each other and therefore synchronization features are added to impose the weak synchronization. The synchronization features are indicator functions relating to the labels of the nodes that can be synchronized. The introduction of these features creates an inference problem that cannot be solved exactly using known algorithms. Instead mean field inference is used.

Burkett, Blitzer, and Klein (2010) uses the English-Chinese translation treebank (Bies et al., 2007) and reports F-scores even better than what was reported by Burkett and Klein (2008), and also get better alignments comparable to a state-of-the-art supervised word aligner. Again they also show improvement in a downstream syntactic MT evaluation.

2.4.7 Grammar-Based Approaches

In parsing and alignment of bitexts a lot of work has been done within what we here call *grammar-based* approaches. We call them this, because

they have in common some formal notion of grammar that models the syntax of the languages in question. We will not try to give a comprehensive overview of these approaches but instead focus on why we do not think that these approaches are appropriate for solving the task we address.

One of the most influential grammar-based approaches is Inversion Transduction Grammars (ITG) (Wu, 1997). The analyses that the grammar allows are typically not like the ones found in treebanks annotated by humans. Another problem in relation to the task we are addressing is the run-time of grammars like these. Melamed (2003) reports the run-time of ITGs to be $O(n^6)$. Within the literature of grammar-based approaches the question addressed is often the expressivity of different formalisms, i.e. the range of structures the grammars allow. Melamed (2003) introduces Multitext-Grammars (MTGs) which apart from allowing more than two languages are also more expressive than many other synchronous grammar formalisms, for instance ITGs. But when the expressivity increases the run-time often follows and some of the parsers introduced by Melamed (2003) have run-times of $O(n^{10})$.

The run-time imposes strong restrictions on what can actually be parsed with these kinds of grammars. Smith and Smith (2004) use a version of MTG to parse a Korean-English corpus that is lexicalized only on one side, which leads to a $O(n^7)$ run-time. This leads them to use sentences of 15 words and shorter.

We are interested in creating treebanks that are based on the annotation of humans and also we are interested in creating them for sentences of any length. Grammar-based approaches are not suitable for solving this problem because of the restrictions described above.

2.4.8 Bitext Parsing Terminology

It is not entirely clear what is meant by the term 'bitext parsing' in the litterature. We distinguish between two different kinds of bitext parsing. The first we called *bilingually informed monolingual parsing* in the sections above and the other *joint parsing*. We will try to describe exactly why we make this distinction, as we will return to it later.

Bilingually informed monolingual parsing is monolingual parsing where the parser is informed by features that depend on another language - the

target language. These features will often depend on syntactic information from the target language, but not necessarily as in the work by Huang, Jiang, and Liu (2009). This kind of parsing is essentially monolingual parsing as only the structure of one language is created. Of course this can be done on both languages in a parallel corpus, but the two parsing models are independent of each other.

Joint parsing is parsing of bitext where the parsing of one side is *not* independent of the parsing of the other language. The reranking approach used by Burkett and Klein (2008) *jointly* models the parsing on both sides. The important distinction is not of whether the parsing is actually done simultaneously but whether the model used for parsing on one side is independent of the model parsing the other.

Joint parsing and alignment is similar to joint parsing, only the alignment will also be modeled. This is the case in the work of Burkett and Klein (2008), but only as a bi product. In the work of Burkett, Blitzer, and Klein (2010) the alignments are explicitly modeled.

Grammar-based approaches will typically be instances of joint parsing (and alignment) as the grammar will model the two (or more) languages simultaneously.

Chapter 3

Data, Evaluation and Tools

3.1 Data

Throughout this thesis we use data from the Copenhagen Dependency Treebank (formerly Danish Dependency Treebank) (Kromann and Lynge, 2004; Buch-Kromann, Wedekind, and Elming, 2007; Buch-Kromann and Korzen, 2010). The latest version contains text in five languages, Danish, English, Spanish, Italian and German although there is a big difference in how much text has been annotated in the five languages.

All of the text has been annotated based on the theory presented by Buch-Kromann (2006). There exists some hand-annotated word alignment between Danish and the other languages, most between Danish and English (Buch-Kromann, Wedekind, and Elming, 2007). In later versions of the treebank some of the texts have been annotated with discourse structures (Buch-Kromann and Korzen, 2010). We will not try to parse these so we will simply ignore them.

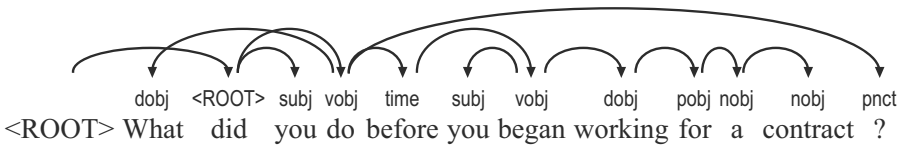

Figure 3.1: Non-projective structure from CDT.

Central to the syntactic theory behind the annotation style is the notion of discontinuous or non-projective structures (Buch-Kromann, 2006). Fig-

	Date	Texts	Sentences	Tokens	Dep. relations
Danish	Jan 22, 2011	457	4,670	85,204	106
English	Jan 22, 2011	457	4,670	94,916	110
Danish	Mar 11, 2011	55	519	10,222	88
Spanish	Mar 11, 2011	55	519	11,172	94

Table 3.1: Statistics for data used in experiments.

ure 3.1 shows such a structure. The structure is discontinuous because the phrase dominated by ”do” spans the words from position 1 ”What” to 12 ”?” but the words ”did” and ”you” are not dominated by ”do”. This is also called non-projective structures and a formal definition is given in section 2.2.1.

The theory also allows for so-called secondary dependencies. An example of this is shown in figure 2.3. We will ignore these secondary dependencies.

The treebank also contains PoS-tags. The Danish part of the treebank contains gold-standard tags but the other languages are automatically tagged.

The treebank is work-in-progress and is continuously being modified. Both because new annotation is being added, but also because the syntactic theory has been revised during the creation of the treebank. For this reason the data used in all experiments is a snapshot of how the treebank looked at a certain point in time. If the data was downloaded today, it would be different. For that reason table 3.1 also contains dates of when the snapshots used were created.

For Danish and English there are approximately 100.000 words annotated, also with alignments, for the other languages somewhat less. We only look at Danish-English and Danish-Spanish so the numbers for the rest of the languages are omitted from table 3.1.

We split each of the two data sets sequentially into four sets, *training - 70%*, *development - 10%*, *validation - 10%* and *evaluation - 10%*.

The term *test* data will be used to describe *unseen* data. In all but the final experiments this will be the development set. In the final experiments it is the evaluation set. The treebank is available from `http://code.`

google.com/p/copenhagen-dependency-treebank[1].

3.2 Evaluation

To evaluate the results of the different approaches we investigate, we need to decide what evaluation metrics to use. In this section we will describe the evaluation metrics used in the following sections.

In the introduction we mentioned that if the purpose of the created treebanks is to help human-annotators in creating a hand-aligned treebank, then the ideal metric will be measuring how much time a human annotator will need to correct the errors made by the automatic method used. This is not a realistic measure because it will require human annotation every time we need to evaluate the output of a system. Instead, some kind of edit-distance can be used under the assumption that this is a reflection of the ideal measure. An even simpler approach is to measure the amount of errors in the output of the system, as these are the ones that the annotators need to address. We choose to use metrics based on the number of errors as this is simple and allows us to use standard metrics from parsing and alignment, as these are based on the number of errors in the output.

3.2.1 Parsing

In dependency parsing the standard metrics are the following:

Labeled Attachment Score (LAS) The percentage of tokens that have the correct head and the correct label.

Unlabeled Attachment Score (UAS) The percentage of tokens that have the correct head.

Labeled Accuracy score (LA) The number of tokens with the correct label.

Often only non-punctuation tokens are included in the evaluation. This is the case in CoNLL shared tasks on dependency parsing (Buchholz and Marsi, 2006; Nivre et al., 2007), and as we use the evaluation script[2] from CoNLL-07 we also exclude punctuation in the evaluation.

[1] The exact snapshots used for experiments is available by contacting the author.

[2] http://nextens.uvt.nl/depparse-wiki/SoftwarePage#eval07.pl

Another commonly used metric is *Exact Match*. This is the percentage of sentences that are parsed completely correct. We do not use this metric here.

3.2.2 Alignment

In the word alignment literature there is often a distinction between *sure* and *possible* links, where the latter are more questionable links. Some aligners also include this distinction in their output, but not all. The metrics used in alignment are *precision*, *recall* and *AER*. If S are the sure links in the gold-standard, P the possible (and sure) links and A the links in the alignment being evaluated the metrics are defined as follows (Och and Ney, 2003):

$$\text{Precision} = \frac{|P \cap A|}{|A|}$$

$$\text{Recall} = \frac{|S \cap A|}{|S|}$$

$$\text{AER} = \frac{|P \cap A| + |S \cap A|}{|A| + |S|}$$

It is important to note that the P set includes both the possible and the sure links. The idea is that you will get rewarded for having correct possible links but not punished for having missed possible links.

If the distinction between probable and sure alignments is dropped, the metrics will be standard recall and precision, and AER will be equal to $1 - F_1$-score, where the F_1- score is the harmonic mean between precision and recall.

We will use AER to evaluate alignments. We will also report precision and recall on both sure an possible links. I.e. we will report standard precision and recall on both of these, not the combined predicion and recall defined above[3] .

3.2.3 Joint Parsing and Alignment

In most experiments we will simply report both parsing metrics for both languages and AER for the alignment. In some cases, we will also report

[3]This is was is reported by the `wa_eval_align.pl`-script from the shared task in the ACL 2005 Workshop on Building and Using Parallel Texts. The script is available from `http://www.cse.unt.edu/~rada/wpt/code/wa_check_align.pl`

a joint metric for the whole task of joint parsing and alignment. In most other work on creating parallel treebanks, phrases-structure based parsing is used. For this F_1-scores are often used, and so it is straight forward to use F_1-scores for the whole structure - i.e. the two trees and the alignment. F_1-score is not used in dependency parsing, but this is simply because that the single-head requirement implies that recall is equal to precision. The number of edges the parser suggest will always be equal to the number of edges in the gold-standard. Therefore we could use F_1-score over the entire structure as well. We will almost do this. We will use a weighted average of UAS for the two sentences and $1 - AER$ for the alignment, i.e. the parallel treebank score (PTS) will be:

$$PTS_{\alpha_a,\alpha_b,\alpha_{ab}} \frac{\alpha_a \cdot UAS_a + \alpha_b \cdot UAS_b + \alpha_{ab} \cdot (1 - AER)}{\alpha_a + \alpha_b + \alpha_{ab}}$$

We do this to retain the sure/possible distinction and the exclusion of punctuation tokens in the parsing evaluation. We use UAS instead of LAS because we are generally more interested in the structure of the parsers than the labels. We use $\alpha_a = \alpha_b = \alpha_{ab} = 1/3$, but this can be changed if one the parts is to be weighed higher than the others.

3.2.4 Significance Tests

We test statistical significance of the results from different approaches in all experiments. For parsing, we test using McNemar's test - we do this with MaltEval (Nilsson and Nivre, 2008). For word alignments we use Dan Bikel's `compare.pl` script[4]. The test uses a type of stratified shuffling. We adapt the script to word alignments, and test only on *sure* links.

Unless otherwise stated we assume that results are significant if $p < 0.05$.

If we compare more than two systems, we use cross-tables to report results from significance tests. If we compare only two systems, we use † to mark significance.

[4] `http://www.cis.upenn.edu/~dbikel/software.html#comparator`

3.3 Tools

In this section we will describe the parsing and alignment tools used in all experiments.

3.3.1 Parsers

MSTParser

In most of our experiments we use a modified version of the MSTParser[5].

Two-stage Parsing

We have modified the MSTParser so it does only unlabeled parsing. The labeling is done in a separate step, as described by McDonald, Lerman, and Pereira (2006). We do this because we are mainly interested in improving the syntactic structure and not so much the labeling, and doing the two stages separately makes it easier to analyze the effects on the unlabeled results of different approaches. The labeler used is from the MSTStacked-parser (Martins et al., 2008). We use only the labeling part of this parser, i.e. we do not use stacked parsing.

We have changed the learning algorithm. We have done this primarily in order to use online learning as we do in the other tools, but also in order to achieve better results. The labeler originally uses logistic regression, and we have replaced this with a number of online learning algorithms. Table 3.2 shows results with different learning algorithms. Table 3.3 shows results

	Danish		English	
	LAS	LA	LAS	LA
Logistic regression	75.41	78.29	78.85	84.67
Perceptron	75.00	77.92	78.72	84.55
MIRA (PA)	75.33	78.27	78.78	84.59
CW	**75.57**	**78.58**	**78.86**	**84.74**

Table 3.2: Accuracy with different learning algorithms for labeling in two-stage parser.

[5] `http://sourceforge.net/projects/mstparser/`

of statistical significance tests for these results.

A	B	C	D	
				A, Logistic Regression
LAS,LA				B, Perceptron
	LAS,LA			C, MIRA
LA	LAS,LA	LAS,LA		D, CW

Table 3.3: Tests for statistical significance for results with different learning algorithms. The lower left triangle is for Danish and the upper right for English.

.

kMST-parser

For the reranking experiments we need a parser that provides k-best lists of parses. We use the kMST-parser described and implemented by Hall (2007)[6].

We use only the k-best part of the parser - i.e. we do not use the reranking.

The kMST-parser uses MaxEnt-learning instead of the MIRA-learning used in the MSTParser. It also uses a different set of features. These two things combined makes it perform sligthy worse in the 1-best case than the MSTParser (Hall, 2007).

In our experiments, however, it performs much worse than the MSTParser. The reason for this, we presume, is that we have used it completely out-of-the-box. We have made no optimizations on neither the features nor the learning.

Baseline Results

Here we will report baseline results on the development data for the parsers described above. This will serve as a reference for future chapters, where we will try to improve these results.

[6]Available from `http://homepage.mac.com/khallbobo/KeithHall/software/depParser0_51.tar.gz`.

Table 3.4 shows results with the MSTParser, and the kMST-parser. In all of these two-stage parsing have been used, and the confidence-weighted classification has been used for labeling.

	Danish			English		
	LAS	UAS	LA	LAS	UAS	LA
MST, 1. order, proj	73.92	86.83	77.10	77.97	83.43	83.74
MST, 1. order, non-proj	74.94	87.99	78.12	78.25	83.59	84.18
MST, 2. order, proj	74.51	87.64	77.92	78.55	84.04	84.27
MST, 2. order, non-proj	**75.57**	**88.79**	**78.58**	**78.86**	**84.21**	**84.74**
kMST (no reranking)	68.60	80.14	74.11	66.90	71.08	76.36

Table 3.4: Baseline results for parsing.

For standard monolingual parsing the kMST-parser does much worse than the MSTParser, and second-order non-projective parsing gives the highest accuracy. In all subsequent experiments, second-order non-projective parsing will be used.

In all experiments with the MSTParser, 10 iterations of training were used. Table 3.5 shows results from tests for statistical significance for the results.

A	B	C	D	E	
	LA	LAS,UAS,LA	LAS,UAS,LA	LAS,UAS,LA	A, 1. order, proj
LA		LAS,UAS	LAS,UAS,LA	LAS,UAS,LA	B, 1. order, non-proj
LAS,UAS,LA			LA	LAS,UAS,LA	C, 2. order, proj
LAS,UAS,LA	LAS,UAS,LA	LAS,UAS,LA		LAS,UAS,LA	D, 2. order, non-proj
LAS,UAS,LA	LAS,UAS,LA	LAS,UAS,LA	LAS,UAS,LA		E, kMST (no reranking)

Table 3.5: Statistical significance tests for baseline parsers. The lower left triangle is for Danish and the upper right for English.

MaltParser

In a few experiments we use the MaltParser (Nivre, Hall, and Nilsson, 2006) which is an open-source[7] transition-based parser as described in section 2.2.3. The default learning algorithm used is support vector machines, more specifically the LIBSVM learner (Chang and Lin, 2001). The default is to use a second-degree kernel, and we do not change this in our experiments.

We use the *NivreEager* algorithm and use the baseline pseudo-projective approach for handling non-projective trees (Nivre et al., 2006). The performance of the parser is very dependent on the features used and of optimization of the hyper-parameters for the SVM learning. Instead of using the parser out-of-the box we use the features and parameters[8] that were used by the team behind the parsers in the CoNLL shared tasks in 2006 and 2007 (Buchholz and Marsi, 2006; Nivre et al., 2007). The Danish data from CoNLL-X (2006) are from the Copenhagen Dependency Treebank, so for Danish, the features and parameters should be quite good. The English parameters and features are optimized for the data from the shared task and not for the data we use, which means that these are probably sub-optimal.

3.3.2 Minimum Cost Flow Aligner

All experiments have been done with an aligner we have implemented. We also tested LinguaAlign, which is described briefly in section 2.4.3, but we found that the results from this aligner are not competitive[9] so we do not use it in any of the following experiments. The results with LinguaAlign are reported below.

The aligner we have implemented is based on the work by Taskar, Lacoste-Julien, and Klein (2005) and Lacoste-Julien et al. (2006) which we have described in section 2.3.2. The implementation though, is a complete reimplementation.

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

[8] Available from `http://maltparser.org/conll/conllx/` and `http://maltparser.org/conll/conll07/`

[9] We did try several different algorithms and combinations of features, but it is still possible that better results could have been obtained by using different features, algorithms and hyper-parameters.

We allow fertility of more than 1 and therefore we cast the inference problem as a minimum cost flow problem.

We do not use higher-order features as described by Lacoste-Julien et al. (2006). Instead of Max-Margin Markov Networks (M^3N) we use MIRA for learning. Taskar, Lacoste-Julien, and Klein (2005) report significantly worse results with averaged perceptron than with M^3N, but M^3N are reported to be slow (Daume, 2006) and we want to use much larger training data sets than the 100 or 200 sentences used in the experiments with M^3N.

Minimum Cost Flow Algorithm

The inference problem we must solve is finding the minimum cost flow in a weighted directed graph. More specifically, we have graphs that look like the one shown in figure 3.2. The task is to find the maximum flow from the source (the leftmost vertex) to the sink (the rightmost vertex), that has the minimum cost.

Several algorithms exists that can solve this problem efficiently (Ahuja, Magnanti, and Orlin, 1993). Here we use the *successive shortest path* algorithm. For general graphs, the algorithm has pseudopolynomial running time $O(nU \cdot SP)$, where U is the upper bound on the supply of any node in the network and where SP is the time it takes to solve a non-negative single source shortest path problem. For solving the shortest path problem we use Dijkstras algorithm which has run-time $O(n^2)$. In the specific kind of network used in this word alignment task, we know what the upper bound for U is, and this makes the algorithm polynomial instead of pseudopolynomial. The only node having supply is the source node, and the supply of this node is equal to the maximum amount of flow it is possible to push through the network. Let l be the number of words in the source sentence, m the number of words in the target sentence and f the maximum fertility allowed in the alignment. The number of nodes in the graph will then be $n = fl + fm + l + m + 2$. The maximum flow possible in the network is $min(l, m) \cdot f$. As the algorithm in the general case terminates in maximum nU iterations (Ahuja, Magnanti, and Orlin, 1993), it will terminate in maximum $n \cdot min(l, m) \cdot f$ iterations in the word alignment case. This makes the run-time of the algorithm $O(n \cdot min(l, m) \cdot f \cdot n^2) \approx O(n^4)$. However, this is artificially high, as we can easily design the graph to make the run-

time $O(n^3)$. If we delete the sink node and source node and instead give all the left-most vertices (the non-word vertices) a suppy of 1 and all the right-most vertices with a demand of -1, then $U = 1$, which makes the run-time $O(n^3)$ instead (and with two fewer vertices also). We suspect that the special topography of the graph will actually lead to even lower run-time, but we have no proof of this.

In practice, edges with score below zero will be left out of the graph reducing the run-time, and as the timings in figure 3.4 show, the aligner is fast in actual use. The aligner uses considerably more time on feature extraction than on actual inference.

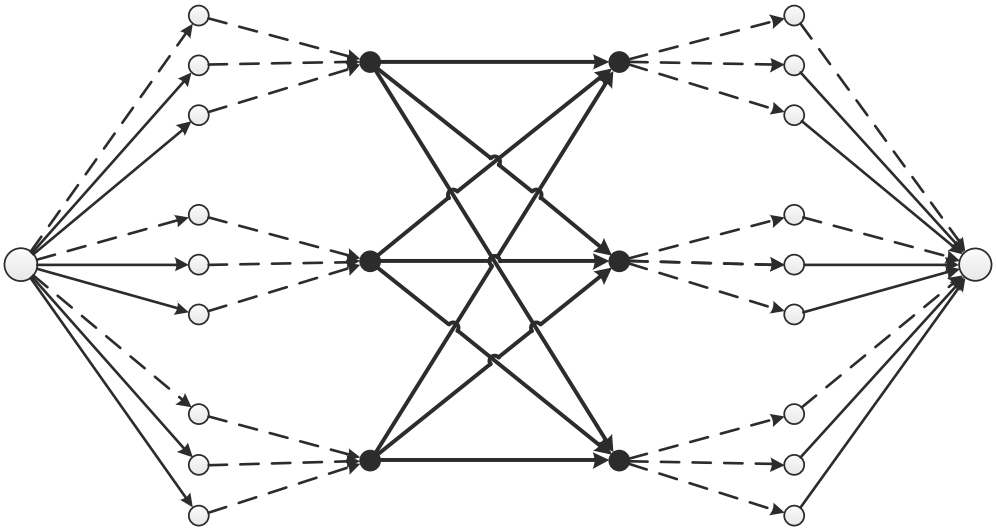

Figure 3.2: Directed graph representing a word alignment task with fertility higher than 1 (3 in this example). Solid dots represent the words. Dashed lines have no cost. The leftmost vertex is called the sink and the rightmost the source. Finding a maximum scoring alignment can be solved using a minimum cost maximum flow cost algorithm that.

Features

Here we will list the features used in the aligner. We will split them into three groups. *Internal features* which are features relating only to the input sentences themselves. *External features*, which are features that are based on

the output of some other tool - but not from the parsers. *Syntactic features* which are features based on the dependency analysis of each of the input sentences.

All features are edge-factored, i.e. they pertain to one possible link in the structure.

Internal

word-pair The form of the pair of words in the link. Also for two previous and two following words.

match Do the two words have the same form? Also when the words are lower cased.

abs-diff-rel-pos The absolute difference between the relative position of the words.

LCS Length of longest common substring.

case Source word, target word or both starts with uppercase letter.

punctuation Features describing if the words begin or end with punctuation or are entirely punctuation.

External

POS PoS-tags for the two words. Also for two previous and two following words.

dice Dice-coefficient for pair of words. Also for two previous and two following words.

GIZA Translation probabilities from GIZA++ in both directions for word pair. Also for two previous and two following words.

Moses Indicate if links exists in output from symmetrized GIZA++ alignments. Also for two previous and two following words.

diff Log Rank Normalized difference in log rank between two words.

Syntactic

Label pair Pair of the label of the incoming dependency relation of each token.

span diff Difference in length of spans dominated by the two tokens.

In sections 2.4.2, 2.4.3 and 2.4.5 we saw that a feature that is often used in sub-tree alignment is the inside-outside feature. As described, there are different variants of this feature, but they all try to capture the same thing, which is consistency in the aligned trees. If node a is aligned to node b and b dominates c it is undesirable that c is aligned to something that is not dominated by a. We have tested using different variants of the inside-outside feature but consistently found that it increased AER.

Training

As mentioned above we use MIRA for training. We use k-best MIRA as this is reported to be much faster than factored MIRA without leading to a large decrease in accuracy. (McDonald, Crammer, and Pereira, 2005; McDonald et al., 2005). For all experiments $k = 1$ as the inference algorithm described above only provides the optimal solution. We are not aware of any exact k-best minimum cost flow algorithms. For first-order non-projective parsing, the MSTParser actually uses a heuristic k-best list[10] with good results. We have not tried using heuristic k-best lists.

We use a fixed set of iterations and average the weights only after the last iteration. The aligner also allows the use of averaged perceptron (Collins, 2002) but as figure 3.3 shows MIRA consistently yields better results.

The loss-function used is the sum of incorrectly predicted links and missing sure links. This means that we optimize towards F_1-score for sure links.

Fertility

In theory, the maximum fertility of the aligner should be set to be the highest fertility we could imagine a word having. However, this could have a

[10] Unlike Hall (2007) who uses exact k-best lists.

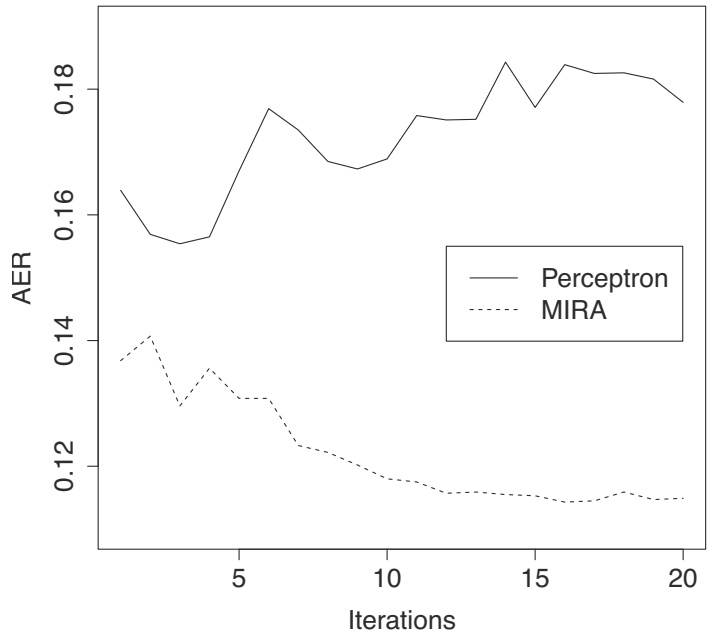

Figure 3.3: Results on development data when training with different learners and different number of iterations.

detrimental effect on the accuracy as the learning problem becomes more difficult and will definitely increase inference time because the graph will contain many more vertices. In the data used, we observe a maximum fertility of 12, but as table 3.6 shows lower fertilities are much more common. Figure 3.4 shows AER on development data and speed of aligner with different settings for maximum fertility. We see that the aligner is very robust with respect to handling large fertility - the best alignment is actually obtained when maximum fertility is set to 8 even though we have seen that only around 0.01% of words has fertility this high. The speed of the aligner of course decreases with higher fertility, but not much. This is due to the fact that the inference in practice is very fast and most of the computation time is used on feature extraction.

We use a maximum fertility of 5 in the following experiments as AER is low with that settings and the aligner is only a little slower than with 3 or 4.

	Danish		English	
Fertility	%	Cumulative	%	Cumulative
0	6.98	6.98	6.42	6.42
1	77.92	84.90	86.69	93.11
2	10.43	95.34	4.95	98.05
3	3.14	98.47	1.48	99.53
4	1.13	99.60	0.32	99.85
5	0.26	99.86	0.09	99.94
6	0.07	99.93	0.02	99.96
7	0.04	99.97	0.00	99.97
8	0.01	99.98	0.02	99.99
9	0.00	99.98	0.00	100.00
10	0.00	99.98	0.00	100.00
11	0.00	99.98	0.00	100.00
12	0.02	100.00	0.01	100.00

Table 3.6: Fertilites of words in corpus.

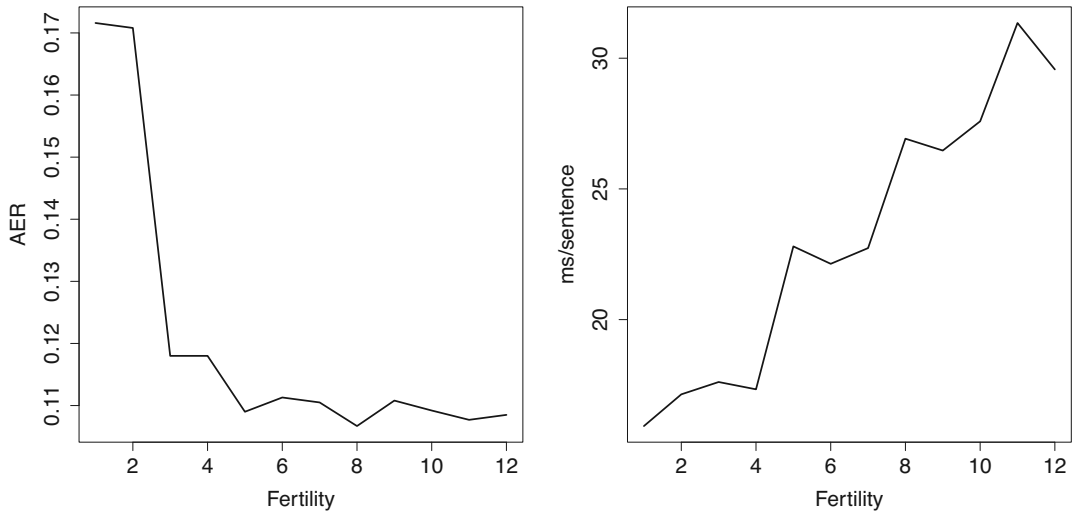

Figure 3.4: AER on development data and speed of aligner with different settings for maximum fertility.

Training Data

The aligner is trained on gold-standard alignments, but the question is what to do with the trees. In some cases there will be gold-standard trees available, but the problem with training on these is that the information from these will be too reliable compared to when the aligner is used on new data (assuming that this does not have gold-standard alignments).

Table 3.7 shows results when training on gold-standard trees and when using 10-fold jack-knifing to create trees for the training data. The difference is not huge, but we still see that using gold-standard trees is not a good idea. Table 3.8 shows statistical significance for these results.

	prec (S)	rec (S)	prec (P)	rec (P)	AER
LinguaAlign (gold)	86.36	79.61	89.71	77.62	15.54
LinguaAlign (10-fold)	86.42	79.67	89.77	77. 68	15.49
MCFAligner (no trees)	88.81	83.18	91.86	80.74	12.63
MCFAligner (gold)	89.52	83.52	92.60	81.08	12.10
MCFAligner (10-fold)	**89.73**	**83.86**	**92.84**	**81.45**	**11.80**

Table 3.7: Scores when using either gold trees or jack-knifed trees as input when training aligners.

A	B	C	D	E	
	P R	P R	P R		A, LinguaAlign (gold)
		P R	P R	P R	B, LinguaAlign (10-fold)
			P	P R	C, MCFAligner (no trees)
					D, MCFAligner (gold)
					E, MCFAligner (10-fold)

Table 3.8: Statistical significance tests for alignment results. P means that *precision* of sure alignments is significantly better, R means that *recall* of sure alignments is significantly better.

3.3.3 Reranker

For reranking we use SVM^{rank} [11] (Joachims, 2002). We have described the theory behind this in section 2.1.4.

3.3.4 Sizes

We are interested in developing a method that can be used both for creating resources for machine translation but also can make the creation of hand-aligned parallel treebanks less time consuming. For the latter task it will not often be the case that a large amount of annotated data already exists. Instead the process could start with hand-annotating a small number of sentences, and then use these to train a system to automatically create annotation, which then could be corrected by hand. Therefore, we are interested in how our different approaches perform when different amounts of training data are available.

Figure 3.5 shows the performance of the baseline parsers depending on the size of the training data[12]. As expected the performance increases with the number of training sentences.

[11] Available from `http://www.cs.cornell.edu/People/tj/svm_light/svm_rank.html`

[12] We have used the same parameters for all parsers. This probably means that for the smaller training sets there is a huge risk of overfitting as the parsers have been trained for 10 iterations.

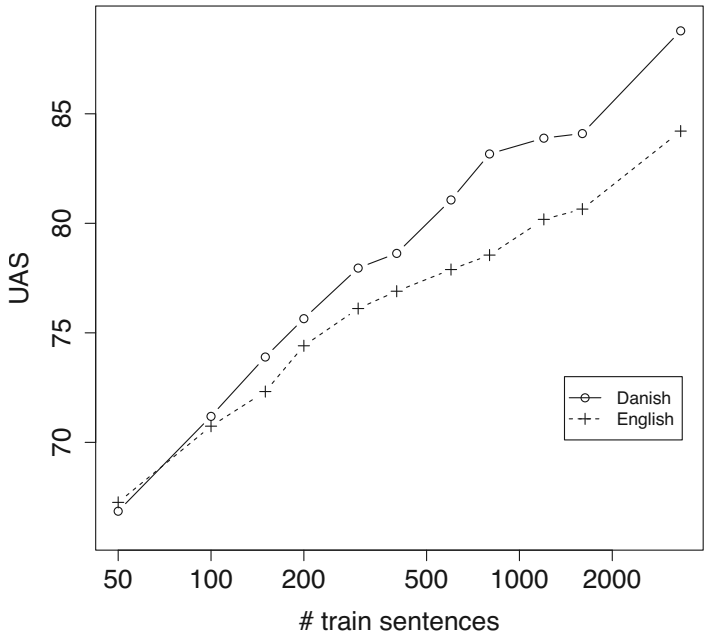

Figure 3.5: UAS of baseline parsers with different amounts of training data.

Chapter 4

Bilingually Informed Parsing

In this chapter and the next we will present different approaches to creating parallel treebanks. First we will look at how *bilingually informed* parsing, can improve the parses on each side in a parallel treebank. Bilingually parsing in itself does not create a parallel treebank, but when we want to create a parallel treebank a parallel corpus is available. This makes bilingually informed parsing possible. And the assumption is that bilingually informed parsing leads to better parses than standard monolingual parsing.

After a formal definition of the task of creating a parallel treebank we will turn to a more general discussion of approaches to creating these. Especially why we believe bilingual parsing will work.

This will be followed by a more in-depth analysis of bilingual parsing for closely related languages.

4.1 Formal Definition

Defining the task of creating the structures in a parallel treebank formally consists in combining the definition of the parsing task and the alignment task.

Let S and T be two sentences as defined in definition 1, i.e. $S = w_0^s w_1^s \dots w_n^s$ and $T = w_0^t w_1^t \dots w_n^t$. To create a parallel structure we need to find a well-formed dependency graph $G^s = (V^s, A^s)$ for S and $G^t = (V^t, A^t)$ for T, as defined in definition 4. In this work we use non-projective trees as defined in definition 7.

The two sentences also need to be aligned as defined in definition 10. The crucial point is that the two node sets used for the alignment, and the two node set used for the two trees are the same - this is what combine the structures. There is one exception though, and that is the artificial root-tokens used in parsing. We do not align these. This means that $V^a = V^s \setminus \{w_0^s\}$ and $V^b = V^t \setminus \{w_0^t\}$ where V^a and V^b are the two disjoint sets in the bipartite graph that is the alignment (definition 10).

4.2 Baseline Approach

The most straight forward approach to doing bitext dependency parsing is to create the three structures separately. This means using a monolingual parser on one language, using one on the other language and then using a word aligner for aligning the words in the two sentences. This will create the desired structure.

The work presented here, including bilingually informed parsing, focuses on how to combine alignment and parsing. Presumably, the result will be better if we combine the prediction in a way that will make the three structures depend on each other.

4.2.1 Why Can We Improve the Baseline?

The baseline approach will create the desired structure, however we expect that we can do better than creating the tree structures separately. The three structures should be able to affect each other in a beneficial way. First let us consider how the two syntactic structures might help the word alignment process. Consider the bitext in figure 4.1. This example is easy to

<ROOT> Han er forsvundet
He is disappeared

<ROOT> He has disappeared

Figure 4.1: Parallel sentences.

align but in some cases it is not obvious if the words ”has” and ”er” should be aligned. If the input to the word aligner is instead as in figure 4.2 The aligner has a lot more useful information. Both of the words are the

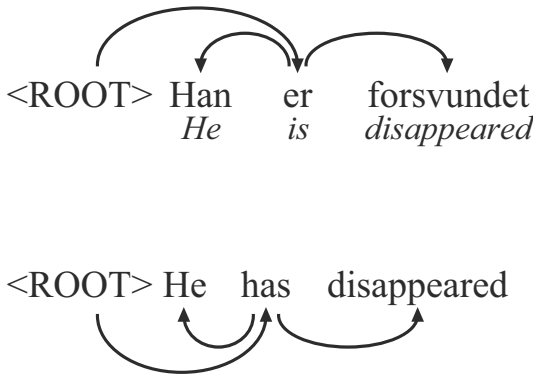

Figure 4.2: Parallel trees with dependency analyses.

root in the sentence and their dependents are probable translations of each other. With this extra information it is considerably more likely that the two words should be aligned. Now let us turn to the main focus of this chapter, how parsing of one language can benefit from an existing word alignment and a parse for the other language as illustrated in figure 4.3. If the parser

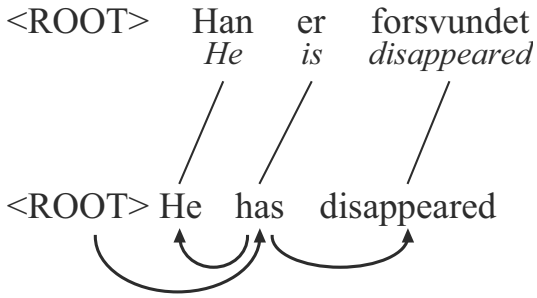

Figure 4.3: Example of how alignment and target side tree can help source side parsing.

has to decide whether or not ”er” should be the head of ”forsvundet”, it can now look at the alignments and check if the word aligned to ”er” is the head of the word aligned to ”forsvundet”. In this case it is, and that makes it more likely that ”er” should be the head of ”forvundet”.

This example is good to illustrate the basic idea behind bilingually in-

formed parsing, but the example is not very realistic because the Danish sentence will probably not be a problem for the Danish parser. And if the Danish parser has problems with this construction there is a good chance that the English will as well. If the English parser also has problems there is a large risk that the output from this will be erroneous, which will make the input to the bilingually informed parser incorrect. The reason for this is that the two sentences are highly parallel. Most work on bilingually informed parsing actually focuses on languages that are not highly parallel. For instance Chen, Kazama, and Torisawa (2010) use the example in figure 4.4 to motivate why bilingually informed parsing is useful. In English pp-attachement is a problem, but apparently not in Chinese. For this reason the Chinese sentence, where there is no ambiguity, can help disambiguate the English sentence where there is.

We focus on languages that are closely related, primarily Danish-English. This leaves the question whether bilingually informed parsing will work for closely related languages. We go into this question in detail in section 4.4.

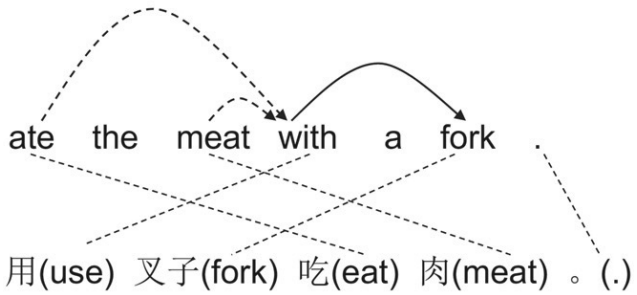

Figure 4.4: Example of how Chinese parse tree can disambiguate English parsing.

We discussed the problem with erroneous input from a parser, but still the example shown here oversimplifies the issue. There are several potential problems to consider. The most obvious is that we will not have gold-standard trees and alignments to use in practice. For the word alignment case, this means that the trees available on the two languages may contain errors. For the parsing case in means that not only can the word alignment be wrong, but the tree on the other language can also be wrong. In the ex-

ample above, the fact that there is a relation between the two words that the considered words are aligned with is a very strong indication that there should be a relation between the two words. In real life, this might not be the case because of errors in the word alignment and in the tree on the other side.

4.2.2 Graph-Based Approach

We focus mainly on graph-based approaches, and therefore it is natural to consider whether we can formulate the problem in a way where we can apply some graph-algorithm to solve the problem. For the parsing problem we saw that MST-algorithms can be used, and for alignment we can use assignment or minimum cost flow algorithms. These cannot simply be combined, but it is possible that the problem can somehow be described in a way where one algorithm can create all three structures simultaneously. We have not pursued this direction because of the following problem. If this algorithm requires edge factored features the results will be the same as creating the structures independently of each other. The factorization will make all scores of the edges of the parses independent of the rest of the structure, also the other parse and the alignment, and the alignment scores will be independent of the parses. Therefore there will be no interaction between the structures.

With edge-factorization it will make no difference to treat the three structures simultaneously. The question is then whether we can use a richer features-structure. As discussed earlier second-order non-projective parsing is NP-hard. This implies that an algorithm for solving everything at once will also be NP-hard if we want non-projective parsing. Of course a hill-climbing approach can be used to change the projective parse trees into non-projective trees as described earlier, but then the edges that the alignment and the other tree are based on will be changed.

The problems described above do not imply that a graph-based approach will not work. It only implies that it will be an approximate approach. We can only find the optimal solution with edge-factorization and this is equivalent to creating the three structures independently of each other.

4.3 Extended Parser

Before turning to the analysis of bilingually informed parsing we will describe the parser we have used for doing experiments with bilingually informed parsing.

4.3.1 Modified MSTParser for Extended Parsing

We have modified the MSTParser (see section 3.3.1) to also do bilingually informed (extended) parsing. Extended parsing allows the parser to use information given by a word alignment and a syntactic (dependency) structure of the sentence aligned to.

This modification consists in reading input containing this information, and extending the feature-extraction part of the parser to use this information.

The extra features will have weights attached to them which are learned in the same way as the weights for the standard monolingual features. This way of treating the bilingual information corresponds to weak synchronization (Burkett and Klein, 2008; Burkett, Blitzer, and Klein, 2010) or sloppy transfer (Smith and Eisner, 2006; Smith and Eisner, 2009), i.e. the parser is not forced to generate a structure corresponding to the target side structure - it only has information from this to guide the parsing.

All the standard features from the MSTParser are left unchanged. To these we add a few features that use the information from the alignment and the analysis of the parallel sentence. The features are first-order, i.e. they relate only to the possible relation between two words. All the features indicate whether or not there exists a relation in the parallel analysis between tokens aligned to the head and tokens aligned to the dependent.

head1-dep1 The head and the dependent is each aligned to exactly one token and there is a relation from the token aligned to the head to the token aligned to the dependent.

head1-depm The head is aligned to exactly one token and there is a relation from this to a token aligned to the dependent.

headn-dep1 The dependent is aligned to exactly one token, and there is a relation to this token from a token aligned to the head.

headn-depm There is a relation from a token aligned to the head to a token aligned to the dependent.

In addition to these there are corresponding features that indicate if a relation in the opposite direction exists.

As mentioned extended parsing is an instance of bilingually informed monolingual parsing as described in 2.4.8. The parsing approach is similar to the one used by Chen, Kazama, and Torisawa (2010), although they also use second-order features.

4.3.2 Training Data

To train an extended parser the training data needs to have alignments and trees on the target language. We do have gold-standard treebanks with both alignments and trees available so we could simply train the parsers on this.

This is not a good idea though. We have already seen in section 3.3.2 that the aligner performs better when not trained on gold-standard trees. Instead we can use jack-knifing to create training data where the target side structures and the alignments are output from a parser and an aligner instead of gold-standard. The idea is that by doing this, the extended input at training time will be similar to the extended input at test time.

Trees	Alignments	Danish	English
gold	gold	84.03	84.44
gold	cross	85.99	85.27
cross	gold	**88.89**	**85.93**
cross	cross	88.79	85.72
Baseline		88.79	84.21

Table 4.1: Accuracy (UAS) of extended parsers when trained of different combinations of gold-standard data and parser/aligner output data.

Table 4.1 shows results when using different combinations of gold-standard data for training the extended parsers - 10-fold jack-knifing was used here and is used in all subsequent experiment. We see that training on gold-standard trees on the target side is a bad idea - especially for Danish where there is an increase of around 3 points by using the jack-knifed parses of the target side. On the English side the difference is smaller but still there. We actually see that using gold-standard alignments is slightly better than using the output from the aligner. This is counter intuitive as we expect that the system performs better when the training data resembles the data at test time. Later experiments with other features also showed the opposite effect so jack-knifing will be used to create both the alignments and the trees in the training data for the extended parsers in all subsequent experiments. Table 4.2 also shows that the difference between the two is not statistically significant.

A	B	C	D	
	UAS	UAS	UAS	A, gold-gold
UAS		UAS	UAS	B, gold-cross
UAS	UAS			C, cross-gold
UAS	UAS			D, cross-cross

Table 4.2: Tests for statistical significance of UAS with different training data. The lower left triangle is for Danish and the upper right for English.

.

4.3.3 Sizes

Figure 4.5 shows the difference between the UAS of the baseline parser and the extended parser for different amounts of training data. We see quite clearly that the extended parsing works better with small amounts of training data. The reason may simply be that there is much more room for improvement.

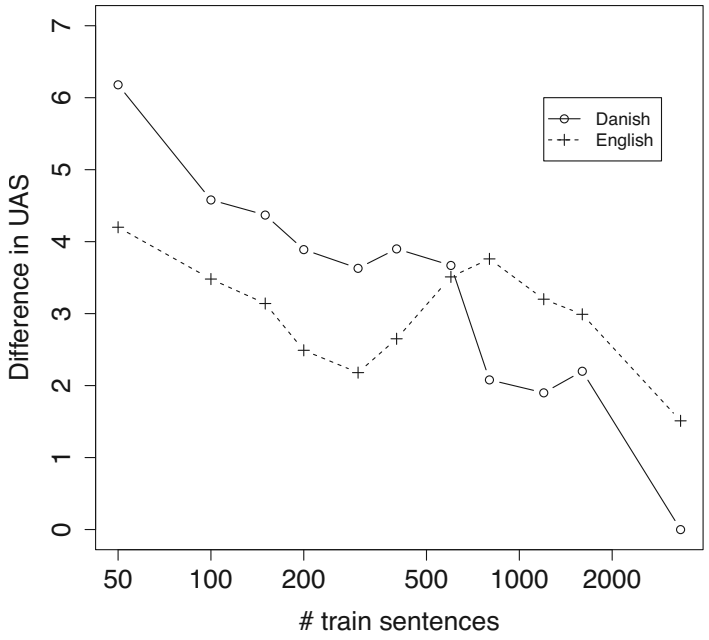

Figure 4.5: Difference in UAS between baseline parsers and extended parsers with different amounts of training data.

4.4 Analysis

In this section we will try to analyze the Danish-English data in order to shed more light on some of the basic assumptions underlying the work in bitext dependency parsing. Furthermore, this analysis will provide some insight into the kind of features that are necessary in order to obtain good results.

In the analysis we will look both at the data that is used in the experiments, and at the output from the extended parsers in order to analyze which errors these make, and if it is possible to reduce the number of errors.

4.4.1 An Example

Before we turn to the more in-depth analysis, let us reconsider why we believe that bilingual information can actually help parsing. We saw a Chinese-English example, but the question remains whether there are good

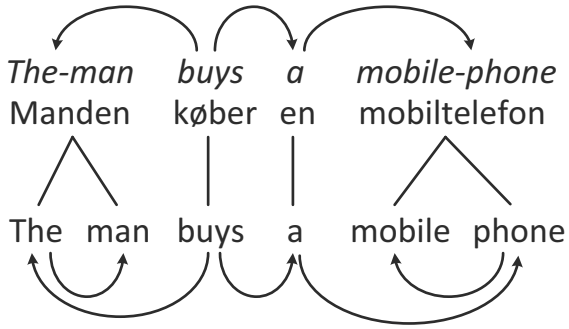

Figure 4.6: Example of systematic difference between Danish and English.

reasons to believe that it will help with Danish-English.

We believe that it will. One reason for this is the use of compound words in Danish. Another concerns the different use of definite determiners in the Danish. Figure 4.6 shows an example that contains both of these phenomena. We see two cases where the Danish word is written as one, and the English equivalent is written with two. In both cases there is a relation between the two English words. The hyphens in the glosses also heavily suggest that these relations exist. The hypothesis is that if two (and only two) English words are aligned to one (and only one) Danish word, there is a relation between the two English words. If this is true, this could be a big help in parsing English. The Danish orthography will always predict a relation correctly (although not in which direction it is).

	2-1	Baseline UAS
Danish	92%	95%
English	96%	93%

Table 4.3: Statistics for 2-1 alignments. The ”2-1” column shows part of 2-1 configurations where there is a relation between the two source side tokens. The ”Baseline UAS” column shows accuracy of baseline parser on these relations.

The example is made up, and we need to test this assumption in real

data. To do this we look at the CDT data for 2-1 alignments and check how often there is a relation (in either direction) between the two words. For completeness, we include the information in the other direction (two words in Danish and one in English) although we do not hypothesize anything about this.

The results can be seen in the 2-1 column in table 4.3. We see that it is very often the case that there is a relation - especially when we look at English. The reason for this is of course the compound words and different use of determiners in Danish, which we discussed earlier and saw in figure 4.6. It is not always the case though. The reason for this is the annotation used in the CDT. Figure 4.7 shows an example where there is a 2-1 relation, but not a relation between the two words.

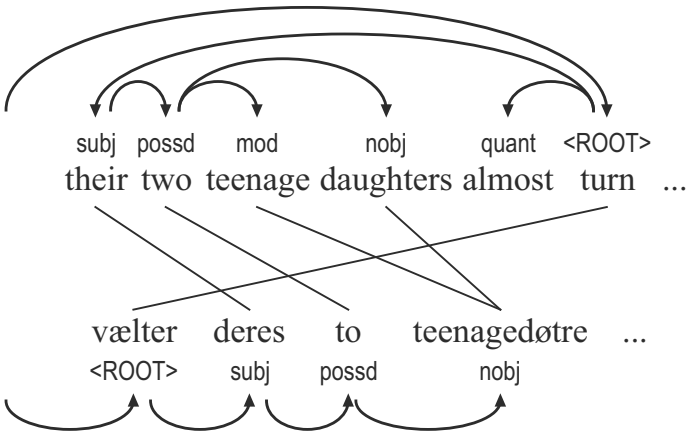

Figure 4.7: Example of 2-1 alignment without a relation between the two tokens.

In order to avoid the 4% where the hypothesis does not hold, and hopefully find information that is even more reliable for extended parsing, we try to look for a more specific pattern. If we look at the example in figure 4.6 we see that in both of the 2-1 cases, the heads of the tokens involved in the alignment are also aligned. Therefore we will look at situations where the head of the Danish token in the 2-1 alignment is aligned to a token that is the head of only one of the two English tokens. Figure 4.8 shows how this configuration can look.

If we do the same statistics for these as we did for the more generic 2-1 configuration, we see a very high correspondence. Table 4.4 shows this.

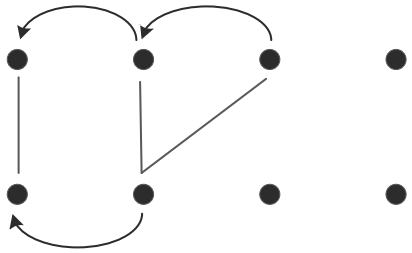

Figure 4.8: More specific 2-1 configuration.

The correspondence is very strong in both cases, but this may not be very

	spec. 2-1	Baseline UAS
Danish	92%	95%
English	99.6%	92%

Table 4.4: Statistics for more specific 2-1 alignments where there is a relation between the two source side tokens, and accuracy of baseline parser on these relations.

helpful because many of these relations may be easy for the baseline parser which means there is very little to gain. This is in fact the case, as table 4.3 and table 4.4 show. The accuracy of the relation between the two words in a 2-1 configuration (where there is supposed to be a relation) is very high. Nonetheless, it is possible that thisaccuracy can become even higher by using the fact that the relation very often exists.

4.4.2 Correspondence

The idea, that joint parsing of bitexts leads to better parser accuracy, rests on the assumption that there is some syntactic correspondence between the two sentences. This assumption has been empirically justified by research that shows that better results can be achieved using joint or bilingually informed parsing. Several examples of this have been discussed in section 2.4.

Furthermore, we saw that Hwa et al. (2002) investigated the assumption more systematically and found that projecting analyses directly from English to Chinese gives a F_1-score of only 38.1, but that by hand-crafting simple rules this number increases to 67.3.

The data we use here, differs from the data used in other experiments, which makes it interesting to retest the assumption of correspondence. First of all, the data used in our experiments has both hand-annotated dependencies and alignments. This is also the case in (Hwa et al., 2002), but otherwise most work relies on automatically created alignments. Furthermore, in our case the languages are closely related, and the annotation on each language is done using the same underlying syntactic theory (Buch-Kromann, Wedekind, and Elming, 2007).

Instead of looking at the parser accuracy when projecting an analysis from one language to another, we test the assumption by looking at configurations where the analysis on one language could possibly be beneficial when parsing the other.

Initially, we look at four types of configurations. Figure 4.9 illustrates these.

TRUE For a given dependency arc, both the dependent and the head is aligned to exactly one token in the other sentence and there exists a dependency-relation between these (in the same direction).

FUZZY For a given dependency arc, the head or the dependent (or both) is aligned to more than one token and there exists a relation (in the right direction) between some of the tokens aligned to the head and some of the tokens aligned to the dependent.

FALSE There is no relation between the token(s) aligned to the head and the token(s) aligned to the dependent.

NEITHER Either the head or the dependent is not aligned.

Table 4.5 shows the distribution of the types of configurations on a development set of 416 sentences in Danish and English. We see that around

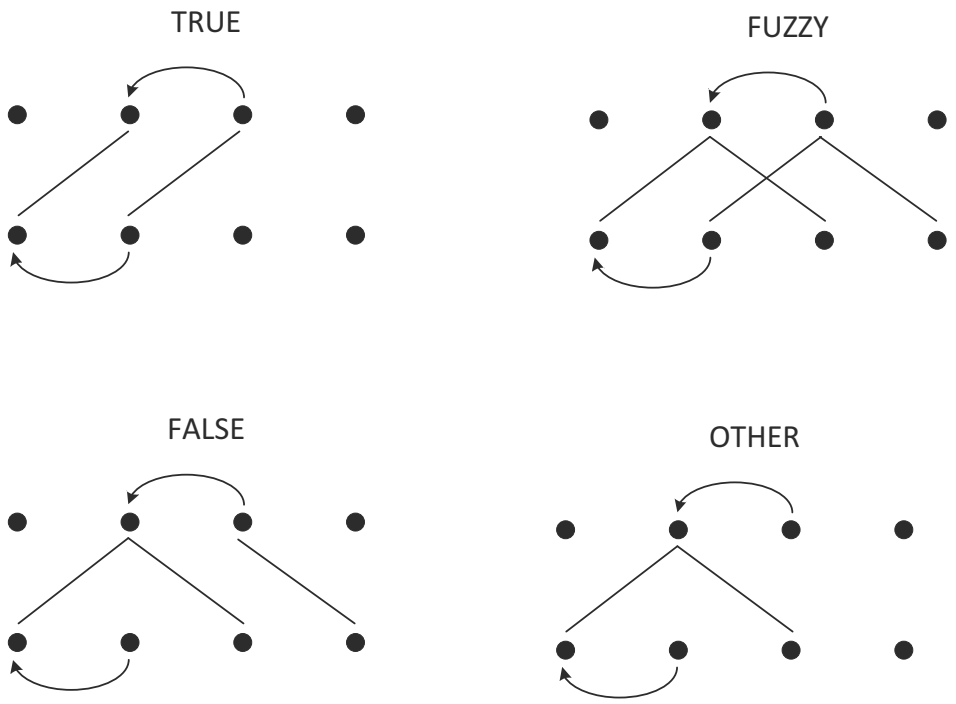

Figure 4.9: Types of configurations.

60% of all dependencies are of the type TRUE. This is high, but not surprising, as Danish and English are closely related languages and the same annotation scheme has been used for both languages.

	Danish	English
TRUE	59.7%	65.1%
FUZZY	28.5%	15.2%
FALSE	7.9%	16.8%
NEITHER	3.9%	2.9%

Table 4.5: Distribution of types on configuration for Danish-English.

More interestingly, there is a big difference between the numbers for FUZZY and FALSE. There are two main reasons for this, rooted in the differences between English and Danish we discussed above. Figure 4.10 shows an example of why this affects the distribution. The two sentences

are highly parallel, but we see that where the Danish side is classified with FUZZY the English side is classified with FALSE.

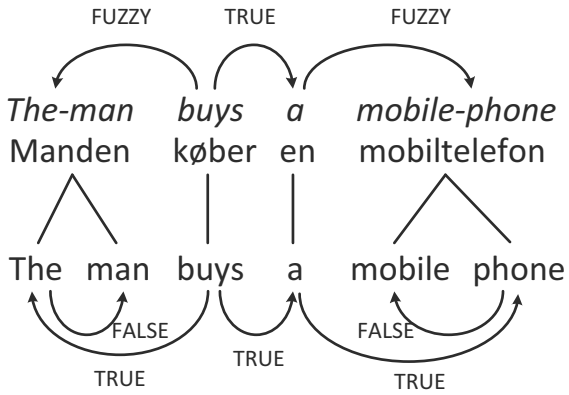

Figure 4.10: Example of configurations leading to more FALSE in English than in Danish.

We conclude that the categories are too coarse for measuring correspondence. To get a better idea of this, we introduce two new configurations, based on the more specific 2-1 configuration we looked at above. These are illustrated in figure 4.11.

P-TRUE For a dependency arc, the dependent is aligned to one token, the head is aligned to (only) the same token and the head of the head is aligned to (only) the head of the token the dependent and head is aligned to.

P-FUZZY For a dependency arc, the head and dependent is aligned to the same token and there exists a relation between at least one of the tokens the dependent is aligned to and at least one of the tokens the head of the head is aligned to.

This leads to a new distribution of configurations, shown in table 4.6. We see that a large part of the FALSE on English were due to this kind of configuration. Figure 4.12 shows the example sentence again, but with the new categories. We see that these categories better captures the fact that the sentences are parallel.

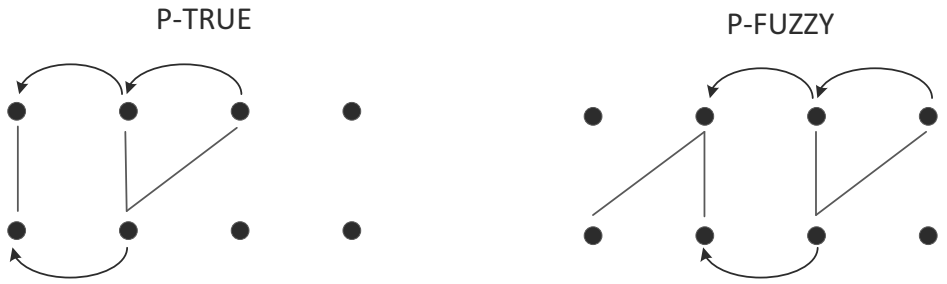

Figure 4.11: New P-Types.

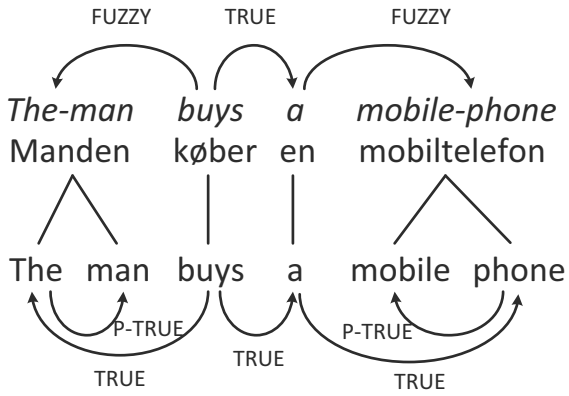

Figure 4.12: Same example as above but with new configurations. Now there are no FALSE arcs in English.

This systematic difference underlines the importance of designing good features for the extended parsing. Even with such closely related languages as English and Danish there are some systematic differences that need to be taken into account.

Correspondence in Parser Errors

Above we saw that there is a high degree of correspondence between the analyses in Danish and English in the treebank. This can be explained by

	Danish	English
TRUE	59.7%	65.1%
P-TRUE	2.0%	7.9%
P-FUZZY	0.5%	0.8 %
FUZZY	28.5%	15.2%
FALSE	5.4%	7.9%
NEITHER	4.0%	3.0%

Table 4.6: Distribution of types on configuration for Danish-English.

two facts, namely that English and Danish are closely related languages[1], and that the annotation guidelines were the same for both languages. This poses the question whether something can actually be learned from the parsed parallel sentences in the other languages. If the analyses are basically the same, then there is a good chance that the parsers will make the same kind of mistakes. If this is the case, the Danish parser cannot learn from the output of an English, as the errors will largely be in the same places.

We now investigate this and determine whether or not it is in fact the case that the parsers make the same errors. Table 4.7 shows the distribution of configurations when we look only at the tokens which a standard parser assigns the wrong head. As the correspondence is based on output from parsers on both sides, a large degree of correspondence will mean that the parsers makes the same errors, i.e. even in the errors in the parser output, the analyses on both languages are the same. As we see, the correspondence is much lower than the general correspondence we saw above. This implies that the analyses from the parsers are not the same.

Unfortunately, this does not mean that the output from e.g. the English parser can help the Danish. We know that in many cases where the Danish parser makes mistakes, the English parser does not create the same analysis. But we do now know whether or not the English parser creates the correct analysis, and even if it does, it might not be an analysis that can

[1]In the Copenhagen Dependency Treebank in particular as the translators were instructed to translate from Danish to English as directly as possible (Buch-Kromann, Wedekind, and Elming, 2007).

	Danish	English
TRUE	21.9%	37.6%
P-TRUE	0.8%	3.0%
P-FUZZY	0.1%	0.2 %
FUZZY	15.6%	11.6%
FALSE	57.5%	45.24%
NEITHER	4.2%	2.9%

Table 4.7: Correspondence in parser errors.

help the Danish parser.

To try and get an idea of whether or not the parser output can help, we will look at the data in another way. For every token in the source language that has the wrong head in the parser output, we look at the token that this is aligned to (gold-standard) and see if this token has the correct head. If it is aligned to more than one token, we see if any of these tokens have the correct head. The results from this analysis are shown in table 4.8.

	Danish-English		English-Danish	
Alignment type	incorrect	correct	incorrect	correct
1-1	47%	35%	35%	54%
1-M	5%	13%	3%	8%

Table 4.8: How often there is help available in the parse of the parallel sentence.

We see that there is help available from the parses on the other language. If we look at the Danish-English case where the output from the English parser should help the Danish parser, we see that in 35% of the cases the token is aligned to only one token and that token has the correct head. In these situations it is definitely possible that an extended parser can learn something from the analysis on the target language. In 13% of the cases one of the aligned tokens has a correct head. This is very vague and is not necessarily situations where the extended parser can benefit from the target side analysis. For the remaining 52% there is no help to get, as

the analysis on the target side is wrong. In the other direction the situation is slightly better as 54% of the 1-1 aligned tokens has the correct head.

4.4.3 Different Annotation Style

We have discussed the problem that if two similar languages are used and the same annotation guidelines are used, the parsers for the two languages will often make the same kind of mistakes or at least make mistakes in the same places. In work on joint or bilingually informed parsing it is also often noted that it is the difference between the two languages that leads to improvements. For instance that pp-attachment is ambiguous in English but not in Chinese (Chen, Kazama, and Torisawa, 2010). While we cannot use different languages when parsing Danish-English, we could try to use different structures on one of the languages to see if the structural divergence that arises from this will help the parsers. Earlier, we have seen (section 4.3.2) that the best input to an extended parser is the output of a target language parser and not the gold-standard trees on the target language. This implies that we do not need a parallel treebank with different annotation styles on the two languages to perform this experiment - we only need two different parsers.

To try this approach we use data from the Penn Treebank (Marcus, Santorini, and Marcinkiewicz, 1994). We add noun phrase-structure as described (and implemented) by Vadas and Curran (2007). We then convert the treebank to dependency notation using 'The LTH Constituent-to-Dependency Conversion Tool for Penn-style Treebanks' [2] (Johansson and Nugues, 2007).

To make a more reasonable comparison we randomly select sentences from the PTB to make the training set the same size as the set we use from the CDT (3,333 sentences). The annotation styles of the two treebanks are very different. We trained a parser on the PTB-dependency data and applied it on the CDT data. The UAS was below 50%. There are two reasons for this. One is of course the difference in annotation style. The other is that the parser will parse data that at least to some degree is out of domain. However, this guarantees divergence between the structures on the Danish

[2] http://nlp.cs.lth.se/software/treebank_converter/

side and on the English side which we use for input to the extended Danish parser. Table 4.9 shows the results from this experiment. We see that using the English data parsed with the PTB-style annotation is not helpful for the extended parser - it actually decreases accuracy slightly. The conclusion from this small experiment is that it is not necessarily better to use structures with larger divergence.

	UAS
Baseline	88.79
CDT-input	88.79
PTB-input	88.42

Table 4.9: Scores of extended parser with different structures in the extended input.

4.4.4 Different Parsers

In this section we will again address the problem that because the same parsers and the same annotation guideline is used for parsing both languages, the errors in the parser output may be in the same places on the two languages. In the section above we tried varying the annotation guideline for one language by training a parser on completely differently annotated data.

In this section we will try to use another parser to parse one language. It is well known that different kinds of parsers make different kinds of mistakes (McDonald and Nivre, 2011). This means that a parser can possibly benefit from the output of another parser. This is often called *stacked* parsing and has been shown to work well (Nivre and McDonald, 2008; Martins et al., 2008).

What we propose here is a kind of stacking, where the input to the parser is on a different language than the language it is parsing. In some sense bilingually informed parsing is just stacked parsing, but it is more difficult as alignment is not trivial.

We use the MaltParser described in section 3.3.1 as the parser on the target language. We do not perform the experiment in the opposite direc-

tion as we have not implemented a version of the MaltParser that can do extended parsing. Both the training data and the test data is parsed with the MaltParser. The training data is made with 10-fold jack-knifing as in the other experiments.

Stacking parsers on one language has been shown to provide improvements, so we need to consider if any improvement in our experiment is simply due to the stacking and not the 'bilingual stacking', so we also try using the output from the MaltParser on one language to train an extended parser on the same language. This is stacked parsing, but we use the same features we use for the bilingual parsing (all alignments will of course be 1-1 so only the **head1-dep1** feature will be used).

	Danish	English
Extended	88.79	85.72
Stacked	89.44	84.57
Extended, Malt input	88.39	86.05

Table 4.10: UAS when using MaltParser output as input to the extended parser.

Table 4.10 shows the results. We see that the results are quite contradictory. For Danish, stacked parsing is helpful but not extended parsing with MaltParser input. For English, the result is the opposite. As the results are inconclusive we will not pursue this direction further.

4.4.5 Errors From Extended Parsers

We saw in the baseline results for the extended parser that the extended parsers yielded better results than the baseline parser for English, but not for Danish. The output from the standard Danish parser is not the same as the output from the extended Danish parser, although the accuracy is the same. This implies that the extended parser makes both good and bad changes compared to the standard parser. This is probably also the case for the extended English parser, although the overall accuracy is better. In the following we will investigate this further.

Total	8,681			std true	ext true
Diff	512			6.00%	36.28%	40.69%
not aligned				6.91%	47.22%	36.11%
aligned to one				84.84%	35.97%	40.95%
	aligned token has true head			49.10%	20.74%	66.36%
	head is root			1.81%	12.50%	62.50%
	head not aligned			2.26%	30.00%	40.00%
	head aligned to one			79.41%	36.71%	39.32%
		head aligned to true head		36.71%	7.58%	91.97%
		head aligned to head of aligned token		65.81%	38.10%	47.61%
aligned to many				8.25%	30.23%	41.86%
	head is root			9.30%	25.00%	75.00%
	head not aligned			0%	-	-
	head aligned to one			69.77%	40.00%	50.00%
	connected			65.12%	25.00%	50.00%

Table 4.11: Error analysis on extended Danish parsing.

Quantitative Analysis

Table 4.11 and 4.12 show some statistics from the output of the standard parser and the monolingual parser on the development data. In table 4.11 we see that when there is a difference between the standard and the extended parser, it is more common that the extended parser makes the correct analysis than it is that the standard makes the correct analysis. This does not seem to match with the previous evaluation, which showed that the extended parser was not more accurate. The only explanation is that the main part of the difference is in non-scoring tokens (punctuation) and that the rest is too little to change the overall accuracy[3].

The tables give an overview of the situations where there is a difference between the output of the standard parser and that of the extended parser. We see that although the extended parsers are correct in more cases than the standard parsers, they actually make a lot of wrong decisions as well. In general, the distributions over the two classes are the same for the differ-

[3]Evaluation with punctuation confirms this. UAS for baseline parser is 87.29 and for the extended parser it is 87.56.

Total	9,464		std true	ext true
Diff	901	9.52%	30.41%	44.62%
not aligned		4.44%	40.00%	30.00%
aligned to one		92.34%	29.93%	45.43%
	aligned token has true head	65.14%	19.19%	65.31%
	head is root	3.61%	30.00%	63.33%
	head not aligned	2.16%	16.67%	38.89%
	head aligned to one	90.38%	30.19%	44.95%
	head aligned to true head	51.99%	15.35%	78.26%
	head aligned to head of aligned token	77.79%	29.57%	51.79%
aligned to many		3.22%	31.03%	41.38%
	head is root	3.45%	100.00%	0%
	head not aligned	0%	-	-
	head aligned to one	89.66%	30.77%	46.15%
	connected	79.31%	26.09%	47.83%

Table 4.12: Error analysis on extended English parsing.

ent situations we look at. There are some notable differences though. There is one situation where the standard parser is better[4]. This is when the dependent token on the source side is not aligned. In this case, the extended parser can of course not get any help from the target side, but there is no reason that is should do worse than the standard parser. These results suggest that the non-extended part of the model turns out worse than in the standard case.

The two other categories that differ most from the overall distribution are 'aligned to one - aligned token has true head' and 'aligned-to-one - head-aligned to one - head aligned to true head'. In these categories, the extended parser is much better than the standard parser. This is not surprising as these are situations that are indicative of good input. In the first, the analysis of the token aligned to the source-side dependent is correct. In the second, the head is aligned to the true head of the token aligned to the source side dependent.

[4]Excluding the 'aligned to many' - 'head is root', which we exclude because there is only one example.

Although the analysis only gives a very coarse view of what is happening in the extended parsers, it does show that the extended parsers work as we expect - when the input is good so is the output. Apart from this conclusion, it is difficult from this analysis to identify situations that can help us in designing additional features.

Qualitative Analysis

In the following, we present a qualitative analysis of the errors made by the extended parser and not by the standard parser. We do this by analyzing situations where the extended parsers make errors and the standard parsers do not, to investigate why this happens. We hope that this analysis will help us to design features to help prevent these errors. This can be seen as a conservative way of increasing the quality of the output of the extended parser. Instead of trying to get it to make more of good changes compared to the standard parser, we focus on how to help it make fewer incorrect changes.

We will show some examples to illustrate the configurations we are discussing. In these examples, the structure at the top will be the output of the extended parser and the structure at the bottom will be the extended input to the extended parser. Dependency arcs that are incorrect are drawn with dashed lines. The token with the vertical lines around it is the central token in the analysis.

Prepositions and Punctuation

Prepositions and punctuations are overrepresented when looking at the dependents that are incorrect in the output from the extended parsers, compared to the output from the standard parsers. These are high frequency words that often carry little meaning, and are often considered difficult to align correctly. There will often be more than one punctuation token in a sentence, which can make it difficult to pick the correct one. With respect to the prepositions, these are often part of 1-n, m-1 or n-m alignment - also making it difficult to align them correctly. Figure 4.13 shows an example where the extended parser makes an error involving a preposition. We see that ”foran” gets the wrong head. There is no clear indication why this is the case, but as said, we see an overrepresentation of prepositions and

punctuation when looking at the errors.

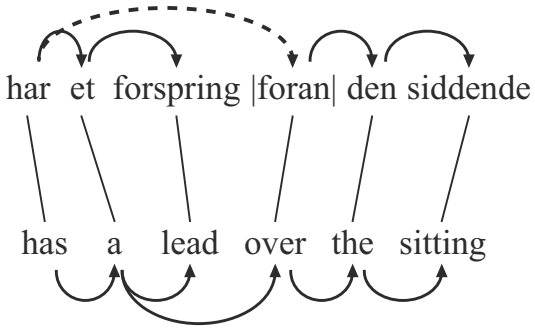

Figure 4.13: Error from extended parser involving a preposition.

Head and Dependent Aligned to Same

Situations where the head and dependent on the source side are aligned to the same token on the target side also seem to be overrepresented when looking at the errors from the extended parser. Figure 4.14 shows an example of this. We have no really good reason why this causes errors in the extended parser.

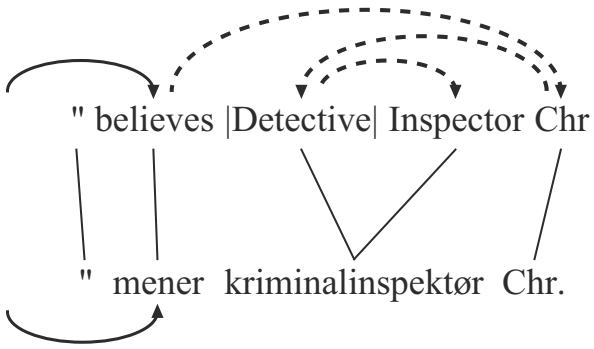

Figure 4.14: Error from extended parser involving head and dependent aligned to the same token.

Wrong Input

Most of the errors are caused by the parser being misguided by either wrong alignments or a wrong analysis on the target side. To reduce the number of variables in our analysis we have tried redoing it with gold-standard alignments instead. The biggest source of errors after this is a

wrong analysis on the target side language. Figure 4.15 shows an example of this.

To remedy this, one would have to be able to do some kind of prediction on how likely it is that the target side analysis is correct. This is almost parsing, as it requires predicting how likely a dependency arc is, so this really reduces to making the target side better. An alternative would be to make some soft-link features that return some score to indicate how likely the target side analysis is, given the target side model. We have not pursued this idea further.

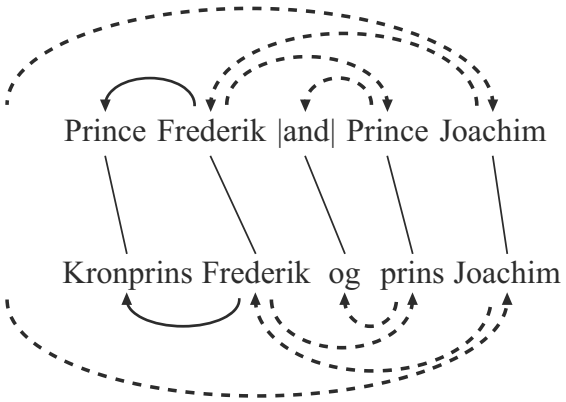

Figure 4.15: Error from extended parser involving a wrong analysis on the target side.

n-1 Alignments

We could identify one other major source of errors in the extended parser - i.e. situations where the monolingual parser makes the correct analysis, and the extended parser does not. These are situations where the parser is misguided by n-1 alignments. Figure 4.16 shows an example of this.

If the possible dependent is part of such a feature, the **head1-dep1** and **headn-dep1** -features are activated, but the information from the target sentence is a lot less reliable if the target-side token is also aligned to more words in the source sentence.

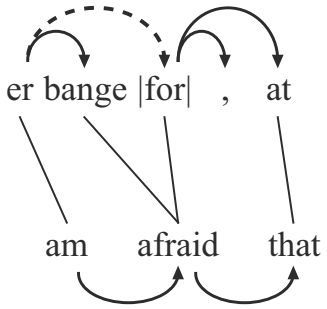

Figure 4.16: Error from extended parser involving a n-1 alignment.

4.5 More Features for Extended Parsing

The analysis of correspondence and the analysis of the errors from the extended parsers points to some configurations that could be helpful for the parser to identify. This leads us to introduce new features that are designed to avoid the errors.

4.5.1 Correspondence

Based on the analysis in section 4.4.2, we include features that indicate whether the current possible edge is part of the additional configurations described:

p-true The dependent is aligned to exactly one token, the head is aligned to (only) the same token and the head of the head is aligned to (only) the head of the token the dependent is aligned to.

p-fuzzy The dependent and head are aligned to the same token, and there exists a relation between at least one of the tokens the dependent is aligned to and at least one of the tokens the head of the head is aligned to.

4.5.2 2-1 Alignment

We also try to capture the more general 2-1 configurations with the following feature.

2-1 Head and dependent token are aligned to the same token, and only that token, and this token is only aligned to these.

4.5.3 Prepositions and Punctuation

We saw in section 4.4.5 that certain word classes seemed to give the extended parser more problems than others. In order to deal with this, we introduce the possibility of combining all the other extended features with the PoS-tag of the dependent, with the PoS-tag of the head and with this PoS-tag for both. Hopefully, this can help the extended parser to trust the extended input more with certain word-classes than with others.

4.5.4 Head and Dependent Aligned to Same

To reduce the number of errors related to the problem of the head and dependent being aligned to the same token, we introduce the following features.

same Head and dependent token are aligned to the same token, and only that token. The aligned token can be aligned to any number of tokens.

same-fuzzy Head and dependent token are aligned to the same token, and at least one of them is also aligned to another token. The aligned token can be aligned to any number of tokens.

This **same**-feature is very similar to the **2-1**-feature but allows the target side token to be aligned to other tokens as well. This make the **2-1**-feature a special case of the **same** feature.

4.5.5 n-1 Alignment

The error analysis suggests introducing features to help the parser to avoid being misguided by n-1 alignments. If the possible dependent is part of such a feature, the **head1-dep1** and headn-dep1-features are activated. The following features help prevent errors in configurations like these.

depn1 **headn-depm** and dependent token is part of n-1 alignment.

headn1 **headn-depm** and head token is part of n-1 alignment.

depheadn1 **headn-depm** and both dependent and head token are part of (possible different) n-1 alignments.

The **depheadn1**-feature is different than **same**-feature and the **2-1**-feature because it requires that there is a relation between tokens the head and dependent are aligned to. This is not possible with the two other features as the head and dependent are aligned to the same token in these.

4.5.6 Empirical Evaluation of Features

We have introduced features based on different analyses of the data. Although we have good reasons to believe that these features should help the parsers and increase the quality of the output from these, this has to be tested empirically.

Table 4.13 shows the results from the baseline parser, from the extended parser and from the extended parser with each of the new features discussed here. The variation in the accuracy on the development data seems

	Development		Cross-validation	
	Danish	English	Danish	English
baseline	88.79	84.21	87.24	83.06
extended	88.79	85.72	87.72	84.85
+ PoS	88.78	86.17	87.71	85.12
+ same	88.93	85.89	87.69	85.14
+ same + PoS	88.83	85.99	87.68	85.20
+ p	88.48	86.09	87.59	85.02
+ p + PoS	88.55	85.84	87.60	85.14
+ n-1	**89.53**	86.06	87.67	85.05
+ n-1 + PoS	88.89	85.88	87.59	85.12
+ 2-1	88.87	85.99	87.57	85.06
+ 2-1 + PoS	88.83	86.07	87.65	85.18

Table 4.13: UAS with simple features.

somewhat random. For instance, the addition of PoS-tags does not have a large effect in most cases, but with the n-1 feature it leads to a large drop in accuracy. Because of this we have tested all the additional features with 10-fold cross-validation as well. Here we see a smaller variation, which leads us to believe that these results are more reliable.

Earlier, we saw that the extended parsing did not help on Danish. When using cross-validation, the result is different. Here extended parsing helps, but on the other hand none of the additional features help. In English, all additional features increase accuracy. In English, combining the features with their PoS-tags consistently improves accuracy. In Danish, there is no clear tendency with respect to this.

In this first evaluation we look at each of the features in isolation, but of course we need to see if they will work when using more features at the same time. Table 4.14 shows results when the features are used together. In general, there is no clear benefit from combining the features, but the highest accuracies are found with combined features. In Danish, the best combination of features is 'same + p + n-1'. In English, there are two feature combinations that has the highest score, '+ p + 2-1 + PoS' and '+ same + n-1 + 2-1 + PoS'. We let the results on the development data break the tie and assume that '+ same + n-1 + 2-1 + PoS' is the best combination of features to use.

These feature combinations will be used in all subsequent experiments involving extended parsing on Danish and English.

Table 4.15 summarizes the results from the test of features and shows results from significance results when comparing the simple extended features to the ones we found to perform best.

4.5.7 A Note on PoS-Tags

We have seen that the baseline parser performs better on Danish, most likely because of gold-standard PoS-tags and also that the extended parser improves much more on English than on Danish. We believe that this is basically because the Danish baseline parser is better. To test it, we try to use non-gold-standard PoS tags on Danish. We do not have a PoS-tagger available for Danish so we create the tags using 10-fold jack-knifing with

	Development		Cross-validation	
	Danish	English	Danish	English
baseline	88.79	84.21	87.24	83.06
extended	88.79	85.72	87.72	84.85
+ same + p	88.74	85.77	87.74	85.14
+ same + p + PoS	88.81	85.95	87.71	85.21
+ same + n-1	88.75	86.07	87.73	85.18
+ same + n-1 + PoS	88.99	86.04	87.46	85.16
+ p + n-1	89.02	85.98	87.66	85.06
+ p + n-1 + PoS	89.07	85.93	87.62	85.12
+ same + 2-1	89.29	86.01	87.70	85.18
+ same + 2-1 + PoS	88.90	85.89	87.70	85.20
+ p + 2-1	88.74	86.01	87.74	85.05
+ p + 2-1 + PoS	88.50	85.71	87.63	**85.24**
+ n-1 + 2-1	89.15	85.99	87.60	85.18
+ n-1 + 2-1 + PoS	88.78	85.92	87.57	85.12
+ same + p + 2-1	88.75	86.16	87.73	85.11
+ same + p + 2-1 + PoS	88.93	85.99	87.58	85.18
+ same + n-1 + 2-1	89.02	**86.27**	87.65	85.13
+ same + n-1 + 2-1 + PoS	88.91	86.12	87.64	**85.24**
+ p + n-1 + 2-1	89.02	86.15	87.65	85.06
+ p + n-1 + 2-1 + PoS	88.83	85.99	87.69	85.23
+ same + p + n-1	89.13	86.17	**87.81**	85.09
+ same + p + n-1 + PoS	89.02	85.98	87.59	85.17
+ same + p + n-1 + 2-1	88.85	85.98	87.64	85.14
+ same + p + n-1 + 2-1 + PoS	89.05	86.04	87.69	85.21

Table 4.14: UAS with combined features.

the SVMTool-tagger[5] (Giménez and Màrquez, 2004).

In the experiment, we exclud all information in the treebank except the PoS tags, which is why the English baseline results also change. Table 4.16 shows baseline results and results with the simple extended features. We

[5] http://www.lsi.upc.edu/~nlp/SVMTool/

	Development		Cross-validation	
	Danish	English	Danish	English
simple	88.79	85.72	87.72	84.85
best	**89.13**	**86.17**	**87.81**	**85.24†**

Table 4.15: Results with simple features and best features for extended parsing.

see that the Danish parser is still better than the English, but the difference is a lot smaller. We also see that with the extended parsing, the Danish parser now improves on the baseline, and that the English improves less than it did with gold-standard PoS tags. This confirms that the reason the English extended parser works better than the Danish, is simply that the input it gets, i.e. the output from the Danish parser, is better.

PoS-tags		Danish	English
Gold	Baseline	88.79	84.21
	Extended	88.79	85.72
	Difference	0	1.51
Non-Gold	Baseline	85.00	83.83
	Extended	85.72	84.64
	Difference	0.72	1.01

Table 4.16: Results for extended parsing with data sets with non-gold PoS-tags.

Chapter 5

Joint Models

In the previous chapter we saw how bilingually informed parsing could improve parsing accuracy, and earlier we saw how sub-tree alignment could improve alignment accuracy compared to word alignment. The models used in bilingually informed parsing rely on bilingual data, but not on each other. The model used for extended parsing on Danish is independent of the model used for extended parsing on English. In this chapter we will present two approaches where the models are not independent of each other. The hope is, that this will lead to even better results as the bilingual information can be used even better.

We use the term ”joint models” to describe approaches where the models for the different sub-structures are dependent of each other. The actual processing of the sub-structures is not necessarily done simultaneously.

5.1 An Iterative Approach

In this section we will describe an approach to bitext dependency parsing which we call the *iterative* approach.

There are three basic assumptions behind this approach. The first two are that bilingually informed parsing is better than standard parsing, and that sub-tree alignment gives better results than word alignment. We have seen that both of these assumptions hold. The extended parsers are better than the standard parsers and the aligner is better with trees in the input than without. The third assumption is that the higher the quality of the

input is, the higher the quality of the output will be.

5.1.1 Better Input Makes Better Output, Random Errors

Before we turn to the describing the approach we will investigate the assumption that better input leads to better output in the case of extended parsing.

To test the assumption we introduce some random errors in the data to simulate different levels of accuracy of the parser used on the parallel sentences. More concretely we run through the English train and development data and randomly introduce wrong dependencies with a certain probability. We do so on both train and development data so the number of errors in the input at train and test time is the same. The hypothesis is then, that as the number of errors in the English data decrease the accuracy of the Danish parser that uses this as input will increase. Figure 5.1 shows that this is indeed the case. In this case the alignments used are gold-standard alignments.

We actually see that using data with random errors does a lot better than using actual output from parsers. For instance with an error rate of around 20% we see that the parser is 4 points better than the baseline. This is far more than the gain we get when using parser output where we saw no increase in accuracy for Danish. The reason for this, we believe, is that when using randomly created wrong edges, the chance of the extended parsing actually changing the results of the parser is much smaller than when using output from a parser. We believe there is one main reason for this. There is a good chance that the randomly created relation is very unlikely. This means that a relation on the source side matching this will have a very low score. The relation is simply not competitive with the correct edge so even though the extended features fire for this relation, it will not receive high enough score to be chosen. When the score of the source side edge is high and it matches a target side edge, this edge is probably not incorrect.

We see that the assumption holds when synthetic data is used, but we do not know for sure if the assumption also holds for real data.

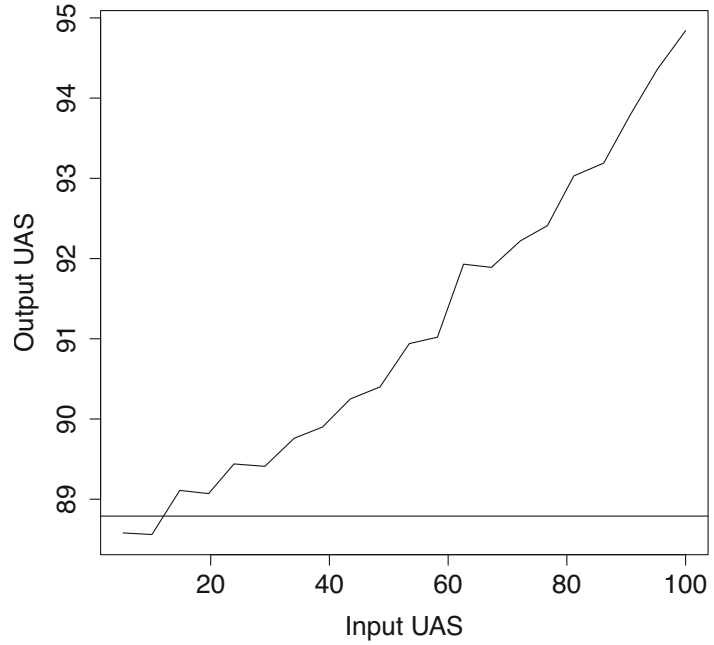

Figure 5.1: Effect of quality of input on quality of output in extended parsing. Based on synthetic data with random errors. The horizontal line shows the accuracy of the baseline parser.

5.1.2 Basic Iterative Approach

In the iterative approach the different sub-tasks of bitext parsing and alignment will inform each other in an iterative fashion. The idea is, that as the input to one sub-task gets better, so does the output. And as this is the input to another sub-task, the output of this sub-task and so on. Of course the quality will stop increasing at some point, but hopefully the quality at this point will be better than simply doing the extended parsing once.

The tools needed to use the iterative approach are extended parsers and a sub-tree aligner.

The iterative process can begin with either parsing or alignment. He we will consider the case where parsing is performed as the first step.

The process will start with the training of a standard monolingual parser for each language and an extended parser for each language. The standard parsers will then be used to parse the unseen data. Furthermore a sub-tree

aligner is trained. This aligner will then be used to align the unseen data using the output from the parsers as input.

With these things in place the actual iterative process can begin. Together the output from the parsers and the aligner provide the input for doing extended parsing. So now the extended parsers will be applied on the unseen data using the output of the two other sub-tasks as input. We choose to run the parsers in parallel so in each iteration the trees from the previous iteration are used. The alternative would be to first do one language and then use this as input for the other.

Algorithm 3: Iterative - no retraining

Data: *trainA, trainB, extTrainA, extTrainB, trainAB, test*
Result: *test* parsed and aligned
train parser A_0 on *trainA*; train parser B_0 on *trainB*;
apply A_0 on *testA* $\rightarrow parsedA_0$; apply B_0 on *testB* $\rightarrow parsedB_0$;
train ext-parser A'_0 on *extTrainA*; train ext-parser B'_0 on *extTrainB*;
train aligner AB_0 on *trainAB*;
for $i \leftarrow 1$ **to** $maxIter$ **do**
  apply AB_{i-1} on test with $parsedA_{i-1}$ and $parsedB_{i-1}$ as input $\rightarrow$ $alignedAB_i$;
  apply A'_{i-1} on $alignedAB_i \rightarrow parsedA_i$;
  apply B'_{i-1} on $alignedAB_i \rightarrow parsedB_i$;
end

The number of iterations can either be given in advance or a stop-criterion can be used. Algorithm 3 describes the iterative approach with a fixed number of iterations.

Figure 5.2 shows the scores from the output after each iteration when using the iterative approach. The best features found in section 4.5 are used.

We see that overall the approach does work. The largest improvement is in the first iteration but there are small improvements after this. Interestingly, we also see that the accuracy goes up and down periodically, and that this change is shifted between Danish and English. In one iteration the accuracy of the English parser drops, then in the next the accuracy of the Danish parsers drops and then the English drops and so on. This seems to

confirm the third assumption - better input means better output. When the English output gets worse, the input to the Danish parser in the next iteration gets worse, which leads to worse Danish output and so on. Shifted from this we see the opposite effect. When the English output gets better, the Danish gets better in the next iteration and so on. The alignment output gets slightly worse compared to the simple sub-tree alignment (iteration 0).

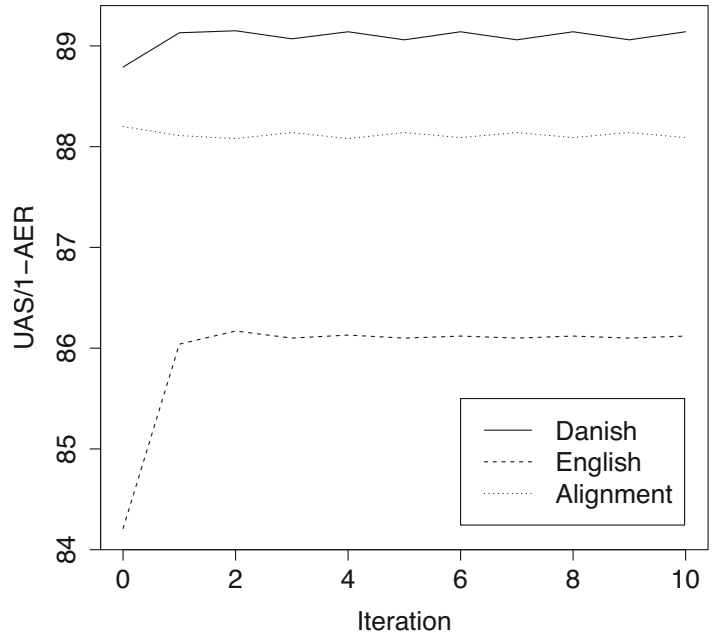

Figure 5.2: Result per iteration with the basic iterative approach.

5.1.3 Iterative With Validation

For the iterative approach to work it is necessary that the quality gets better after each iteration, and this is of course not guaranteed to be the case. If the quality of the output decreases in an iteration this will hurt the parser on the other language in the next iteration. Figure 5.2 shows an example where this happens and leads to the output of the parsers going up and down.

To avoid this, we use a validation set. After each iteration the validation

set is parsed and only if the accuracy on this increases compared to the best parse so far do we use this as input in the next iteration. This also provides a natural stop-criterion. When neither the accuracy of the parsers nor the accuracy of the aligner on the validation data increase, the algorithm stops.

Figure 5.3 shows the accuracy after each iteration using this approach. In principle it works as we do not see any decreases in accuracy. But it stops very early and for English it does no continue after the initial extended parsing. We also see that the best score from the approach without validation was actually higher than the best scores using this approach.

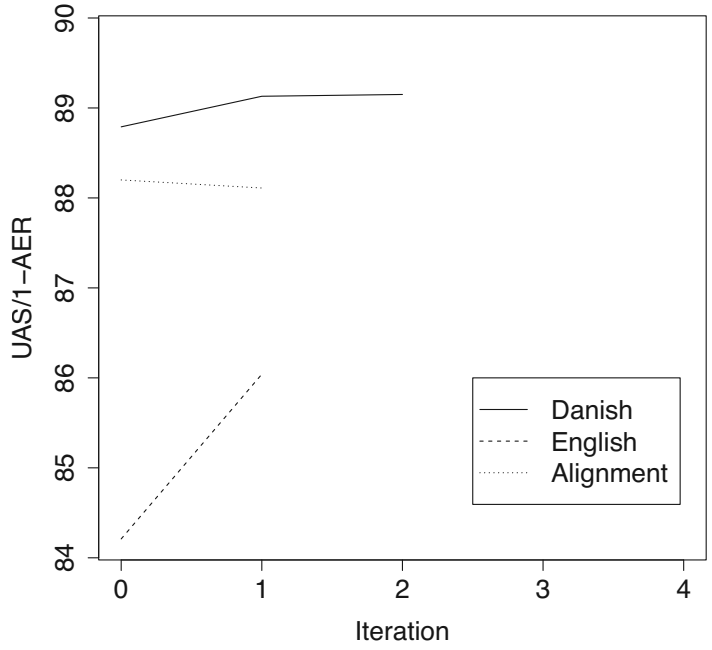

Figure 5.3: Result per iteration with the iterative approach with validation.

5.1.4 Iterative With Retraining

In the approach described above the only thing that changes after each iteration is the input data to the extended parsers and the input to the aligner. In the experiments with the training data (sections 3.3.2 and 4.3.2) we saw the importance of the training data matching the test data. If we train on

gold-standard data and then use non gold-standard data at test time we will not get good results. This also points to a possible problem with the iterative approach as described above. The models are static and therefore reflect the quality of the data used for training them. This data was created using jack-knifing of a standard parser. As the quality of the input data at test time hopefully increases as a result of the iterative approach, there will be a gap between the quality of the training data used for the models and the input data at test time. We will now describe an approach that tries to deal with this problem.

We want to make the quality of the training data match the quality of the test data. To do this we need to retrain the models in each iteration. We cannot use the trained model that we use when parsing the test data, to parse the training data and then use this as input when training the extended parser in the other language, because the quality of the parses on the data used for training will be too high. To deal with this we can use jack-knifing in the same way we used it for creating the original training data. This means splitting the training data for the extended parser into n parts and then training n parsers on $n-1$ parts and use each of these for training the held out part. We choose a similar approach that leads to a little less retraining. In each iteration we choose $n-1$ parts for training and 1 part as a left-out part. We then train the parsers on the $n-1$ parts and parse the left-out part. If the new parses of the left-out part is better than the previous parses on this part, we replace the old parses with the new. This means that we update the training data for the extended parser on the target language if the parses on the source languages gets better and vice versa. In each iteration we also train a model on all the parts and use this to parse the test data. The idea is as described above. In each iteration the quality of the left-out part should increase which leads to an increase in the quality of the training data for the extended parser in the next iteration. And hopefully this retraining of the model leads to a better correspondence between the model and the quality of the input to the extended parser at test time.

This method has the validation-approach described above build-in. After each iteration we only update the parses on the left-out part if they are better than the best parses so far. We also only update the test data input if

the left-out part improves. Algorithm 4 describes the algorithm. We have left out the alignment part of the algorithm for clarity.

Algorithm 4: Iterative - with retraining

Data: *trainA, trainB, trainAB, extTrainA, extTrainB, trainAB, testA, testB, testAB, testAparsed, testBparsed*
Result: *test* parsed and aligned
split *trainA, trainB, extTrainA, extTrainB* into n parts;
for $i \leftarrow 1$ **to** $maxIter$ **do**
  for $leftOut \leftarrow 1$ **to** n **do**
    $leftOutA = leftout$-part of *trainA*;
    $leftOutB = leftout$-part of *trainB*;
    train extended parser on $n-1$ parts of *extTrainA* $\rightarrow modA_i$;
    train extended parser on $n-1$ parts of *extTrainB* $\rightarrow modB_i$;
    train extended parser on *extTrainA* $\rightarrow modTA_i$;
    train extended parser on *extTrainB* $\rightarrow modTB_i$;
    apply $modA_i$ on $leftOutA \rightarrow parsedA_i$;
    apply $modB_i$ on $leftOutB \rightarrow parsedB_i$;
    apply $modTA_i$ on *testA* with *testBparsed* as input $\rightarrow$ $parsedTA_i$;
    apply $modTB_i$ on *testB* with *testAparsed* as input $\rightarrow$ $parsedTB_i$;
    if $parsedA_i$ *better than* $leftOut$ *part of* extTrainB **then**
      update *extTrainB* with $parsedA_i$;
      testAparsed = $parsedTA_i$;
    end
    if $parsedB_i$ *better than* $leftOut$ *part of* extTrainB **then**
      update *extTrainA* with $parsedA_i$;
      testAparsed = $parsedTA_i$;
    end
  end
end

Figures 5.4 and 5.5 show the accuracy after each iteration of both the data used as input to the extended parsers and the accuracy of the output. We

see that the accuracy on the training data increases consistently. The biggest increase is in the first 10 iteration where baseline parser output is replaced with extended parser output. But also after iteration 10 we see improvements. The accuracy is monotone as we only update the training data if the accuracy on the left-out part increases.

Unfortunately, the correspondence between the accuracy of the training data and the output from the extended parsers are difficult to see. We do not see any consistent increase in accuracy on the test data. The best results of all iterations (Danish, 89.26 and English 86.41) are actually better than the best results from the other iterative approaches (Danish 89.15 and English 86.17), but it seems very difficult to predict in advance which iteration will yield the best results.

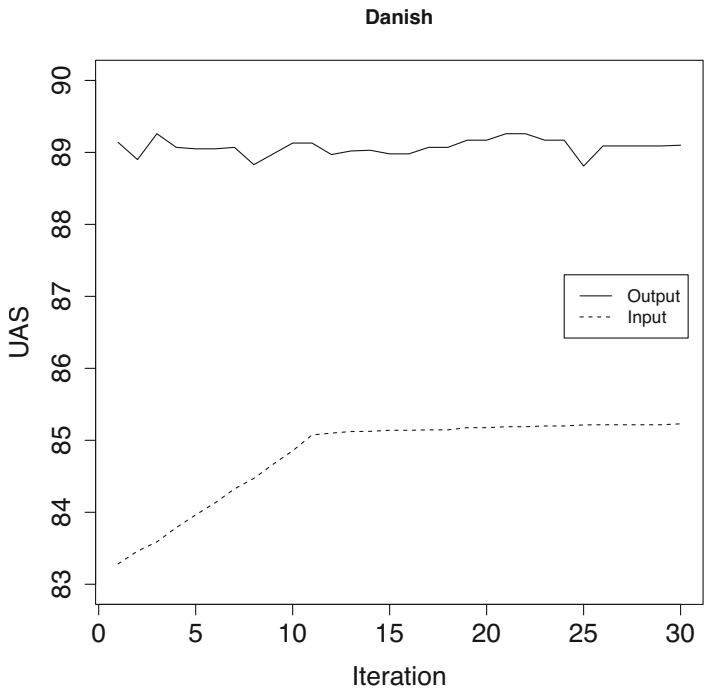

Figure 5.4: Accuracy of input and output of Danish extended parser in the iterative-with-retraining approach.

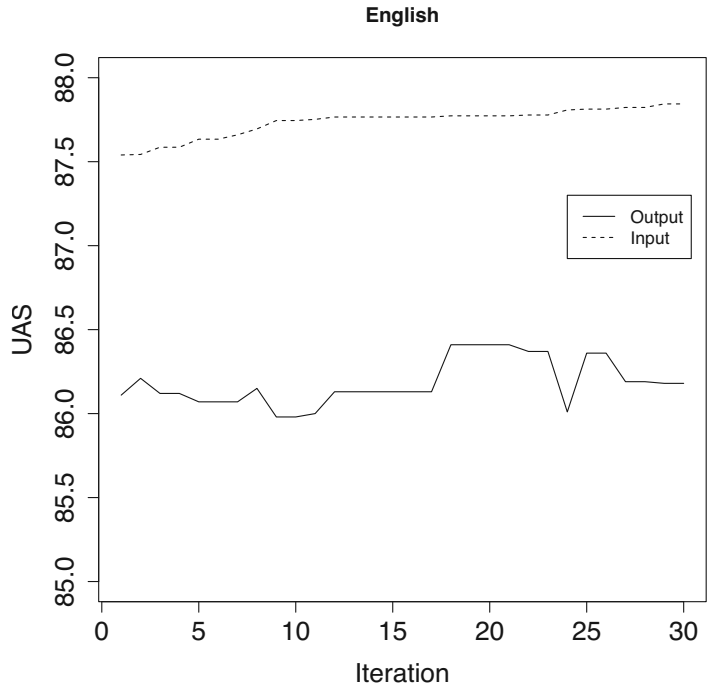

Figure 5.5: Accuracy of input and output of English extended parser in the iterative-with-retraining approach.

5.1.5 Sizes

In section 4.3.3 we investigated the influence of the size of the training data on the extended parsing. In this section we will do the same for the iterative approach. We will leave out the initial approach without validation, because it is difficult to know which iteration to use. We will not perform the experiments for the 'with-retraining' approach. We saw that the lack of correspondence between training data and test data seems to cause problems in this approach. This problem will only be more severe with less training data, so we choose to skip these experiments.

With validation

Figures 5.6 and 5.7 show the results from running the iterative approach with validation for different subsets of the training data. The figures show the relative UAS compared to the baseline parser. We see that especially for Danish the improvement is bigger for smaller training sets. We see though

that the biggest improvement is still in iteration 1, so the question remains if it is only the extended parsing part of the approach that works better, or if it is actually the iteration-part of it that improves the results.

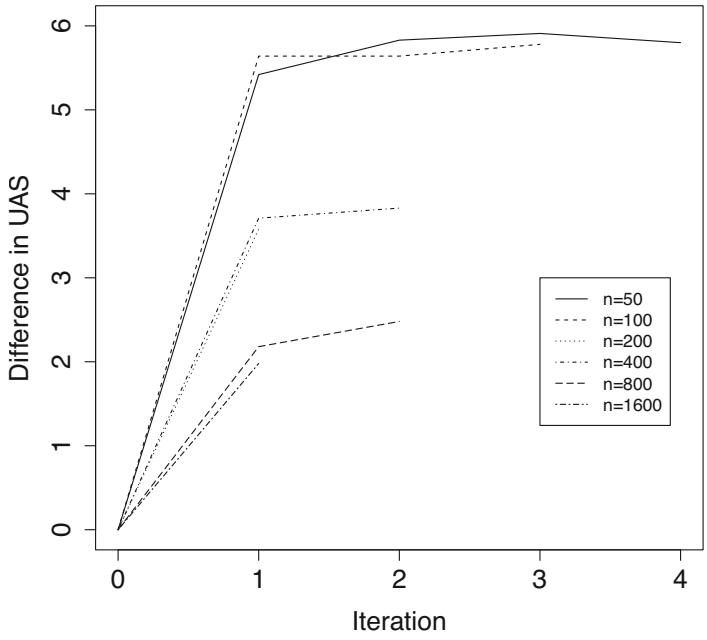

Figure 5.6: Relative UAS per iteration for different training set sizes (Danish).

Figure 5.8 shows the UAS of the iterative approach relative to the extended parser for different training set sizes. In general there are small improvements, and it seems to be the case that the iterative approach works better with small training sets.

5.2 Reranking

Reranking is a method often used in NLP. The idea behind reranking is that we have some method for doing some task that outputs a list of hypotheses for each input it receives. Subsequently, a reranker is applied to (re)rank the hypotheses, and we can then use the best one (if we are only interested in one). The reason why this makes sense is because the initial methods used often require some restriction of the feature space. We saw this both

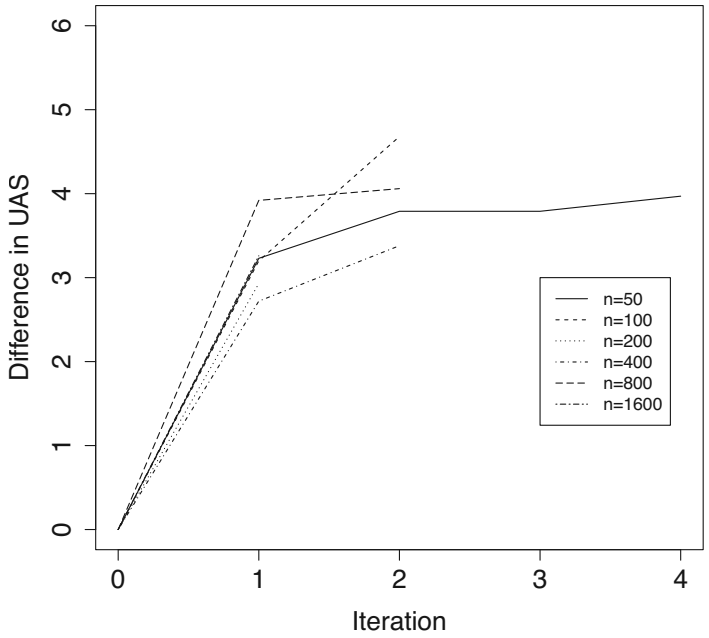

Figure 5.7: Relative UAS per iteration for different training set sizes (English).

for dependency parsing and alignment. A reranking model will not be restricted by this, and therefore features can be included in the reranking model, which cannot be used in the method that creates the hypotheses.

Reranking has been used both in parsing (Collins and Koo, 2005; Charniak and Johnson, 2005; Hall, Havelka, and Smith, 2007; Hall, 2007), word alignment (Venkatapathy and Joshi, 2007) and joint parsing and alignment (Burkett and Klein, 2008).

Here we will also try to use reranking to create better parallel treebanks but in a more classical reranking sense than the work of Burkett and Klein (2008). We will create k-best lists for parses for both languages and rerank all combinations of these in hope of obtaining better parses. We will not use k-best lists for the alignments because the aligner we use cannot provide reliable k-best lists.

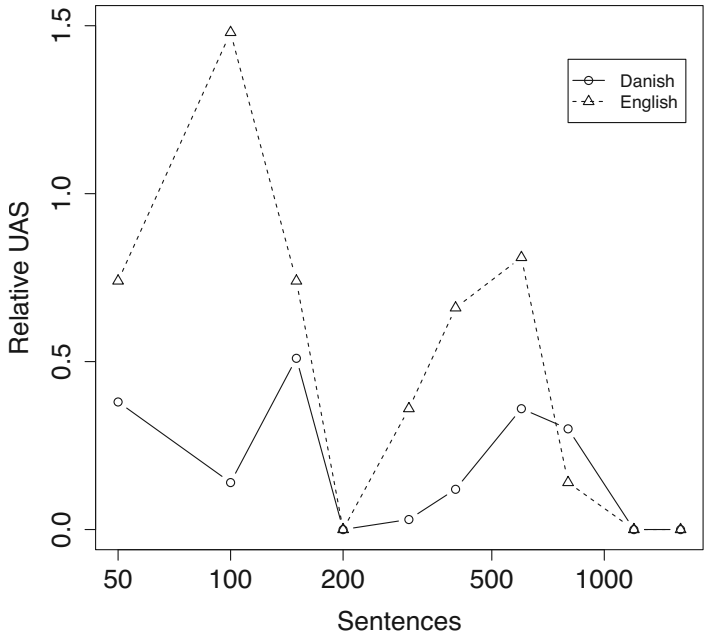

Figure 5.8: Comparison of extended parsing and iterative approach for different training set sizes.

5.2.1 Features

We use the scores of the kMST-parser as features. These scores from the parser are log-scores, and we use both the non-log version of these and normalized log-scores.

Apart from the scores from the parser we do not include any monolingual features. This is because we want to see what benefit we can achieve from using both languages. We are not interested in the possibilities of improving the output using reranking in general.

All the features used in the extended parser are also used for the reranking. These features are described in sections 4.3.1 and 4.5.

In addition to these we use the following features:

non-consistent If token s is aligned to token t, and s is the head of s_i and s_i is aligned to t_i, and the head of t_i is not t the token-pair (s, t) is considered inconsistent. The value of the feature is the ratio of inconsistent token pairs in a sentence alignment.

non-consistent-transitive If token s is aligned to token t, and s is a transitive head of s_i and s_i is aligned to t_i, and a t_i is not a transitive head of t_i, the token- pair (s, t) is considered inconsistent. The value of the feature is the ratio of inconsistent token pair in a sentence alignment.

correspondence This feature measures correspondence between the sentences and is very similar to the basic features we used for extended parsing (section 4.3.1). Two aligned tokens correspond if their heads are also aligned. The features are divided into 6 sub-features depending on the alignment configuration. These are 1-1, 1-m, n-1, n-m, root-1, root-m where the first parts denotes the number of tokens the head on the source side is aligned to and the second part the number of tokens the dependent on the source side is aligned to. The feature values are the ratio of corresponding tokens in the sentence and are included in both direction, i.e. source-target and target-source.

5.2.2 Experiments

We use the kMST-parser described in section 3.3.1 for creating the k-best parse lists, the MCFAligner to create the alignment and the SVM$^{\text{rank}}$-tool described in section 3.3.3 to do the reranking.

In all experiments we use $k = 50$ which means that there are up to 2,500 hypotheses per sentence. This number of hypotheses creates a huge amount of constraints in the learning, and we need to reduce this number. For the 10 best parses on each language we use all combinations - i.e. 100 combination. For the rest we randomly sample 10% of the combinations. This leads to approximately 340 hypotheses per training example.

The reranker has to be trained on output from the parsers. We use 10-fold jack-knifing to create 50-best list for the training data. When we create the training data for the reranker, we need to provide a ranking for the hypotheses. This ranking will depend on the loss-function we decide to use. Because we are not reranking the alignments, we use only the loss from each parse. We do not consider one language more important than the other so for the reranking task we use the sum of these two losses as our loss [1].

[1] It is worth remembering though, that as the English parser is not as good as the Danish,

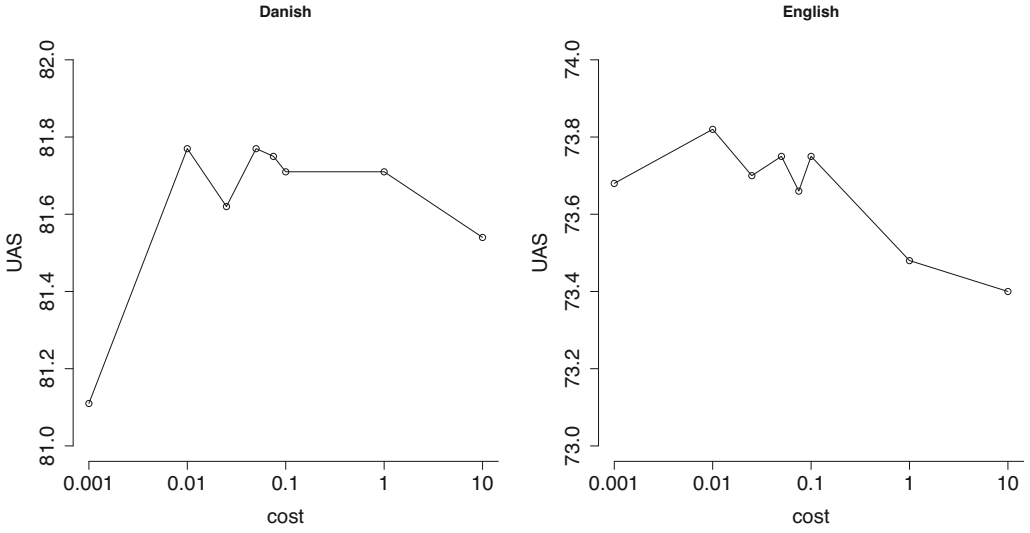

Figure 5.9: Scores on reranked output depending on cost parameter of reranker.

The results of support vector machines are sensitive to the value of the cost-parameter, so we have optimized this on a development set. Figure 5.9 shows the scores on the development sets depending on the value of the cost-parameter.

Table 5.1 shows the results from the reranking, together with the baseline results (1-best), the average score on the parses in the k-best list, and the oracle results from the k-best list. We see that the accuracy increases with about 1.5 point on Danish and 2.5 point on English. We have earlier seen that the English seems to gain more from the joint strategy than Danish and this pattern is confirmed here. Again we have to note that there is also more room for improvement in the English parses. The absolute improvements are better with this approach than the improvement we saw from the extended parser and with the iterative approach. Unfortunately, we cannot compare these results as the baseline results from the parser are much lower.

the loss from the English sentences will often be bigger than from the Danish sentences.

	Danish	English
1-best	80.14	71.08
Average (50-best)	76.90	68.63
Oracle (50-best)	87.95	79.78
Reranked (50-best)	**81.77†**	**73.75†**

Table 5.1: Results from reranking experiment on development data. Both reranked results are significant compared to the baseline.

Sizes

We also investigate how the result of the reranking approach is influenced by different training set sizes. We use the same approach for all sizes. This means that we sample even though it is not necessary with the smaller data sets and we use the cost parameter optimized on the entire data set for all sizes. Figure 5.10 shows the results from the experiments. We do not see the same pattern we did for the other approaches. If anything the effect is opposite. The improvement gets bigger with bigger data sets. A possible explanation is that the reranker overfits on the smaller data sets.

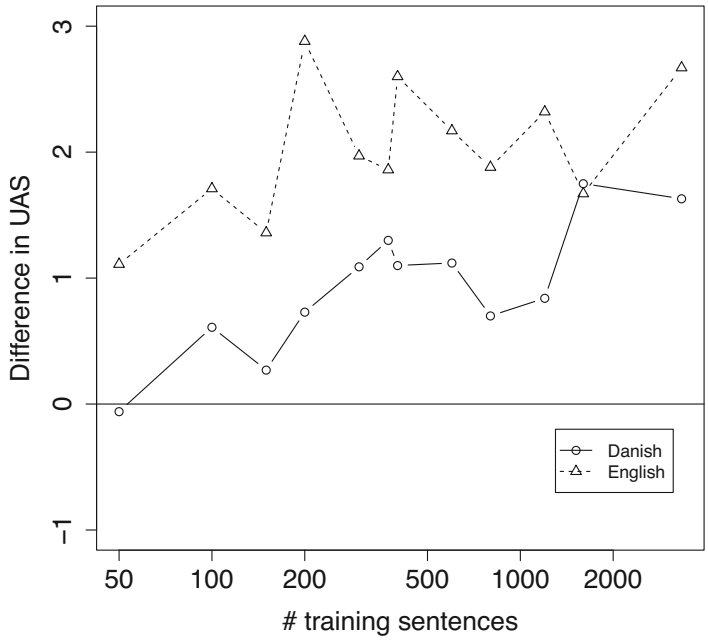

Figure 5.10: Difference in UAS between baseline parser and reranked approach depending on training set size.

Chapter 6

More Experiments

6.1 Danish-Spanish

In the previous chapters we have worked with and analyzed Danish-English data. We have discussed in some detail how the relatedness of the two languages impacts the results and the approaches. In this section we will repeat the different experiments with Danish-Spanish data. We are still working with two closely related languages, especially compared to most work on bitext parsing that works with English-Chinese, but the languages are less related than Danish and English. Furthermore the amount of data available for this experiment is much smaller than with Danish-English (see section 3.1).

6.1.1 Baseline and Extended

Table 6.1 shows results with the baseline parsers[1] and the extended parser.

6.1.2 Analysis

We will not do an error analysis on the Danish-Spanish data, but restrict ourselves to looking at correspondence, and to empirically test the features we found in the analysis of the Danish-English data.

[1] We have tested first-order and projective parsing, but also for the Danish-Spanish data set second-order non-projective parsing gave the best results.

	Danish			Spanish		
	LAS	UAS	LA	LAS	UAS	LA
MST, 2. order, non-proj	68.70	82.02	72.42	66.91	80.54	71.73
Extended, 10 fold	**71.28†**	**84.50†**	**74.28†**	**67.28**	**81.65**	**71.46**

Table 6.1: Results for baseline and extended parsing for Danish-Spanish.

Correspondence

First we will look at the 2-1 configurations, which we argued was a strong argument in favor of parsed Danish parallel text being able to help an English parser. The hypothesis was that if two English words were aligned to one Danish word there would be a relation between them. This hypothesis was not entirely correct, primarily because of the analyses used in the treebank.

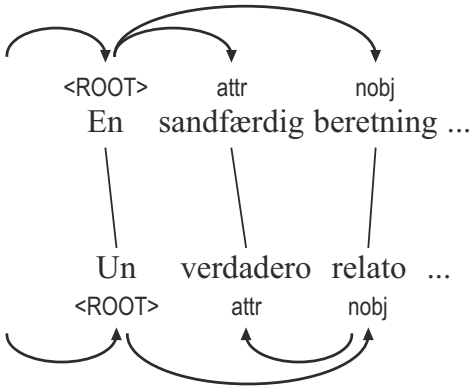

Figure 6.1: Example of different analyses in Danish and Spanish.

With respect to the compound words and determiners there is the same relation between Spanish and Danish as between English and Danish. Furthermore the analyses on Spanish in the treebank are different from those on Danish and English. Figure 6.1 shows an example of this. With this in mind we would expect that the hypothesis will actually hold for Danish-Spanish. Table 6.2 shows that this is not the case. The numbers are roughly the same as we saw for Danish-English. The only real difference is that the accuracy of the parsers on the relations in question is even higher here. By looking at the data we again see that the lack of correspondence is pri-

	2-1	Baseline UAS
Danish	91%	100%
Spanish	94%	98%

Table 6.2: Statistics for 2-1 alignments. The ”2-1” column shows part of 2-1 configurations where there is a relation between the two source side tokens. The ”Baseline UAS” column shows accuracy of baseline parser on these relations.

marily due to the choice of annotation used in the treebank. Most of the situations where the hypothesis does not hold are due to alignments like the one seen in figure 6.2, where the middle ”de” is not aligned.

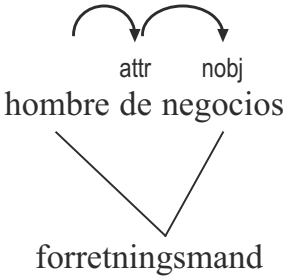

Figure 6.2: Example of Spanish-Danish 2-1 alignment where there is no relation between the two Spanish tokens.

Table 6.3 shows the numbers for the more specific 2-1 configuration described earlier. Here the numbers are lower than for Danish-English, and there is absolutely nothing to gain with respect to accuracy in these situations.

	2-1	Baseline UAS
Danish	94%	100%
Spanish	96%	100%

Table 6.3: Statistics for more specific 2-1 alignments where there is a relation between the two source side tokens, and accuracy of baseline parser on these relations.

We test for correspondence in the same way we did for Danish-English.

Table 6.4 shows results from these, including the results for Danish-English for comparison. The numbers show quite clearly that the correspondence is a lot weaker between Danish and Spanish than between Danish and English.

	Danish	Spanish	Danish	English
TRUE	32.5%	34.8%	59.7%	65.1%
P-TRUE	1.8%	6.1%	2.0%	7.9%
P-FUZZY	0.7%	2.2%	0.5%	0.8 %
FUZZY	36.5%	29.1%	28.5%	15.2%
FALSE	15.5%	20.4%	5.4%	7.9%
NEITHER	13.0%	8.3%	4.0%	3.0%

Table 6.4: Distribution of types on configuration for Danish-Spanish.

Empirical Evaluation of Features

Table 6.5 and table 6.6 show results with the different features both on development data and with cross-validation. We see a very big variance on the results on development data, and as the development set is quite small (52 sentences) these results are probably not reliable. If we look at the cross-validated results we see that most of the features improve on the basic extended features, but that there is no gain from combining features.

We see that extended parsing with the features selected from cross-validation leads to improvement in the same magnitude as we saw for Danish-English. This time it is in the opposite direction though - Danish improves more than English. The comparison is not quite fair though, as the Danish-English data set is much larger than the Danish-Spanish. If we use only a subset of the Danish-English data with the same size as the Danish-Spanish the picture is quite different. Table 6.7 shows this. We see that the improvement in Danish-English with this small data set is much bigger than in Danish-Spanish. We have not analyzed in more detail why this is the case, but it must be remembered that the features we use have been designed on the basis of the Danish-English data, and on the basis of the errors the extended parser made on the Danish-English data set. Table

	Development		Cross-validation	
	Danish	Spanish	Danish	Spanish
baseline	82.02	80.54	79.83	76.49
extended	82.85	80.91	81.02	78.78
+ PoS	83.37	**81.65**	80.28	78.59
+ same	83.47	81.19	80.28	78.59
+ same + PoS	82.85	81.09	80.46	**79.05**
+ p	82.02	80.72	80.84	78.60
+ p + PoS	83.37	80.63	79.98	78.58
+ n-1	83.16	80.91	**81.23**	78.71
+ n-1 + PoS	83.79	81.19	80.47	78.56
+ 2-1	81.51	80.72	80.32	78.43
+ 2-1 + PoS	**84.40**	**81.65**	80.66	78.80

Table 6.5: UAS with simple features.

6.8 summarizes the results from the test of features and shows results from significance tests when comparing the simple extended features to the ones we found to perform best.

6.1.3 Iterative

Figure 6.3 shows results from using the iterative approach. We see pretty much the same picture as we did with Danish and English.

Figure 6.4 shows results with the iterative-with-validation approach. We see that for Danish the match between the development and test set is poor, so the accuracy on the test set drops after the first iteration. On the Spanish data and on the alignments there are small improvements after the initial iteration with extended parsing.

Figure 6.5 shows the results from the iterative-with-retraining on the Danish-Spanish development set. We see the same pattern as we did for Danish-English. The accuracy of the data the extended parsers are trained on increases but the accuracy of the unseen data goes up and down. Again the results at the best iteration are better then from the other approaches, but this could be due to chance.

	Development		Cross-validation	
	Danish	Spanish	Danish	Spanish
baseline	82.02	80.54	79.83	76.49
extended	82.85	80.91	81.02	78.78
+ same + p	84.50	81.00	80.49	78.78
+ same + p + PoS	83.68	80.72	80.55	78.58
+ same + n-1	84.09	80.54	80.46	78.51
+ same + n-1 + PoS	82.02	81.28	80.24	78.74
+ p + n-1	83.37	81.00	80.22	78.64
+ p + n-1 + PoS	82.44	80.63	80.58	78.12
+ same + 2-1	82.95	80.82	80.39	78.66
+ same + 2-1 + PoS	81.30	81.28	80.18	78.84
+ p + 2-1	83.47	81.09	80.33	78.57
+ p + 2-1 + PoS	83.16	81.46	81.10	78.55
+ n-1 + 2-1	84.09	82.48	79.99	78.59
+ n-1 + 2-1 + PoS	84.09	81.19	80.49	78.74
+ same + p + 2-1	82.85	82.30	80.85	78.53
+ same + p + 2-1 + PoS	83.78	80.72	80.71	78.85
+ same + n-1 + 2-1	83.78	81.65	80.05	78.64
+ same + n-1 + 2-1 + PoS	82.85	80.44	80.11	78.58
+ p + n-1 + 2-1	82.75	81.56	80.61	79.01
+ p + n-1 + 2-1 + PoS	83.57	80.91	80.38	78.05
+ same + p + n-1	83.57	80.63	80.48	78.89
+ same + p + n-1 + PoS	82.23	81.37	80.59	78.68
+ same + p + n-1 + 2-1	83.26	81.00	80.29	78.57
+ same + p + n-1 + 2-1 + PoS	83.37	82.21	80.41	78.56

Table 6.6: UAS with simple features.

6.1.4 Reranking

We also also repeat the reranking approach for the Danish-Spanish data. Although the data set is much smaller and we can probably use the entire data set for training we performe the sampling described in section 5.2.2.

Table 6.9 shows the results from the experiment. Again we compare

	Danish	English/Spanish
Danish - English, full	0.34	1.91
Danish - English, 373 sent.	3.44	3.35
Danish - Spanish	1.14	0.55

Table 6.7: Improvement in UAS with different language-pairs.

	Development		Cross-validation	
	Danish	Spanish	Danish	Spanish
simple	82.02	80.54	79.83	76.49
best	**83.16**	**81.09**	**81.23†**	**79.05†**

Table 6.8: Results with simple features and best features for extended parsing.

	Danish	Spanish
Baseline	77.58	66.08
Average (k=50)	77.57	64.53
Oracle (k=50)	85.53	72.85
Reranked	**79.24†**	**68.21†**

Table 6.9: Reranking results for Danish-Spanish.

the improvements with the improvements for Danish-English. Table 6.10 shows this comparison. For Danish, we see the same pattern as with the extended parsing. The improvement in Danish-Spanish is roughly the same as the improvement in Danish-English for the full data set, but much larger for Danish-English with the smaller data set. For the smaller data set though, the improvement for English is smaller than in the other two cases.

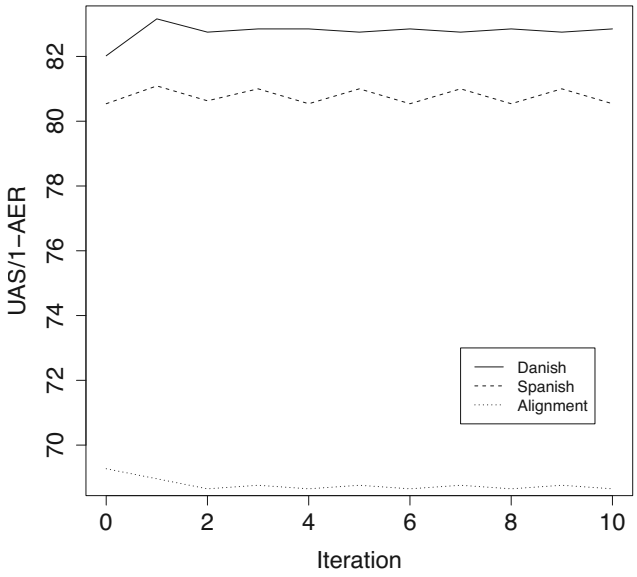

Figure 6.3: Result per iteration with the basic iterative approach.

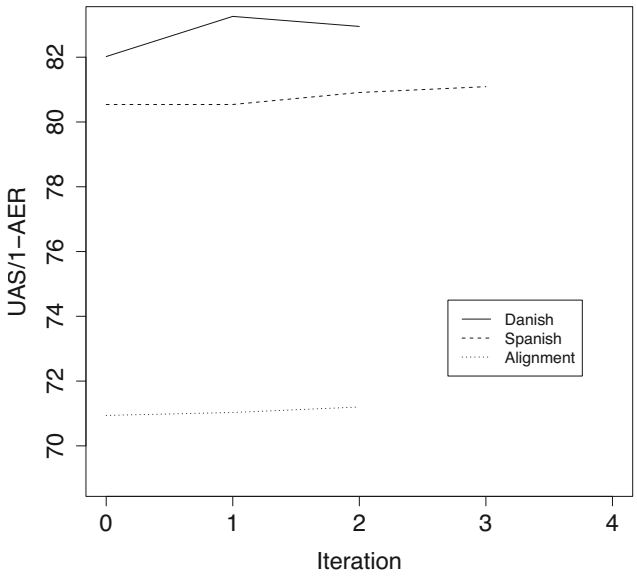

Figure 6.4: Result per iteration with the iterative approach with validation.

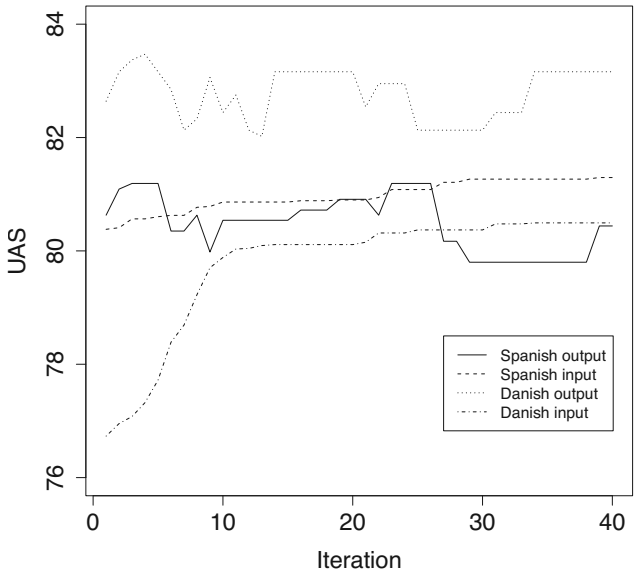

Figure 6.5: Result per iteration with the iterative approach with retraining.

	Danish	English/Spanish
Danish - English, full	1.63	2.67
Danish - English, 373 sent.	7.00	1.69
Danish - Spanish	1.66	2.13

Table 6.10: Improvement in UAS with different language-pairs. Reranking

6.2 Extrinsic Evaluation - SMT

The evaluation metrics traditionally used for parsing and alignment are not necessarily appropriate for measuring the usefullness of an automatically created parallel treebank for SMT. In fact the lack of correlation between AER and MT evaluation metrics such as BLEU are well-studied (Lopez and Resnik, 2006). Zhechev and Way (2008) also reports a lack of correspondence between the intrinsic measures and the quality of the MT-output.

In this section we will discuss how parallel dependency treebanks can be used SMT and discuss if the methods we present are efficient enough for this.

6.2.1 Using Parallel Dependency Treebanks in SMT

Most of the work done in creating parallel treebanks automatically is aimed at improving SMT and are often evaluated with respect to this. Some of these evaluations are done with SMT-systems that directly uses the treebanks. For instance Zhechev and Way (2008) evaluates the treebanks using the Data-Oriented Translation System.

Tinsley, Hearne, and Way (2009) present a more indirect way of using parallel treebank in SMT. Phrase-based SMT (PBSMT) does not directly use any linguistic structures, but are based on n-grams that are obtained using automatic word alignment. Tinsley, Hearne, and Way (2009) show that by expanding the (aligned) phrases obtained by the standard PBSMT-system with phrases extracted from the parallel treebank, a higher quality of translations can be obtained. Phrases are extracted from the treebank simply by letting a phrase-pair consist of the words dominated by the aligned word on one side to the words dominated by the aligned word on the other side. This method can easily be used with dependency grammars, and it has been shown that there is no significant difference in translation quality between using phrase-structure parsers and dependency parsers for this (Tinsley, 2010).

Tinsley, Hearne, and Way (2009) try a number of different ways of combining the phrases from the standard PBSMT-system and the phrases from the treebank and find that simply adding them leads to the best results. The only exception is that it is beneficial to exclude all 1-1 phrases.

We have tried to follow the approach described by Tinsley, Hearne, and Way (2009) with the Moses system (Koehn et al., 2007) and Europarl data (Koehn, 2005), but have not been able to get any positive results. In all experiments the addition of extra phrases lead to lower translation quality (BLEU and NIST) than the baseline system. We have tried this for both Danish-English and Danish-Spanish, with corpora of different sizes[2].

An explanation for this could be that in some sense we do not do sub-tree alignment, but sub-tree informed word alignment. For instance we have no condition that requires the trees to be consistent. Also we found that the inside-outside features used in most work on tree alignment lead to an increase in AER so we have not used these. We have tried filtering the phrases from our system to include only consistent phrase-pairs, but this did not change the outcome.

By manual inspection the phrases extracted look to be of at least the same quality as the ones extracted by Moses, but they consistently lead to lower translation quality. Figure 6.6 shows the 20 first extracted phrase-pairs for Danish-English (with the baseline parser) to illustrate the quality of the extracted phrases.

6.2.2 Efficiency

In this section we will look at the efficiency of the methods we have discussed. We have aimed at using only methods that are fast enough to process large amount of data. Europarl is often used in SMT, and we will use this data for testing the empirical run-time of the tools we use.

Time Complexity

We will focus on the empirical run-time but briefly discuss the time complexity of the tools used.

The baseline parser runs in $O(n^3)$ because it uses second-order parsing. In the data we have used we have filtered out sentences with more than 100 words or more as GIZA++ cannot handle these. With these sentences we have not had any problems with the baseline parser. The same is the

[2]Tinsley (2010) reports that the effect of the extra phrases seems to disappear when the corpus is big enough.

et skoleeksempel	an extreme exercise
i disse værdier	for the said values
alle forudsætningerne	all the conditions
af ruslands aktuelle intentioner	of russia 's current intentions
i georgien	in georgia
særlig relevant	particularly relevant
ruslands aktuelle intentioner	russia 's current intentions
test af ruslands aktuelle intentioner	test of russia 's current intentions
på prøve	to the test
række medlemsstater	several member states
inden invasionen	prior to the invasion
200 observatører	hundred observers
til konfliktområderne	to the scenes of conflict
med rusland	with russia
af georgien	of georgia
i regionen	in the region
den russiske invasion af georgien	the russian invasion of georgia
med den russiske invasion af georgien	of the russian invasion of georgia
eu ' s energisikkerhed	the union 's energy security
for eu ' s energisikkerhed	to the union 's energy security

Figure 6.6: Phrases extracted from automatically created treebank based on Europarl.

case for the aligner which also has run-time $O(n^3)$. The kMST-parser runs in $O(kn^2)$ and we have not had any problems with this either.

The algorithm used in extended parsing is the same as in baseline parsing, i.e. runs in $O(n^3)$. Before applying an extended parser a baseline parser has to be used on the other language, so in practice the run-time will double. We have not tested the iterative approaches, but in these the parsers will also be used a constant number of times - not changing the theoretical run-time. Given that the number of features extracted for the reranker is linear in the number of words in the sentences the run-time of the reranker is linear in the number of words. This means that the run-time of the reranker will be $O(k^2n)$, because there will be k^2 combinations per sentence.

Empirical Run-Time

We have tested the empirical run-time by processing 10,000 Europarl sentences, with less than 100 words. We have only tested on Danish with English as target language because the methods used are the same for all languages. Table 6.11 shows timings for these experiment, and also the projected time it would take to process the entire Europarl Danish-English data set[3]. We see that none of the approaches we present are considerable slower than using the baseline parser. It may seems strange that the reranker is not faster than the parsers, but the reason for this is that for 10,000 examples it has to rerank 25,000,000 hypotheses.

	10,000	1.7 mil.
Baseline parser	24m	68h
Extended parser	30m	85h
Aligner	6m	17h
kMST	62m	176h
Reranker	29m	81h

Table 6.11: Timings for processing Europarl data with different tools.

[3]Experiments were performed on an Intel Xeon Quad-Core 2.26 GHz CPU, but all of the tools use only one CPU.

Chapter 7

Results, Future Work, and Conclusion

7.1 Results and Discussion

In this section we will present results on evaluation data, since all previous experiments have been evaluated on development data. We will discuss the different results from the different approaches and draw more general conclusions about the task of creating parallel treebanks for related languages. We report PTS for approaches where the alignments vary but otherwise we leave out the results for alignment. The results are similar to those we saw for the development data, i.e. sub-tree alignment is better than word alignment, and the iterative approach does not result in any significant improvements.

7.1.1 Bilingually Informed Parsing

Table 7.1 shows results for bilingually informed parsing for Danish-English. We see a significant improvement over the baseline (for Danish only in UAS). Most previous work in bilingually informed parsing has focused on relatively different language pairs, so the question we posed was whether or not bilingually informed parsing would also work for related languages, and it does.

Table 7.2 shows results for Danish-Spanish. Here we see improvements in UAS, but they are not significant. In the analyses of correspondence we

saw that the correspondence between English and Danish is much larger than between Spanish and Danish. We have argued that divergence is necessary for bilingually informed parsing to work, but as we have seen in several experiments more divergence does not necessarily lead to better results. This is apparently confirmed by the experiments on the Danish-Spanish data as we do not see any significant improvements. On the other hand, the reason for the lack of significance may simply be the small evaluation set used (56 sentences).

	Danish			English		
	LAS	UAS	LA	LAS	UAS	LA
Baseline	74.38	87.70	77.54	77.46	83.14	83.82
Extended	**74.72**	**88.30†**	**77.67**	**79.25†**	**85.23†**	**85.11†**

Table 7.1: Evaluation of extended parsing on evaluation data. Danish-English.

	Danish			Spanish		
	LAS	UAS	LA	LAS	UAS	LA
Baseline	**67.70**	80.14	**72.53**	63.99	79.06	68.63
Extended	67.59	**80.35**	71.91	**65.06**	**79.95**	**69.88**

Table 7.2: Evaluation of extended parsing on evaluation data. Danish-Spanish.

In section 4.3.3 we saw that the increase in accuracy from using extended parsing was bigger when the training set was smaller. Figure 7.1 shows results on the evaluation data for the baseline and extended parsers with different training set sizes. We see that the results follow the pattern reported by Smith and Eisner (2009). Training an extended parser on n sentences gives roughly the same results as training a standard parser on $2n$ sentences. The baseline results are of course worse when there is less data, which means that there is more room for improvement. However, this in itself cannot explain the results. The smaller the training set, the larger the risk of some construction being learned incorrectly. When we add the ex-

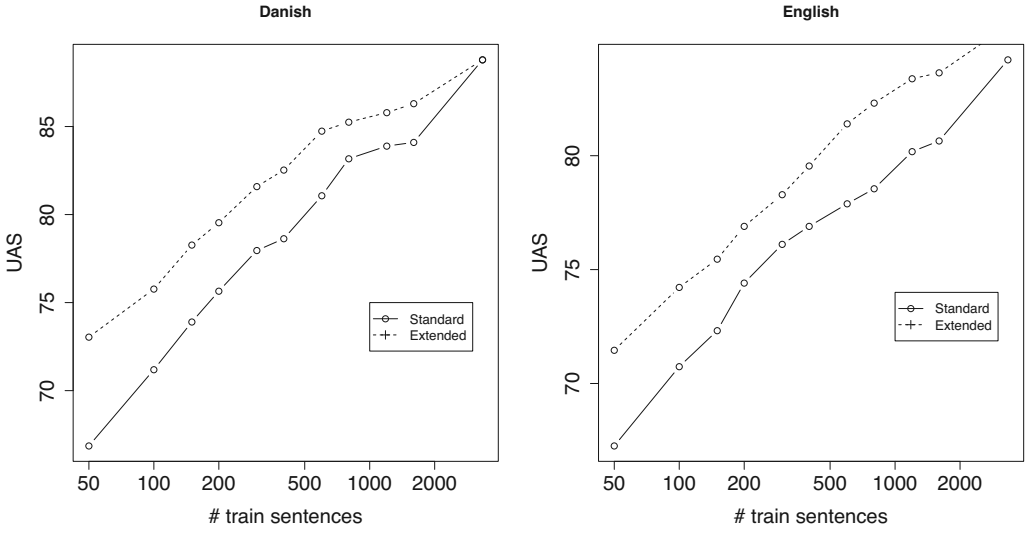

Figure 7.1: UAS of baseline parsers and extended parsers with different amounts of training data.

tra information that is used in extended parsing, there is a chance that this construction was learned correctly in the other language, so in a way the training data is doubled. Of course, large parts of the data correspond so there is little to learn from these. However, we have seen that there is not 100% correspondence and this is enough to allow the extended parsers to learn how to parse constructions correctly where the baseline parser could not.

When we looked at different features for extended parsing we saw changes in parsing accuracy, which did not always seem logical. For instance, combining two apparently good features did not provide good results. It is often difficult to predict which features that will work, but it seems that there may be a general problem related to learning the weights for extended features. It is difficult to say what the problem is. The features used are quite general so overfitting does not seem plausible. It seems more plausible that the features are actually too general, which makes it difficult to learn when the bilingual information is helpful and when it is not.

Overall the conclusion with respect to bilingually informed parsing for related languages is that it works, and that is works better when little training data is available.

7.1.2 Joint Models

Iterative

Table 7.3 shows results on evaluation data using the iterative approaches. For Danish the results are worse than the extended parsing, and for English better, but none of the differences are significant. This is in line with the results on development data.

	Danish			English			
	LAS	UAS	LA	LAS	UAS	LA	PTS$_{\frac{1}{3},\frac{1}{3},\frac{1}{3}}$
Extended	**74.72**	**88.30**	**77.67**	79.25	85.23	85.11	**87.07**
Iterative, basic	74.60	88.15	77.64	79.25	85.28	**85.14**	87.04
Iterative, validation	74.60	88.15	77.64	79.25	85.28	**85.14**	**87.07**
Iterative, retraining	74.66	87.96	77.55	**79.49**	**85.52**	85.13	87.06

Table 7.3: Evaluation of the iterative approach on evaluation data. Danish-English. Significance is compared to extended parsing.

Table 7.4 shows the same results for Danish-Spanish. Here, there are no significant improvements (although the iterative-with-validation approach is significantly better than the baseline on LAS and UAS). The results from

	Danish			Spanish			
	LAS	UAS	LA	LAS	UAS	LA	PTS$_{\frac{1}{3},\frac{1}{3},\frac{1}{3}}$
Extended	67.59	80.35	71.91	**65.06**	79.95	69.88	76.45
Iterative, basic	67.59	80.35	71.91	**65.06**	79.95	69.88	76.51
Iterative, validation	67.59	**80.56**	72.22	64.88	**80.21**	69.25	**76.67**
Iterative, retraining	**68.21**	80.04	**72.33**	64.71	79.77	**69.96**	76.29

Table 7.4: Evaluation of the iterative approach on evaluation data. Danish-Spanish. Significance is compared to extended parsing.

the iterative approaches are not too convincing. We do not see a consistent and significant improvement over the extended parser. For smaller data sets the results were better as shown in section 5.1.5.

Reranking

Table 7.5 shows the results of the reranking approach on Danish-English. We see consistent improvements but only the improvements for English are significant. Table 7.5 shows the results for Danish-Spanish.

	Danish			English		
	LAS	UAS	LA	LAS	UAS	LA
Baseline	68.43	80.06	73.92	65.84	69.97	75.52
Reranked	**68.70**	**80.44**	**74.33**	**68.57†**	**73.06†**	**77.68†**

Table 7.5: Evaluation of the reranking approach on evaluation data. Danish-English.

	Danish			Spanish		
	LAS	UAS	LA	LAS	UAS	LA
Baseline	63.37	75.51	69.44	55.53	67.02	62.83
Reranked	**64.81**	**77.57†**	**69.96**	**56.15**	**67.65**	**63.55**

Table 7.6: Evaluation of the reranking approach on evaluation data. Danish-Spanish.

The overall conclusion with respect to the reranking approach is that we see good results.

7.1.3 Sizes

We have commented on the effect of using different training set sizes above but we will take one more look at this. Table 7.7 shows the relative UAS with the different training sets for all three approaches. We have chosen the iterative-with-validation approach here, because these results are the most stable of the three iterative approaches. The results on the evaluation data confirm the results on the development data. For extended parsing and for the iterative approach the improvements are bigger for smaller data sets. For the reranking approach this is not the case. Table 7.7 also shows that for smaller data sets the improvements are significant in most cases.

	extended		iterative		reranking	
	da	en	da	en	da	en
50	5.75†	2.87†	1.17†	0.89†	-0.12	1.46†
100	4.42†	3.66†	0.19	0.52†	0.78†	1.74†
150	2.69†	3.80†	0.71†	0.61†	0.17†	1.15†
200	3.89†	2.67†	0.26	0.81†	1.16	2.18†
300	2.95†	3.60†	0.30	-0.02	0.58	1.21†
373	3.95†	3.03†	0.37†	0.86†	0.23	1.64†
400	3.65†	2.88†	0.41	1.10†	0.98†	1.91†
600	3.12†	3.44†	0.09	0.66†	0.26	1.92†
800	2.16†	3.72†	0.23	0.49†	0.01	1.59†
1200	2.26†	3.36†	0.37†	0.55†	0.94†	2.35†
1600	2.26†	3.24†	0.00	0.00	0.30	1.91†
3333	0.60†	2.09†	-0.15	0.05	0.38	3.09†

Table 7.7: Relative UAS for all smaller data sets with the three approaches. The results for extended and reranking are compared to the two baseline parsers. For iterative, it is compared to extended parsing.

7.2 Future Directions

In this section we will discuss ways to continue our research on the task of creating parallel treebanks automatically. We will divide the discussion into two parts. First, we will discuss future work that relates directly to the work presented here. Following this, we will discuss other approaches to solving the task.

7.2.1 This Work

Learning

In many cases the results from the different approaches, in particular from the iterative, seemed to vary more than we expected. We saw this with regards to the empirical tests of new features, and also with the iterative approach, which in general worked better for smaller training sets, but did not work at all on the development set with 200 sentences (see table 5.8

in section 5.1.5). We believe that in general the learning for the extended parsing is not stable enough and we would like to look into this in more detail. One possibility would be to first train the standard parser, then fix the parameters learned in this, and finally learn the parameters for the extended features. Presently, all the parameters in the parser are learned simultaneously. The alternative is not guaranteed to give better results, but will make it easier to analyze the results because the baseline parameters will not vary.

Features and Optimization

Although we have tested a number of different features, we still have a lot of work to do with respect to these. For instance, we have not tested combining the extended features with the standard features. The idea in the kind of data-driven approach we adapt is that the parser itself has to learn when to use the extended information. This might possibly be easier if the extended features are combined with the standard features.

Another natural continuation of the work will be to add second-order features.

In the aligner, we made some initial feature selection when implementing it, but we did not do a systematic feature selection on the data used in the experiments. We believe that the accuracy of the aligner will benefit substantially from this.

We have done almost no optimization of hyper-parameters of the tools used[1]. This includes the number of training iterations used in the learning in the parsers and the aligner, but also, for instance, the weighing of precision and recall in the aligner. We weigh them equally, but this may not be optimal. Even for word alignment some work weigh one higher than the other (Lacoste-Julien et al., 2006) and especially in sub-tree alignment it might be better not to optimize directly on AER.

The challenge is that there will most likely be an interference between the different hyper parameters, which makes it necessary to optimize them jointly.

[1]The only exception being the cost parameter for the reranker

Data

Another line of experiments we would like to conduct is to use different data. We would like to use different language-pairs to see how the approaches work with these - also to make our work more comparable with other work.

None of the suggested approaches actually requires a parallel treebank because we always train the models on output from parsers. This makes it possible to do experiments where there are more training data available on one language than the other. This is a quite typical case when one is interested in enriching the resources of a low-resource language using the resources from a high-resource language.

7.2.2 Other Approaches

As with many tasks, there are several possible approaches to the task of creating parallel treebanks. We will briefly discuss some other approaches we have considered.

Post Processing

The challenge in creating parallel treebanks where the trees and the alignments affect each other is the need for higher-order features to account for this interaction. We saw that one general approach to allowing higher-order interaction was reranking. There are other approaches where we first create the parallel tree using some method and then afterwards change the analysis in hope of creating a better analysis. One approach is to define higher-order features and then search for a better tree, using local search. An example of this is the hill-climbing used in the MSTParser for second-order non-projective parsing (see section 2.2.2). Another similar approach is the error-corrective approach described by Hall and Novák (2010).

Transition-Based

We have discussed earlier why a graph-based approach will be difficult for bitext parsing and alignment (section 4.2.2). The other approach to structured prediction we have discussed is reducing this to classification, and for

parsing, we have in particular looked at transition-based parsing. Maybe it is possible to do bitext parsing and alignment with a transition-based system. It will probably require two buffers instead of one, as we have two input sentences. If we just consider a system that is basically two times a normal transition-based system but with only one decision process, it is possible to imagine how the two parses can benefit from each other. If an arc is created in one sentence first, this can inform the parallel sentence and vice versa. If the alignment process is to be included, the system will have to be extended. If we imagine a system where, for instance, an alignment between the two tokens at the head of the two stacks can be added, it is not possible to create crossing alignments. Therefore, something will have to be added to the system to include alignments in the process.

7.3 Conclusion

The task we have addressed is how to create parallel treebanks using data-driven methods. We have presented a number of approaches that all exploit the information that is available in parallel data.

To make use of the bilingual data in parsing, the data needs to be aligned at sub-tree level. We have implemented an aligner that can do this, based on state-of-the-art word aligners.

We have primarily investigated bilingually informed parsing. We have discussed and analyzed why this works even for closely related languages that are annotated with the same kind of syntactic structures. We have also showed that for both Danish-English and Danish-Spanish bilingually informed parsing consistently increases parsing accuracy compared to a baseline parser.

Building on the bilingually informed parsing, we have introduced a number of iterative approaches to bitext parsing and alignment. These approaches are based on the hypothesis that when the input to the bilingually informed parsers gets better, so should the output. We have seen several indications that this hypothesis holds, but we have found it difficult to produce consistent improvements with any of the iterative approaches.

We have also investigated a traditional reranking approach and found that this also results in consistent improvements of parsing accuracy. For

the two first approaches we have shown that the improvement compared to the baseline increases when the size of the training data decreases. This is especially interesting because it allows the possibility of enriching parsing on low-resource languages with parsers from high-resource languages.

References

Ahuja, Ravindra K., Thomas L. Magnanti, and James B. Orlin. 1993. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, NJ.

Bies, Ann, Martha Palmer, Justin Mott, and Colin Warner. 2007. English chinese translation treebank v 1.0. web download. In *In LDC2007T02*.

Buch-Kromann, Matthias. 2006. *Discontinuous Grammar. A model of human parsing and language acquisition*. Copenhagen Business School.

Buch-Kromann, Matthias. 2007. Dependency-based machine translation and parallel parsing without the projectivity and edge-factoring assumptions. a white paper. working paper.

Buch-Kromann, Matthias and Iørn Korzen. 2010. The unified annotation of syntax and discourse in the copenhagen dependency treebanks. In *Proceedings of the Fourth Linguistic Annotation Workshop*, pages 127–131, Uppsala, Sweden, July. Association for Computational Linguistics.

Buch-Kromann, Matthias, Jürgen Wedekind, and Jakob Elming. 2007. The copenhagen danish-english dependency treebank v. 2.0. Department of Computational Linguistics, Copenhagen Business School.

Buchholz, Sabine and Erwin Marsi. 2006. Conll-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164, New York City, June. Association for Computational Linguistics.

Burkett, David, John Blitzer, and Dan Klein. 2010. Joint parsing and alignment with weakly synchronized grammars. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 127–135, Los Angeles, California, June. Association for Computational Linguistics.

Burkett, David and Dan Klein. 2008. Two languages are better than one (for syntactic parsing). In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 877–886, Honolulu, Hawaii, October. Association for Computational Linguistics.

Camerini, P. M., L. Fratta, and F. Maffioli. 1980. The k best spanning arborescences of a network. *Networks*, 10(2):91–109.

Chang, Chih-Chung and Chih-Jen Lin, 2001. *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Charniak, Eugene and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 173–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chen, Wenliang, Jun'ichi Kazama, and Kentaro Torisawa. 2010. Bitext dependency parsing with bilingual subtree constraints. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 21–29, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chen, Wenliang, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009. Improving dependency parsing with subtrees from auto-parsed data. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2*, EMNLP '09, pages 570–579, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chen, Wenliang, Jun'ichi Kazama, Min Zhang, Yoshimasa Tsuruoka, Yujie Zhang, Yiou Wang, Kentaro Torisawa, and Haizhou Li. 2011. Smt helps bitext dependency parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 73–83, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Chu, Y.J. and T.H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.

Čmejrek, Martin, Jan Cuřín, Jiří Havelka, Jan Hajič, and Vladislav Kuboň. 2004. Prague Czech-English Dependency Treebank. Syntactically Annotated Resources for Machine Translation. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, volume V, pages 1597–1600, Lisboa. European Language Resources Association.

Collins, Michael. 2002. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, pages 1–8, Morristown, NJ, USA. Association for Computational Linguistics.

Collins, Michael and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Comput. Linguist.*, 31:25–70, March.

Cortes, Corinna and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning*, 20:273–297. 10.1007/BF00994018.

Crammer, Koby, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *J. Mach. Learn. Res.*, 7:551–585, December.

Crammer, Koby, Mark Dredze, and Alex Kulesza. 2009. Multi-class confidence weighted algorithms. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 496–504, Singapore, August. Association for Computational Linguistics.

Crammer, Koby, Mark Dredze, and Fernando Pereira. 2008. Exact convex confidence-weighted learning. In Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Léon Bottou, editors, *NIPS*, pages 345–352. MIT Press.

Crammer, Koby and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *J. Mach. Learn. Res.*, 3:951–991, March.

Das, Dipanjan and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 600–609, Portland, Oregon, USA, June. Association for Computational Linguistics.

Daume, III, Harold Charles. 2006. *Practical structured learning techniques for natural language processing*. Ph.D. thesis, Los Angeles, CA, USA. AAI3337548.

Dredze, Mark, Koby Crammer, and Fernando Pereira. 2008. Confidence-weighted linear classification. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 264–271, New York, NY, USA. ACM.

Edmonds, Jack R. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240.

Eisner, Jason M. 1996. Three new probabilistic models for dependency parsing: an exploration. In *Proceedings of the 16th conference on Computational linguistics - Volume 1*, COLING '96, pages 340–345, Stroudsburg, PA, USA. Association for Computational Linguistics.

Freund, Yoav and Robert E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Mach. Learn.*, 37(3):277–296.

Georgiadis, Leonidas. 2003. Arborescence optimization problems solvable by edmonds' algorithm. *Theor. Comput. Sci.*, 301:427–437, May.

Giménez, Jesús and Lluís Màrquez. 2004. SVMTool: A general POS tagger generator based on Support Vector Machines. In *Proceedings of the 4th LREC*.

Haghighi, Aria, John Blitzer, John DeNero, and Dan Klein. 2009. Better word alignments with supervised itg models. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 923–931, Suntec, Singapore, August. Association for Computational Linguistics.

Hall, Keith. 2007. K-best spanning tree parsing. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 392–399, Prague, Czech Republic, June. Association for Computational Linguistics.

Hall, Keith, Jiří Havelka, and David A. Smith. 2007. Log-linear models of non-projective trees, k-best MST parsing and tree-ranking. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 962–966, Prague, Czech Republic, June. Association for Computational Linguistics.

Hall, Keith, Ryan McDonald, Jason Katz-Brown, and Michael Ringgaard. 2011. Training dependency parsers by jointly optimizing multiple objectives. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1489–1499, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Hall, Keith and Václav Novák. 2010. Corrective dependency parsing. In Harry Bunt, Paola Merlo, and Joakim Nivre, editors, *Trends in Parsing Technology: Dependency Parsing, Domain Adaptation, and Deep Parsing*, Text, Speech and Language Technology. Springer.

Hearne, Mary and Andy Way. 2006. Disambiguation strategies for data-oriented translation. In *Proceedings of the 11th Conference of the European Association for Machine Translation*, pages 59–68.

Huang, Liang, Wenbin Jiang, and Qun Liu. 2009. Bilingually-constrained (monolingual) shift-reduce parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3*, EMNLP '09, pages 1222–1231, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hwa, Rebecca, Philip Resnik, Amy Weinberg, and Okan Kolak. 2002. Evaluating translational correspondence using annotation projection. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 392–399, Stroudsburg, PA, USA. Association for Computational Linguistics.

Joachims, Thorsten. 2002. Optimizing search engines using clickthrough data. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 133–142.

Joachims, Thorsten. 2006. Training linear svms in linear time. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226, New York, NY, USA. ACM.

Johansson, Richard and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proceedings of NODALIDA 2007*, pages 105–112, Tartu, Estonia, May 25-26.

Koehn, Philipp. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the 10th Machine Translation Summit (MT Summit X)*.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: open source toolkit for statistical machine translation. In *Proceedings of the 45rd Meeting of the Association for Computational Linguistics (ACL'07): Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic.

Koo, Terry and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1–11, Stroudsburg, PA, USA. Association for Computational Linguistics.

Koo, Terry, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 1288–1298, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kromann, Matthias T. and Stine K. Lynge. 2004. Danish Dependency Treebank v. 1.0. Department of Computational Linguistics, Copenhagen Business School.

Kübler, Sandra, Ryan T. McDonald, and Joakim Nivre. 2009. *Dependency Parsing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Lacoste-Julien, Simon, Ben Taskar, Dan Klein, and Michael I. Jordan. 2006. Word alignment via quadratic assignment. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 112–119, Morristown, NJ, USA. Association for Computational Linguistics.

Lopez, Adam and Philip Resnik. 2006. Word-based alignment, phrase-based translation: what's the link? In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas (AMTA'06)*, pages 90–99.

Marcus, Mitchell P., Beatrice Santorini, and Mary A. Marcinkiewicz. 1994. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19.

Martins, André F. T., Dipanjan Das, Noah A. Smith, and Eric P. Xing. 2008. Stacking dependency parsers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 157–166, Stroudsburg, PA, USA. Association for Computational Linguistics.

McDonald, Ryan, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 91–98, Ann Arbor, Michigan, June. Association for Computational Linguistics.

McDonald, Ryan, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 216–220, New York City, June. Association for Computational Linguistics.

McDonald, Ryan and Joakim Nivre. 2011. Analyzing and integrating dependency parsers. *Comput. Linguist.*, 37:197–230.

McDonald, Ryan, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

McDonald, Ryan, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 62–72,

Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

McDonald, Ryan T. 2009. Generalized linear classifiers in nlp. Slides for lecture at the *Swedish Graduate School of Language Technology*, `http://www.ryanmcd.com/courses/gslt2009/gslt2009.pdf`.

McDonald, Ryan T. and Fernando C. N. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *EACL*. The Association for Computer Linguistics.

Melamed, I. Dan. 2003. Multitext grammars and synchronous parsers. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 79–86, Morristown, NJ, USA. Association for Computational Linguistics.

Nilsson, Jens and Joakim Nivre. 2008. MaltEval: An evaluation and visualization tool for dependency parsing. In *Proceedings of the Sixth International Language Resources and Evaluation*, Marrakech, Morocco, May. LREC.

Nivre, Joakim. 2006. *Inductive Dependency Parsing (Text, Speech and Language Technology)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Nivre, Joakim. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

Nivre, Joakim, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932.

Nivre, Joakim, Johan Hall, and Jens Nilsson. 2006. MaltParser: A data-driven parser-generator for dependency parsing. In *Proceedings of the fifth international conference on Language Resources and Evaluation (LREC2006)*, pages 2216–2219, May.

Nivre, Joakim, Johan Hall, Jens Nilsson, Gülşen Eryiğit, and Svetoslav Marinov. 2006. Labeled pseudo-projective dependency parsing with

support vector machines. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 221–225, New York City, June. Association for Computational Linguistics.

Nivre, Joakim and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL-08: HLT*, pages 950–958, Columbus, Ohio, June. Association for Computational Linguistics.

Och, Franz Josef and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Comput. Linguist.*, 29(1):19–51.

Petrov, Slav, Dipanjan Das, and Ryan McDonald. 2011. A universal part-of-speech tagset. *CoRR*, abs/1104.2086.

Sagae, Kenji and Jun'ichi Tsujii. 2008. Shift-reduce dependency DAG parsing. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 753–760, Manchester, UK, August. Coling 2008 Organizing Committee.

Samuelsson, Yvonne and Martin Volk. 2007. Automatic phrase alignment. using statistical n-gram alignment for syntactic phrase alignment. In *6th Workshop on Treebanks and Linguistic Theories*.

Shieber, Stuart M. and Yves Schabes. 1990. Synchronous tree-adjoining grammars. In *Proceedings of the 13th conference on Computational linguistics - Volume 3*, COLING '90, pages 253–258, Stroudsburg, PA, USA. Association for Computational Linguistics.

Smith, David A. and Jason Eisner. 2006. Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proceedings of the HLT-NAACL Workshop on Statistical Machine Translation*, pages 23–30, New York, June.

Smith, David A. and Jason Eisner. 2008. Dependency parsing by belief propagation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 145–156, Honolulu, October.

Smith, David A. and Jason Eisner. 2009. Parser adaptation and projection with quasi-synchronous grammar features. In *Proceedings of the Confer-*

ence on Empirical Methods in Natural Language Processing (EMNLP), pages 822–831, Singapore, August.

Smith, David A. and Noah A. Smith. 2004. Bilingual parsing with factored estimation: Using english to parse korean. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 49–54.

Søgaard, Anders. 2011. Data point selection for cross-language adaptation of dependency parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 682–686, Portland, Oregon, USA, June. Association for Computational Linguistics.

Tarjan, Robert E. 1977. Finding optimum branchings. *Networks*, 7:25–35.

Taskar, Ben, Simon Lacoste-Julien, and Dan Klein. 2005. A discriminative matching approach to word alignment. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 73–80, Morristown, NJ, USA. Association for Computational Linguistics.

Taskar, Benjamin, Carlos Guestrin, and Daphne Koller. 2003. Max-margin markov networks. In Sebastian Thrun, Lawrence K. Saul, and Bernhard Schölkopf, editors, *NIPS*. MIT Press.

Tiedemann, Jörg. 2010. Lingua-align: An experimental toolbox for automatic tree-to-tree alignment. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC'2010)*, Valetta, Malta.

Tiedemann, Jörg and Gideon Kotzé. 2009a. Building a large machine-aligned parallel treebank. In Marco Passarotti, Adam Przepirkowski, Savina Raynaud, and Frank Van Eynde, editors, *Proceedings of the 8th International Workshop on Treebanks and Linguistic Theories (TLT'08)*, pages 197–208. EDUCatt, Milano/Italy.

Tiedemann, Jörg and Gideon Kotzé. 2009b. A discriminative approach to tree alignment. In Iustina Ilisei, Viktor Pekar, and Silvia Bernardini, editors, *Proceedings of the Workshop on Natural Language Processing Methods*

and Corpora in Translation, Lexicography, and Language Learning (in connection with RANLP'09), pages 33 – 39, Borovets, Bulgaria, September. Association for Computational Linguistics.

Tinsley, John. 2010. *Resourcing machine translation with parallel treebanks*. Ph.D. thesis, Dublin City University. National Centre for Language Technology (NCLT), Mar.

Tinsley, John, Mary Hearne, and Andy Way. 2009. Parallel treebanks in phrase-based statistical machine translation. In *Proceedings of the International Conference on Intelligent Text Processing and Computational Linguistics (CICLing)*, pages 318–331, Mexico City.

Vadas, David and James Curran. 2007. Adding noun phrase structure to the penn treebank. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 240–247, Prague, Czech Republic, June. Association for Computational Linguistics.

Venkatapathy, Sriram and Aravind K. Joshi. 2007. Discriminative word alignment by learning the alignment structure and syntactic divergence between a language pair. In *Proceedings of the NAACL-HLT 2007/AMTA Workshop on Syntax and Structure in Statistical Translation*, SSST '07, pages 49–56, Stroudsburg, PA, USA. Association for Computational Linguistics.

Volk, Martin, Anne Ghring, Torsten Marek, and Yvonne Samuelsson. 2010. SMULTRON (version 3.0) The Stockholm MULtilingual parallel TReebank. `http://www.cl.uzh.ch/research/paralleltreebanks_en.html`. An English-French-German-Spanish-Swedish parallel treebank with sub-sentential alignments.

Wu, Dekai. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

Zeman, Daniel and Philip Resnik. 2008. Cross-Language Parser Adaptation between Related Languages. In *IJCNLP-08 Workshop on NLP for Less Privileged Languages*, pages 35–42, Hyderabad, India, January.

Zhao, Hai, Yan Song, Chunyu Kit, and Guodong Zhou. 2009. Cross language dependency parsing using a bilingual lexicon. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, pages 55–63, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zhechev, Ventsislav and Andy Way. 2008. Automatic generation of parallel treebanks. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 1105–1112, Stroudsburg, PA, USA. Association for Computational Linguistics.

TITLER I PH.D.SERIEN:

2004

1. Martin Grieger
Internet-based Electronic Marketplaces and Supply Chain Management

2. Thomas Basbøll
LIKENESS
A Philosophical Investigation

3. Morten Knudsen
Beslutningens vaklen
En systemteoretisk analyse of moderniseringen af et amtskommunalt sundhedsvæsen 1980-2000

4. Lars Bo Jeppesen
Organizing Consumer Innovation
A product development strategy that is based on online communities and allows some firms to benefit from a distributed process of innovation by consumers

5. Barbara Dragsted
SEGMENTATION IN TRANSLATION AND TRANSLATION MEMORY SYSTEMS
An empirical investigation of cognitive segmentation and effects of integrating a TM system into the translation process

6. Jeanet Hardis
Sociale partnerskaber
Et socialkonstruktivistisk casestudie af partnerskabsaktørers virkelighedsopfattelse mellem identitet og legitimitet

7. Henriette Hallberg Thygesen
System Dynamics in Action

8. Carsten Mejer Plath
Strategisk Økonomistyring

9. Annemette Kjærgaard
Knowledge Management as Internal Corporate Venturing
– a Field Study of the Rise and Fall of a Bottom-Up Process

10. Knut Arne Hovdal
De profesjonelle i endring
Norsk ph.d., ej til salg gennem Samfundslitteratur

11. Søren Jeppesen
Environmental Practices and Greening Strategies in Small Manufacturing Enterprises in South Africa
– A Critical Realist Approach

12. Lars Frode Frederiksen
Industriel forskningsledelse
– på sporet af mønstre og samarbejde i danske forskningsintensive virksomheder

13. Martin Jes Iversen
The Governance of GN Great Nordic
– in an age of strategic and structural transitions 1939-1988

14. Lars Pynt Andersen
The Rhetorical Strategies of Danish TV Advertising
A study of the first fifteen years with special emphasis on genre and irony

15. Jakob Rasmussen
Business Perspectives on E-learning

16. Sof Thrane
The Social and Economic Dynamics of Networks
– a Weberian Analysis of Three Formalised Horizontal Networks

17. Lene Nielsen
Engaging Personas and Narrative Scenarios – a study on how a user-centered approach influenced the perception of the design process in the e-business group at AstraZeneca

18. S.J Valstad
Organisationsidentitet
Norsk ph.d., ej til salg gennem Samfundslitteratur

19. Thomas Lyse Hansen
Six Essays on Pricing and Weather risk in Energy Markets

20. Sabine Madsen
Emerging Methods – An Interpretive Study of ISD Methods in Practice

21. Evis Sinani
The Impact of Foreign Direct Investment on Efficiency, Productivity Growth and Trade: An Empirical Investigation

22. Bent Meier Sørensen
Making Events Work Or, How to Multiply Your Crisis

23. Pernille Schnoor
Brand Ethos
Om troværdige brand- og virksomhedsidentiteter i et retorisk og diskursteoretisk perspektiv

24. Sidsel Fabech
Von welchem Österreich ist hier die Rede?
Diskursive forhandlinger og magtkampe mellem rivaliserende nationale identitetskonstruktioner i østrigske pressediskurser

25. Klavs Odgaard Christensen
Sprogpolitik og identitetsdannelse i flersprogede forbundsstater
Et komparativt studie af Schweiz og Canada

26. Dana B. Minbaeva
Human Resource Practices and Knowledge Transfer in Multinational Corporations

27. Holger Højlund
Markedets politiske fornuft
Et studie af velfærdens organisering i perioden 1990-2003

28. Christine Mølgaard Frandsen
A.s erfaring
Om mellemværendets praktik i en transformation af mennesket og subjektiviteten

29. Sine Nørholm Just
The Constitution of Meaning – A Meaningful Constitution?
Legitimacy, identity, and public opinion in the debate on the future of Europe

2005

1. Claus J. Varnes
Managing product innovation through rules – The role of formal and structured methods in product development

2. Helle Hedegaard Hein
Mellem konflikt og konsensus – Dialogudvikling på hospitalsklinikker

3. Axel Rosenø
Customer Value Driven Product Innovation – A Study of Market Learning in New Product Development

4. Søren Buhl Pedersen
Making space
An outline of place branding

5. Camilla Funck Ellehave
Differences that Matter
An analysis of practices of gender and organizing in contemporary workplaces

6. Rigmor Madeleine Lond
Styring af kommunale forvaltninger

7. Mette Aagaard Andreassen
Supply Chain versus Supply Chain Benchmarking as a Means to Managing Supply Chains

8. Caroline Aggestam-Pontoppidan
From an idea to a standard
The UN and the global governance of accountants' competence

9. Norsk ph.d.

10. Vivienne Heng Ker-ni
An Experimental Field Study on the

Effectiveness of Grocer Media Advertising
Measuring Ad Recall and Recognition, Purchase Intentions and Short-Term Sales

11. Allan Mortensen
Essays on the Pricing of Corporate Bonds and Credit Derivatives

12. Remo Stefano Chiari
Figure che fanno conoscere
Itinerario sull'idea del valore cognitivo e espressivo della metafora e di altri tropi da Aristotele e da Vico fino al cognitivismo contemporaneo

13. Anders McIlquham-Schmidt
Strategic Planning and Corporate Performance
An integrative research review and a meta-analysis of the strategic planning and corporate performance literature from 1956 to 2003

14. Jens Geersbro
The TDF – PMI Case
Making Sense of the Dynamics of Business Relationships and Networks

15 Mette Andersen
Corporate Social Responsibility in Global Supply Chains
Understanding the uniqueness of firm behaviour

16. Eva Boxenbaum
Institutional Genesis: Micro – Dynamic Foundations of Institutional Change

17. Peter Lund-Thomsen
Capacity Development, Environmental Justice NGOs, and Governance: The Case of South Africa

18. Signe Jarlov
Konstruktioner af offentlig ledelse

19. Lars Stæhr Jensen
Vocabulary Knowledge and Listening Comprehension in English as a Foreign Language
An empirical study employing data elicited from Danish EFL learners

20. Christian Nielsen
Essays on Business Reporting
Production and consumption of strategic information in the market for information

21. Marianne Thejls Fischer
Egos and Ethics of Management Consultants

22. Annie Bekke Kjær
Performance management i Proces-innovation
– belyst i et social-konstruktivistisk perspektiv

23. Suzanne Dee Pedersen
GENTAGELSENS METAMORFOSE
Om organisering af den kreative gøren i den kunstneriske arbejdspraksis

24. Benedikte Dorte Rosenbrink
Revenue Management
Økonomiske, konkurrencemæssige & organisatoriske konsekvenser

25. Thomas Riise Johansen
Written Accounts and Verbal Accounts
The Danish Case of Accounting and Accountability to Employees

26. Ann Fogelgren-Pedersen
The Mobile Internet: Pioneering Users' Adoption Decisions

27. Birgitte Rasmussen
Ledelse i fællesskab – de tillidsvalgtes fornyende rolle

28. Gitte Thit Nielsen
Remerger
– skabende ledelseskræfter i fusion og opkøb

29. Carmine Gioia
A MICROECONOMETRIC ANALYSIS OF MERGERS AND ACQUISITIONS

30. Ole Hinz
Den effektive forandringsleder: pilot, pædagog eller politiker?
Et studie i arbejdslederes meningstilskrivninger i forbindelse med vellykket gennemførelse af ledelsesinitierede forandringsprojekter

31. Kjell-Åge Gotvassli
Et praksisbasert perspektiv på dynamiske
læringsnettverk i toppidretten
Norsk ph.d., ej til salg gennem Samfundslitteratur

32. Henriette Langstrup Nielsen
Linking Healthcare
An inquiry into the changing performances of web-based technology for asthma monitoring

33. Karin Tweddell Levinsen
Virtuel Uddannelsespraksis
Master i IKT og Læring – et casestudie i hvordan proaktiv proceshåndtering kan forbedre praksis i virtuelle læringsmiljøer

34. Anika Liversage
Finding a Path
Labour Market Life Stories of Immigrant Professionals

35. Kasper Elmquist Jørgensen
Studier i samspillet mellem stat og erhvervsliv i Danmark under
1. verdenskrig

36. Finn Janning
A DIFFERENT STORY
Seduction, Conquest and Discovery

37. Patricia Ann Plackett
Strategic Management of the Radical Innovation Process
Leveraging Social Capital for Market Uncertainty Management

2006

1. Christian Vintergaard
Early Phases of Corporate Venturing

2. Niels Rom-Poulsen
Essays in Computational Finance

3. Tina Brandt Husman
Organisational Capabilities, Competitive Advantage & Project-Based Organisations
The Case of Advertising and Creative Good Production

4. Mette Rosenkrands Johansen
Practice at the top
– how top managers mobilise and use non-financial performance measures

5. Eva Parum
Corporate governance som strategisk kommunikations- og ledelsesværktøj

6. Susan Aagaard Petersen
Culture's Influence on Performance Management: The Case of a Danish Company in China

7. Thomas Nicolai Pedersen
The Discursive Constitution of Organizational Governance – Between unity and differentiation
The Case of the governance of environmental risks by World Bank environmental staff

8. Cynthia Selin
Volatile Visions: Transactons in Anticipatory Knowledge

9. Jesper Banghøj
Financial Accounting Information and Compensation in Danish Companies

10. Mikkel Lucas Overby
Strategic Alliances in Emerging High-Tech Markets: What's the Difference and does it Matter?

11. Tine Aage
External Information Acquisition of Industrial Districts and the Impact of Different Knowledge Creation Dimensions

A case study of the Fashion and Design Branch of the Industrial District of Montebelluna, NE Italy

12. Mikkel Flyverbom
Making the Global Information Society Governable
On the Governmentality of Multi-Stakeholder Networks

13. Anette Grønning
Personen bag
Tilstedevær i e-mail som interaktionsform mellem kunde og medarbejder i dansk forsikringskontekst

14. Jørn Helder
One Company – One Language?
The NN-case

15. Lars Bjerregaard Mikkelsen
Differing perceptions of customer value
Development and application of a tool for mapping perceptions of customer value at both ends of customer-supplier dyads in industrial markets

16. Lise Granerud
Exploring Learning
Technological learning within small manufacturers in South Africa

17. Esben Rahbek Pedersen
Between Hopes and Realities:
Reflections on the Promises and Practices of Corporate Social Responsibility (CSR)

18. Ramona Samson
The Cultural Integration Model and European Transformation.
The Case of Romania

2007

1. Jakob Vestergaard
Discipline in The Global Economy
Panopticism and the Post-Washington Consensus

2. Heidi Lund Hansen
Spaces for learning and working
A qualitative study of change of work, management, vehicles of power and social practices in open offices

3. Sudhanshu Rai
Exploring the internal dynamics of software development teams during user analysis
A tension enabled Institutionalization Model; "Where process becomes the objective"

4. Norsk ph.d.
Ej til salg gennem Samfundslitteratur

5. Serden Ozcan
EXPLORING HETEROGENEITY IN ORGANIZATIONAL ACTIONS AND OUTCOMES
A Behavioural Perspective

6. Kim Sundtoft Hald
Inter-organizational Performance Measurement and Management in Action
– An Ethnography on the Construction of Management, Identity and Relationships

7. Tobias Lindeberg
Evaluative Technologies
Quality and the Multiplicity of Performance

8. Merete Wedell-Wedellsborg
Den globale soldat
Identitetsdannelse og identitetsledelse i multinationale militære organisationer

9. Lars Frederiksen
Open Innovation Business Models
Innovation in firm-hosted online user communities and inter-firm project ventures in the music industry
– A collection of essays

10. Jonas Gabrielsen
Retorisk toposlære – fra statisk 'sted' til persuasiv aktivitet

11. Christian Moldt-Jørgensen
Fra meningsløs til meningsfuld evaluering.
Anvendelsen af studentertilfredshedsmålinger på de korte og mellemlange videregående uddannelser set fra et psykodynamisk systemperspektiv

12. Ping Gao
Extending the application of actor-network theory
Cases of innovation in the telecommunications industry

13. Peter Mejlby
Frihed og fængsel, en del af den samme drøm?
Et phronetisk baseret casestudie af frigørelsens og kontrollens sameksistens i værdibaseret ledelse!

14. Kristina Birch
Statistical Modelling in Marketing

15. Signe Poulsen
Sense and sensibility:
The language of emotional appeals in insurance marketing

16. Anders Bjerre Trolle
Essays on derivatives pricing and dynamic asset allocation

17. Peter Feldhütter
Empirical Studies of Bond and Credit Markets

18. Jens Henrik Eggert Christensen
Default and Recovery Risk Modeling and Estimation

19. Maria Theresa Larsen
Academic Enterprise: A New Mission for Universities or a Contradiction in Terms?
Four papers on the long-term implications of increasing industry involvement and commercialization in academia

20. Morten Wellendorf
Postimplementering af teknologi i den offentlige forvaltning
Analyser af en organisations kontinuerlige arbejde med informationsteknologi

21. Ekaterina Mhaanna
Concept Relations for Terminological Process Analysis

22. Stefan Ring Thorbjørnsen
Forsvaret i forandring
Et studie i officerers kapabiliteter under påvirkning af omverdenens forandringspres mod øget styring og læring

23. Christa Breum Amhøj
Det selvskabte medlemskab om managementstaten, dens styringsteknologier og indbyggere

24. Karoline Bromose
Between Technological Turbulence and Operational Stability
– An empirical case study of corporate venturing in TDC

25. Susanne Justesen
Navigating the Paradoxes of Diversity in Innovation Practice
– A Longitudinal study of six very different innovation processes – in practice

26. Luise Noring Henler
Conceptualising successful supply chain partnerships
– Viewing supply chain partnerships from an organisational culture perspective

27. Mark Mau
Kampen om telefonen
Det danske telefonvæsen under den tyske besættelse 1940-45

28. Jakob Halskov
The semiautomatic expansion of existing terminological ontologies using knowledge patterns discovered

on the WWW – an implementation and evaluation

29. Gergana Koleva
European Policy Instruments Beyond Networks and Structure: The Innovative Medicines Initiative

30. Christian Geisler Asmussen
Global Strategy and International Diversity: A Double-Edged Sword?

31. Christina Holm-Petersen
Stolthed og fordom
Kultur- og identitetsarbejde ved skabelsen af en ny sengeafdeling gennem fusion

32. Hans Peter Olsen
Hybrid Governance of Standardized States
Causes and Contours of the Global Regulation of Government Auditing

33. Lars Bøge Sørensen
Risk Management in the Supply Chain

34. Peter Aagaard
Det unikkes dynamikker
De institutionelle mulighedsbetingelser bag den individuelle udforskning i professionelt og frivilligt arbejde

35. Yun Mi Antorini
Brand Community Innovation
An Intrinsic Case Study of the Adult Fans of LEGO Community

36. Joachim Lynggaard Boll
Labor Related Corporate Social Performance in Denmark
Organizational and Institutional Perspectives

2008

1. Frederik Christian Vinten
Essays on Private Equity

2. Jesper Clement
Visual Influence of Packaging Design on In-Store Buying Decisions

3. Marius Brostrøm Kousgaard
Tid til kvalitetsmåling?
– Studier af indrulleringsprocesser i forbindelse med introduktionen af kliniske kvalitetsdatabaser i speciallægepraksissektoren

4. Irene Skovgaard Smith
Management Consulting in Action
Value creation and ambiguity in client-consultant relations

5. Anders Rom
Management accounting and integrated information systems
How to exploit the potential for management accounting of information technology

6. Marina Candi
Aesthetic Design as an Element of Service Innovation in New Technology-based Firms

7. Morten Schnack
Teknologi og tværfaglighed
– en analyse af diskussionen omkring indførelse af EPJ på en hospitalsafdeling

8. Helene Balslev Clausen
Juntos pero no revueltos – un estudio sobre emigrantes norteamericanos en un pueblo mexicano

9. Lise Justesen
Kunsten at skrive revisionsrapporter. En beretning om forvaltningsrevisionens beretninger

10. Michael E. Hansen
The politics of corporate responsibility: CSR and the governance of child labor and core labor rights in the 1990s

11. Anne Roepstorff
Holdning for handling – en etnologisk undersøgelse af Virksomheders Sociale Ansvar/CSR

12. Claus Bajlum
Essays on Credit Risk and Credit Derivatives

13. Anders Bojesen
The Performative Power of Competence – an Inquiry into Subjectivity and Social Technologies at Work

14. Satu Reijonen
Green and Fragile
A Study on Markets and the Natural Environment

15. Ilduara Busta
Corporate Governance in Banking
A European Study

16. Kristian Anders Hvass
A Boolean Analysis Predicting Industry Change: Innovation, Imitation & Business Models
The Winning Hybrid: A case study of isomorphism in the airline industry

17. Trine Paludan
De uvidende og de udviklingsparate
Identitet som mulighed og restriktion blandt fabriksarbejdere på det aftayloriserede fabriksgulv

18. Kristian Jakobsen
Foreign market entry in transition economies: Entry timing and mode choice

19. Jakob Elming
Syntactic reordering in statistical machine translation

20. Lars Brømsøe Termansen
Regional Computable General Equilibrium Models for Denmark
Three papers laying the foundation for regional CGE models with agglomeration characteristics

21. Mia Reinholt
The Motivational Foundations of Knowledge Sharing

22. Frederikke Krogh-Meibom
The Co-Evolution of Institutions and Technology
– A Neo-Institutional Understanding of Change Processes within the Business Press – the Case Study of Financial Times

23. Peter D. Ørberg Jensen
OFFSHORING OF ADVANCED AND HIGH-VALUE TECHNICAL SERVICES: ANTECEDENTS, PROCESS DYNAMICS AND FIRMLEVEL IMPACTS

24. Pham Thi Song Hanh
Functional Upgrading, Relational Capability and Export Performance of Vietnamese Wood Furniture Producers

25. Mads Vangkilde
Why wait?
An Exploration of first-mover advantages among Danish e-grocers through a resource perspective

26. Hubert Buch-Hansen
Rethinking the History of European Level Merger Control
A Critical Political Economy Perspective

2009

1. Vivian Lindhardsen
From Independent Ratings to Communal Ratings: A Study of CWA Raters' Decision-Making Behaviours

2. Guðrið Weihe
Public-Private Partnerships: Meaning and Practice

3. Chris Nøkkentved
Enabling Supply Networks with Collaborative Information Infrastructures
An Empirical Investigation of Business Model Innovation in Supplier Relationship Management

4. Sara Louise Muhr
Wound, Interrupted – On the Vulnerability of Diversity Management

5. Christine Sestoft
Forbrugeradfærd i et Stats- og Livsformsteoretisk perspektiv

6. Michael Pedersen
Tune in, Breakdown, and Reboot: On the production of the stress-fit self-managing employee

7. Salla Lutz
Position and Reposition in Networks – Exemplified by the Transformation of the Danish Pine Furniture Manufacturers

8. Jens Forssbæck
Essays on market discipline in commercial and central banking

9. Tine Murphy
Sense from Silence – A Basis for Organised Action
How do Sensemaking Processes with Minimal Sharing Relate to the Reproduction of Organised Action?

10. Sara Malou Strandvad
Inspirations for a new sociology of art: A sociomaterial study of development processes in the Danish film industry

11. Nicolaas Mouton
On the evolution of social scientific metaphors:
A cognitive-historical enquiry into the divergent trajectories of the idea that collective entities – states and societies, cities and corporations – are biological organisms.

12. Lars Andreas Knutsen
Mobile Data Services:
Shaping of user engagements

13. Nikolaos Theodoros Korfiatis
Information Exchange and Behavior
A Multi-method Inquiry on Online Communities

14. Jens Albæk
Forestillinger om kvalitet og tværfaglighed på sygehuse
– skabelse af forestillinger i læge- og plejegrupperne angående relevans af nye idéer om kvalitetsudvikling gennem tolkningsprocesser

15. Maja Lotz
The Business of Co-Creation – and the Co-Creation of Business

16. Gitte P. Jakobsen
Narrative Construction of Leader Identity in a Leader Development Program Context

17. Dorte Hermansen
"Living the brand" som en brandorienteret dialogisk praxis:
Om udvikling af medarbejdernes brandorienterede dømmekraft

18. Aseem Kinra
Supply Chain (logistics) Environmental Complexity

19. Michael Nørager
How to manage SMEs through the transformation from non innovative to innovative?

20. Kristin Wallevik
Corporate Governance in Family Firms
The Norwegian Maritime Sector

21. Bo Hansen Hansen
Beyond the Process
Enriching Software Process Improvement with Knowledge Management

22. Annemette Skot-Hansen
Franske adjektivisk afledte adverbier, der tager præpositionssyntagmer indledt med præpositionen à som argumenter
En valensgrammatisk undersøgelse

23. Line Gry Knudsen
Collaborative R&D Capabilities
In Search of Micro-Foundations

24. Christian Scheuer
Employers meet employees
Essays on sorting and globalization

25. Rasmus Johnsen
The Great Health of Melancholy
A Study of the Pathologies of Performativity

26. Ha Thi Van Pham
Internationalization, Competitiveness Enhancement and Export Performance of Emerging Market Firms:
Evidence from Vietnam

27. Henriette Balieu
Kontrolbegrebets betydning for kausativalternationen i spansk
En kognitiv-typologisk analyse

2010

1. Yen Tran
Organizing Innovationin Turbulent Fashion Market
Four papers on how fashion firms create and appropriate innovation value

2. Anders Raastrup Kristensen
Metaphysical Labour
Flexibility, Performance and Commitment in Work-Life Management

3. Margrét Sigrún Sigurdardottir
Dependently independent
Co-existence of institutional logics in the recorded music industry

4. Ásta Dis Óladóttir
Internationalization from a small domestic base:
An empirical analysis of Economics and Management

5. Christine Secher
E-deltagelse i praksis – politikernes og forvaltningens medkonstruktion og konsekvenserne heraf

6. Marianne Stang Våland
What we talk about when we talk about space:
End User Participation between Processes of Organizational and Architectural Design

7. Rex Degnegaard
Strategic Change Management
Change Management Challenges in the Danish Police Reform

8. Ulrik Schultz Brix
Værdi i rekruttering – den sikre beslutning
En pragmatisk analyse af perception og synliggørelse af værdi i rekrutterings- og udvælgelsesarbejdet

9. Jan Ole Similä
Kontraktsledelse
Relasjonen mellom virksomhetsledelse og kontraktshåndtering, belyst via fire norske virksomheter

10. Susanne Boch Waldorff
Emerging Organizations: In between local translation, institutional logics and discourse

11. Brian Kane
Performance Talk
Next Generation Management of Organizational Performance

12. Lars Ohnemus
Brand Thrust: Strategic Branding and Shareholder Value
An Empirical Reconciliation of two Critical Concepts

13. Jesper Schlamovitz
Håndtering af usikkerhed i film- og byggeprojekter

14. Tommy Moesby-Jensen
Det faktiske livs forbindtlighed
Førsokratisk informeret, ny-aristotelisk ἦθος-tænkning hos Martin Heidegger

15. Christian Fich
Two Nations Divided by Common Values
French National Habitus and the Rejection of American Power

16. Peter Beyer
Processer, sammenhængskraft og fleksibilitet
Et empirisk casestudie af omstillingsforløb i fire virksomheder

17. Adam Buchhorn
Markets of Good Intentions
Constructing and Organizing Biogas Markets Amid Fragility and Controversy

18. Cecilie K. Moesby-Jensen
Social læring og fælles praksis
Et mixed method studie, der belyser læringskonsekvenser af et lederkursus for et praksisfællesskab af offentlige mellemledere

19. Heidi Boye
Fødevarer og sundhed i senmodernismen
– En indsigt i hyggefænomenet og de relaterede fødevarepraksisser

20. Kristine Munkgård Pedersen
Flygtige forbindelser og midlertidige mobiliseringer
Om kulturel produktion på Roskilde Festival

21. Oliver Jacob Weber
Causes of Intercompany Harmony in Business Markets – An Empirical Investigation from a Dyad Perspective

22. Susanne Ekman
Authority and Autonomy
Paradoxes of Modern Knowledge Work

23. Anette Frey Larsen
Kvalitetsledelse på danske hospitaler
– Ledelsernes indflydelse på introduktion og vedligeholdelse af kvalitetsstrategier i det danske sundhedsvæsen

24. Toyoko Sato
Performativity and Discourse: Japanese Advertisements on the Aesthetic Education of Desire

25. Kenneth Brinch Jensen
Identifying the Last Planner System
Lean management in the construction industry

26. Javier Busquets
Orchestrating Network Behavior for Innovation

27. Luke Patey
The Power of Resistance: India's National Oil Company and International Activism in Sudan

28. Mette Vedel
Value Creation in Triadic Business Relationships. Interaction, Interconnection and Position

29. Kristian Tørning
Knowledge Management Systems in Practice – A Work Place Study

30. Qingxin Shi
An Empirical Study of Thinking Aloud Usability Testing from a Cultural Perspective

31. Tanja Juul Christiansen
Corporate blogging: Medarbejderes kommunikative handlekraft

32. Malgorzata Ciesielska
Hybrid Organisations.
A study of the Open Source – business setting

33. Jens Dick-Nielsen
Three Essays on Corporate Bond Market Liquidity

34. Sabrina Speiermann
Modstandens Politik
Kampagnestyring i Velfærdsstaten.
En diskussion af trafikkampagners styringspotentiale

35. Julie Uldam
Fickle Commitment. Fostering political engagement in 'the flighty world of online activism'

36. Annegrete Juul Nielsen
Traveling technologies and transformations in health care

37. Athur Mühlen-Schulte
Organising Development
Power and Organisational Reform in the United Nations Development Programme

38. Louise Rygaard Jonas
Branding på butiksgulvet
Et case-studie af kultur- og identitetsarbejdet i Kvickly

2011

1. Stefan Fraenkel
Key Success Factors for Sales Force Readiness during New Product Launch
A Study of Product Launches in the Swedish Pharmaceutical Industry

2. Christian Plesner Rossing
International Transfer Pricing in Theory and Practice

3. Tobias Dam Hede
Samtalekunst og ledelsesdisciplin – en analyse af coachingsdiskursens genealogi og governmentality

4. Kim Pettersson
Essays on Audit Quality, Auditor Choice, and Equity Valuation

5. Henrik Merkelsen
The expert-lay controversy in risk research and management. Effects of institutional distances. Studies of risk definitions, perceptions, management and communication

6. Simon S. Torp
Employee Stock Ownership:
Effect on Strategic Management and Performance

7. Mie Harder
Internal Antecedents of Management Innovation

8. Ole Helby Petersen
Public-Private Partnerships: Policy and Regulation – With Comparative and Multi-level Case Studies from Denmark and Ireland

9. Morten Krogh Petersen
'Good' Outcomes. Handling Multiplicity in Government Communication

10. Kristian Tangsgaard Hvelplund
Allocation of cognitive resources in translation - an eye-tracking and key-logging study

11. Moshe Yonatany
The Internationalization Process of Digital Service Providers

12. Anne Vestergaard
Distance and Suffering
Humanitarian Discourse in the age of Mediatization

13. Thorsten Mikkelsen
Personligsheds indflydelse på forretningsrelationer

14. Jane Thostrup Jagd
Hvorfor fortsætter fusionsbølgen udover "the tipping point"?
– en empirisk analyse af information og kognitioner om fusioner

15. Gregory Gimpel
Value-driven Adoption and Consumption of Technology: Understanding Technology Decision Making

16. Thomas Stengade Sønderskov
Den nye mulighed
Social innovation i en forretningsmæssig kontekst

17. Jeppe Christoffersen
Donor supported strategic alliances in developing countries

18. Vibeke Vad Baunsgaard
Dominant Ideological Modes of Rationality: Cross functional

integration in the process of product innovation

19. Throstur Olaf Sigurjonsson
Governance Failure and Icelands's Financial Collapse

20. Allan Sall Tang Andersen
Essays on the modeling of risks in interest-rate and inflation markets

21. Heidi Tscherning
Mobile Devices in Social Contexts

22. Birgitte Gorm Hansen
Adapting in the Knowledge Economy Lateral Strategies for Scientists and Those Who Study Them

23. Kristina Vaarst Andersen
Optimal Levels of Embeddedness The Contingent Value of Networked Collaboration

24. Justine Grønbæk Pors
Noisy Management A History of Danish School Governing from 1970-2010

25. Stefan Linder
Micro-foundations of Strategic Entrepreneurship Essays on Autonomous Strategic Action

26. Xin Li
Toward an Integrative Framework of National Competitiveness An application to China

27. Rune Thorbjørn Clausen
Værdifuld arkitektur Et eksplorativt studie af bygningers rolle i virksomheders værdiskabelse

28. Monica Viken
Markedsundersøkelser som bevis i varemerke- og markedsføringsrett

29. Christian Wymann
Tattooing The Economic and Artistic Constitution of a Social Phenomenon

30. Sanne Frandsen
Productive Incoherence A Case Study of Branding and Identity Struggles in a Low-Prestige Organization

31. Mads Stenbo Nielsen
Essays on Correlation Modelling

32. Ivan Häuser
Følelse og sprog Etablering af en ekspressiv kategori, eksemplificeret på russisk

33. Sebastian Schwenen
Security of Supply in Electricity Markets

2012

1. Peter Holm Andreasen
The Dynamics of Procurement Management - A Complexity Approach

2. Martin Haulrich
Data-Driven Bitext Dependency Parsing and Alignment

TITLER I ATV PH.D.-SERIEN

1992

1. Niels Kornum
Servicesamkørsel – organisation, økonomi og planlægningsmetode

1995

2. Verner Worm
Nordiske virksomheder i Kina
Kulturspecifikke interaktionsrelationer ved nordiske virksomhedsetableringer i Kina

1999

3. Mogens Bjerre
Key Account Management of Complex Strategic Relationships
An Empirical Study of the Fast Moving Consumer Goods Industry

2000

4. Lotte Darsø
Innovation in the Making
Interaction Research with heterogeneous Groups of Knowledge Workers creating new Knowledge and new Leads

2001

5. Peter Hobolt Jensen
Managing Strategic Design Identities
The case of the Lego Developer Network

2002

6. Peter Lohmann
The Deleuzian Other of Organizational Change – Moving Perspectives of the Human

7. Anne Marie Jess Hansen
To lead from a distance: The dynamic interplay between strategy and strategizing – A case study of the strategic management process

2003

8. Lotte Henriksen
Videndeling
– om organisatoriske og ledelsesmæssige udfordringer ved videndeling i praksis

9. Niels Christian Nickelsen
Arrangements of Knowing: Coordinating Procedures Tools and Bodies in Industrial Production – a case study of the collective making of new products

2005

10. Carsten Ørts Hansen
Konstruktion af ledelsesteknologier og effektivitet

TITLER I DBA PH.D.-SERIEN

2007

1. Peter Kastrup-Misir
Endeavoring to Understand Market Orientation – and the concomitant co-mutation of the researched, the re searcher, the research itself and the truth

2009

1. Torkild Leo Thellefsen
Fundamental Signs and Significance effects
A Semeiotic outline of Fundamental Signs, Significance-effects, Knowledge Profiling and their use in Knowledge Organization and Branding

2. Daniel Ronzani
When Bits Learn to Walk Don't Make Them Trip. Technological Innovation and the Role of Regulation by Law in Information Systems Research: the Case of Radio Frequency Identification (RFID)

2010

1. Alexander Carnera
Magten over livet og livet som magt
Studier i den biopolitiske ambivalens