Structured Classification for Multilingual Natural Language Processing

A thesis presented
by
Philip Blunsom

to
The Department of Computer Science and Software Engineering
in total fulfillment of the requirements
for the degree of
Doctor of Philosophy

University of Melbourne
Melbourne, Australia
June 2007
Structured Classification for Multilingual Natural Language Processing

Abstract

This thesis investigates the application of structured sequence classification models to multilingual natural language processing (NLP). Many tasks tackled by NLP can be framed as classification, where we seek to assign a label to a particular piece of text, be it a word, sentence or document. Yet often the labels which we'd like to assign exhibit complex internal structure, such as labelling a sentence with its parse tree, and there may be an exponential number of them to choose from. Structured classification seeks to exploit the structure of the labels in order to allow both generalisation across labels which differ by only a small amount, and tractable searches over all possible labels. In this thesis we focus on the application of conditional random field (CRF) models (Lafferty et al., 2001). These models assign an undirected graphical structure to the labels of the classification task and leverage dynamic programming algorithms to efficiently identify the optimal label for a given input. We develop a range of models for two multilingual NLP applications: word-alignment for statistical machine translation (SMT), and multilingual supertagging for highly lexicalised grammars.

The first half of this thesis is dedicated to the task of word-alignment for SMT, which aims to find a mapping from words in a source language sentence to words in a target translation sentence. We treat this problem as one of structured classification with a CRF, where the input is the parallel sentence pair and the output is the index for each word in the source sentence of its aligned translation (or null) in the target sentence. By exploiting the ability of the CRF model to incorporate a diverse range of features we are able to explore many binary and real-valued features. Orthographic and syntactic features are defined which aim to generalise sparse word-to-word translation features. In addition, we define powerful features from unsupervised generative models, and collocation statistics derived...
from a large parallel corpus. The resultant models achieve higher precision and recall than the previous benchmark generative system. By training models using both maximum likelihood and maximum margin algorithms we are able to compare the performance of the two, and exploit biased loss-functions in order to create alignments with either high recall, or high precision.

The second part of this thesis investigates the task of tagging the words in a sentence with lexical types, or categories, from highly lexicalised grammars, referred to as supertagging. We describe the use of a pseudo-likelihood approximation for training a CRF with the large number of tags required to represent the lexical type inventory of a lexicalised grammar. We take an approach independent of language or grammar-formalism, defining a compact feature set extracted from the sequence of tags, the lexemes in the sentence, and the parameterised character set of the language in question. Two separate applications of our supertagging model are described and explored. The first is deep lexical acquisition (DLA), where we supertag sentences with the aim of identifying novel lexical items proposed by the model which can then be passed to a grammar developer for possible inclusion into the grammar. The second application is parse search space reduction, in which the supertagger is used as a multi-tagger to produce a restricted set of probable tags for each word in a sentence. These restricted sets of supertags can then be used by a parser to remove analyses inconsistent with them from its search space. We present experiments on both these applications, with multiple grammar formalisms and languages, which demonstrate the effectiveness of our multilingual approach.

In total, through the applications described in this thesis, we assess and extend the capabilities of CRFs for multilingual structured classification tasks. Our investigation of multiple training objective criteria, diverse features including ones extracted from unsupervised models, and novel label structures all have applications to NLP tasks beyond those presented here.
# Contents

Title Page ...................................................... i  
Abstract ....................................................... iii  
Table of Contents ............................................ v  
List of Figures ................................................ ix  
List of Tables ................................................. xi  
Declaration ..................................................... xiii  
Citations to Previously Published Work ......................... xiv  
Acknowledgments .............................................. 1  

1 Introduction ................................................. 2  
1.1 Aims and Contributions .................................... 5  
1.2 Thesis Structure .......................................... 6  

I Models ......................................................... 10  

2 Linear Classification ........................................... 11  
2.1 The Support Vector Machine ............................. 13  
2.2 The Maximum Entropy Classifier ......................... 16  
2.3 Discussion .................................................. 19  
2.4 Summary .................................................... 21  

3 Structured Classification with Graphical Models .................. 22  
3.1 Generative Models ......................................... 25  
3.1.1 Hidden Markov Models ................................ 25  
Definition ...................................................... 27  
Evaluation ..................................................... 29  
Decoding ....................................................... 31  
Estimation ..................................................... 32  
3.1.2 Summary ................................................ 34  
3.2 Discriminative Models .................................... 34  
3.2.1 Maximum Entropy Markov Models ..................... 34
### II Word Alignment

#### 4 CRFs for Word Alignment

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 Introduction</td>
<td>55</td>
</tr>
<tr>
<td>4.2 Statistical Machine Translation</td>
<td>58</td>
</tr>
<tr>
<td>4.3 Word Alignment</td>
<td>63</td>
</tr>
<tr>
<td>4.3.1 Heuristics</td>
<td>64</td>
</tr>
<tr>
<td>4.3.2 The IBM models for alignment: GIZA++</td>
<td>65</td>
</tr>
<tr>
<td>4.3.3 Discriminative approaches</td>
<td>68</td>
</tr>
<tr>
<td>4.3.4 Evaluation measures</td>
<td>73</td>
</tr>
<tr>
<td>4.4 Experimental Setup</td>
<td>76</td>
</tr>
<tr>
<td>4.4.1 Data</td>
<td>77</td>
</tr>
<tr>
<td>Canadian Hansards</td>
<td>77</td>
</tr>
<tr>
<td>4.5 A CRF for Word Alignment</td>
<td>78</td>
</tr>
<tr>
<td>4.5.1 Conditional random fields for word-alignment</td>
<td>78</td>
</tr>
<tr>
<td>4.5.2 The alignment model</td>
<td>81</td>
</tr>
<tr>
<td>4.5.3 Symmetrisation</td>
<td>83</td>
</tr>
<tr>
<td>4.5.4 Features</td>
<td>83</td>
</tr>
<tr>
<td>4.5.5 All positive features</td>
<td>95</td>
</tr>
<tr>
<td>4.5.6 Training with possible alignments</td>
<td>95</td>
</tr>
<tr>
<td>4.5.7 Model 4 as a feature</td>
<td>96</td>
</tr>
<tr>
<td>4.6 Results</td>
<td>96</td>
</tr>
<tr>
<td>4.7 Analysis</td>
<td>97</td>
</tr>
<tr>
<td>4.7.1 Feature contributions</td>
<td>99</td>
</tr>
<tr>
<td>4.7.2 Smoothing</td>
<td>99</td>
</tr>
<tr>
<td>4.7.3 Many-to-many alignments</td>
<td>101</td>
</tr>
<tr>
<td>4.8 Additional Language Experiments</td>
<td>102</td>
</tr>
<tr>
<td>4.8.1 Romanian</td>
<td>102</td>
</tr>
<tr>
<td>4.9 Summary</td>
<td>105</td>
</tr>
</tbody>
</table>
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.4.1 Baseline: Unigram Supertagger</td>
<td>158</td>
</tr>
<tr>
<td>8.4.2 Benchmark: CCG MEMM Supertagger</td>
<td>159</td>
</tr>
<tr>
<td>8.4.3 Results</td>
<td>159</td>
</tr>
<tr>
<td>8.5 Parsing Experiments</td>
<td>163</td>
</tr>
<tr>
<td>8.5.1 Parse Filtering</td>
<td>164</td>
</tr>
<tr>
<td>8.5.2 Chart Seeding</td>
<td>167</td>
</tr>
<tr>
<td>8.6 Summary and Future Directions</td>
<td>169</td>
</tr>
<tr>
<td>IV Conclusions and Further Work</td>
<td>171</td>
</tr>
<tr>
<td>9 Conclusion</td>
<td>172</td>
</tr>
<tr>
<td>9.1 Extensions and Further Research</td>
<td>177</td>
</tr>
</tbody>
</table>
# List of Figures

2.1 Multiclass classification margin scenarios ........................................... 14  
3.1 Examples of classification tasks ......................................................... 23  
3.2 Named entity detection example .......................................................... 25  
3.3 Example Markov process ......................................................................... 26  
3.4 Example hidden Markov model. ............................................................... 27  
3.5 A trellis algorithm .................................................................................. 29  
3.6 Structured classification with a HMM ....................................................... 32  
3.7 Structured classification with a MEMM ................................................... 35  
3.8 Structured classification with a CRF ...................................................... 37  
3.9 Viterbi algorithm .................................................................................... 40  
3.10 The backtracking step of the Viterbi algorithm ...................................... 41  
3.11 Forward-backward algorithm ................................................................... 42  
3.12 HMM/CRF feature representation .......................................................... 51  
4.1 Word alignment with a CRF ..................................................................... 56  
4.2 IBM Model 3 ........................................................................................... 59  
4.3 A simple word-alignment example. .......................................................... 60  
4.4 Phrase based translation process .............................................................. 61  
4.5 Phrase extraction from word alignments ............................................... 62  
4.6 Generative HMM alignment process ......................................................... 66  
4.7 Maximum weight bipartite matching min-flow graph .............................. 72  
4.8 A word-aligned example from the Canadian Hansards French-English corpus 74  
4.9 The effect of alignment errors on phrases extracted by a translation system .......................................................... 76  
4.10 Representation of the directional alignment process ............................... 82  
4.11 Effect of adding NullM1Sum feature ...................................................... 87  
4.12 The effect of adding the RelPosition feature. ........................................ 89  
4.13 Effect of $ExactMatch$ feature ............................................................... 91  
4.14 The effect of adding the $AbsLDiff$ feature. ........................................... 94  
4.15 An example from the Hansard test set, showing the effect of removing the Markov features from the MAP-CRF model. .................................................. 100
List of Figures

4.16 Relationship between the Gaussian sigma parameter of the MAP-CRF and precision, recall and f-score on the trial set. 101
4.17 Relationship between the C soft-margin parameter of the MM-CRF and precision, recall and f-score on the trial set. 102
5.1 Relationship between alignment metrics and $BLEU$ 120
6.1 The CRF model for supertagging an input sentence using CCG categories as tags. 124
6.2 Existence-based lexical features 128
6.3 The features extracted for the training instance associated with the node for $to$. 129
6.4 Performance of the MAP and pseudo-likelihood trained models with respect to the number of training sentences. 131
7.1 ERG novel lexical item performance versus training sample size. 144
7.2 JACY novel lexical item performance versus training sample size. 145
7.3 The effect of applying a cutoff value on the marginal probabilities when predicting new lexical types for the JACY grammar. 146
7.4 The effect of applying a cutoff value on the marginal probabilities when predicting new lexical types for the ERG grammar. 147
8.1 Elementary tree structures (supertags) for a simple sentence. 150
8.2 CCG categories (supertags) for the words in a simple sentence. 152
8.3 The full CCG parsing model. 154
8.4 Pseudo-likelihood conditioning. 159
8.5 CCG Tags per-word. 161
8.6 JACY Tags per-word. 162
8.7 ERG Tags per-word. 164
8.8 ERG tag accuracy versus training sample size. 165
8.9 JACY tag accuracy versus training sample size. 166
8.10 ERG parse filtering. 168
List of Tables

4.1 Statistics for the parallel corpus. ................................. 77
4.2 Results comparing Dice and Model 1 features ................. 85
4.3 Results comparing contributions of sentence position features. . 88
4.4 Results comparing contributions of orthographic features on the trial set. . . 90
4.5 Results comparing contributions of part-of-speech tag features. .............. 92
4.6 Results comparing contributions of Markov features. ...................... 93
4.7 Adding a global null feature increases performance. .................... 94
4.8 The effect that using possible alignments in training has on many-to-many alignments for the trial set. ........................................ 95
4.9 Adding Model 4 features .............................................. 96
4.10 CRF results on the English-French Hansards test data using the Best feature set ......................................................... 97
4.11 GIZA++ baseline on the English-French Hansards test data. ............ 98
4.12 The resulting F1 after removing groups of features from the Best feature set. 98
4.13 The performance of each feature group on many-to-one alignment accuracy. 103
4.14 Results evaluating the utility of the Dictionary feature for the Romanian-English trial data set. ........................................ 103
4.15 Results on the Romanian test data set with the MAP-CRF model. .......... 104
5.1 BLEU results from translation systems derived from a variety of alignments. 114
5.2 Statistical significance: 95% confidence interval and pairwise sampling of CRF models versus GIZA++. Significance is measure as described in 5.1.4 , using the method of Koehn, 2004b. .................................................. 114
5.3 Alignment system effect on decoder parameters. ........................... 115
5.4 End-to-end results using alignments with varying recall/precision scores, listed in order of alignment f-score. The Spearman rank correlation between alignment recall and BLEU is 0.36, between precision and BLEU is -0.36, and f-score and BLEU is -0.63. ...................................................... 116
6.1 Extracted feature types for the CRF model. .............................. 127
6.2 Wall Street Journal PoS corpus statistics. .............................. 130
6.3 Results of Wall Street Journal PoS tagging .............................................. 130

7.1 Make-up of the English Resource Grammar (ERG) and JACY grammars and treebanks. ......................................................... 138
7.2 Results of supertagging for the ERG ...................................................... 141
7.3 Results of supertagging for the JACY ..................................................... 141
7.4 Individual results when predicted new lexical types for the ERG ............ 143
7.5 Individual results when predicted new lexical types for the JACY grammar 143

8.1 Corpus statistics for the English Resource Grammar (ERG), JACY and CCG treebanks ................................................................. 157
8.2 Results of CCG supertagging at various values of $\beta$ for WSJ section 00 160
8.3 Results of JACY supertagging at various values of $\beta$ ............................ 160
8.4 Results of ERG supertagging at various values of $\beta$ ............................ 163
8.5 Parsing the test set with PET ............................................................... 167
8.6 ERG parse chart seeding ................................................................. 167
Declaration

This is to certify that

1. the thesis comprises only my original work towards the PhD except where indicated in the Preface

2. due acknowledgement has been made in the text to all other material used

3. the thesis is less than 100,000 words in length, exclusive of tables, maps, bibliographies and appendices.

(Philip Blumsom)
Citations to Previously Published Work

Some of the material contained in this thesis has been previously published. The basis of the work presented in Part II of this thesis was first published in:


The basis of the work presented in Part III of this thesis was first published in:


Some results presented in the above publications differ from those presented in this thesis due to changes in the training data and feature sets used. Both these publications were predominately my own work, with minor contributions from the co-authors.
Acknowledgments

I am deeply grateful to my supervisors, Tim Baldwin, Steven Bird and James Curran, who provided valuable guidance and advice throughout my studies. Tim’s constant encouragement and enthusiasm were a driving force in my research. Steven’s experience and advice significantly influenced this thesis, his guidance early in my studies helped establish my research direction. The many interesting and challenging conversations with James were priceless in informing my ideas and shaped this thesis.

I want to thank my colleagues at the University of Melbourne. In particular my discussions and friendship with Trevor Cohn provided an invaluable source of feedback and motivation. My many interesting conversations over lunch with Edward Ivanovic were a welcome diversion from the rigours of study.

I am profoundly grateful to my parents, Ann and Tony, who have given me endless support and encouragement throughout my education. Most of all I’d like to thank my partner Jacquie who’s unfailing support and love are a constant source of happiness and inspiration.
Chapter 1

Introduction

This thesis investigates the application of structured sequence classification models to multilingual natural language processing (NLP). NLP seeks to develop tools and resources for working with natural language texts. These developments range from the annotation of corpora for linguistic investigation, algorithms for extracting targeted information from online texts, to automatic translation systems. Although progress on the difficult problem of full natural language understanding (NLU) has been limited, much progress has been made on sub-problems such as assigning syntactic interpretations to written sentences and detecting mentions of entities of interest in text. Recent advances have been driven by the application of increasingly sophisticated statistical machine learning algorithms to the linguistic representations and corpora developed by previous symbolic NLP work.

Many of the successes of statistical processing in NLP have been characterised by shallow tagging of words in context with small tag-sets (e.g. part-of-speech (PoS) tagging, named entity recognition). However, increasingly complex statistical systems, such as modern phrase-based machine translation decoders, have started to take machine learning into NLP domains traditionally viewed as requiring too much semantic interpretation to be modelled by simple statistics. These applications of machine learning to NLP fit into the category of classification. Normally a large annotated corpus is used to learn weights for a classification model, which can then be used to classify new instances. This classification can take the form of simply selecting the class with the highest “score” from a small set, or can involve a complex search over an exponential number of classes. The second scenario
is referred to as **structured classification** and is characterised by the decomposition of the structure of the problem in order to allow the efficient identification of the optimal label with respect to the model weights, without enumerating all possible labels.

Structured classification takes statistical NLP beyond simply assigning a tag to a word and builds the structure of the language task directly into the classification model. The structure of the classification task may be the sequential nature of tags assigned to words, the tree structure of a syntactic analysis for a sentence, or a process of phrase translation and reordering in a statistical translation system. A classic example is syntactic parsing, where we specifically encode the parent-child relationships between word spans that we seek to classify with a syntactic category, allowing the classification to be performed at the more natural tree level, rather than for individual word-spans.

In this thesis we are concerned with the application of discriminative structured sequence classification. Statistical approaches to NLP tasks have tended to follow a particular pattern of investigation: first a generative model is proposed and evaluated successfully, and then an equivalent discriminative model is described which offers more flexibility and improved performance. For example, generative hidden Markov models initially proved successful for the classic case of Penn Treebank style PoS tagging. Subsequently, discriminative equivalents to these models have been described, such as maximum entropy Markov models and conditional random fields, which offer improved results. A similar progression from generative to discriminative is true of syntactic parsing, and this thesis contributes to this process for statistical machine translation and supertagging.

The conditional random field (CRF) models we employ assign an undirected graph structure to the labels of the classification task and leverage dynamic programming algorithms to efficiently identify the optimal label for a given input (Lafferty et al., 2001). These models have a number of attractive attributes to recommend them:

- They are discriminative models which focus on modelling the optimal choice of label for a given observation. This stands in contrast to generative models which model a joint distribution over labels and observations; such a joint distribution contains more information than strictly required by classification. By focusing on modelling a discriminating function, and not on the probabilities of the observations,
discriminative models can more easily incorporate additional information sources (features) and regularisation.

- Structured classification is a natural fit for many NLP tasks which require the labelling of observations which contain sub-parts (e.g. words in a sentence, sentences in a document etc.). Our CRF sequence models allow the tagging of words in a sentence to be modelled as applying a single sequence label, containing a tag for each word, to a sentence observation. This allows the inter-dependencies amongst word tags to be accounted for and explicitly modelled in a globally optimal fashion.

- CRFs allow for a diverse range of, possibly inter-dependent, features to be extracted from the observation. The ability to easily integrate information from a variety of sources through features gives discriminative models a significant advantage over generative models. Many NLP applications benefit from this ability to model observations on a range of levels. For example we can model sentences on the word, morpheme and orthographic levels for PoS tagging. In addition these features can summarise outputs from other classification models, including generative ones, providing an effective ability for model combination.

- CRFs describe the Markov sequence nature of a structured classification task, but do not restrict what criterion is used to set the feature weights. Traditionally CRFs have used a probabilistic maximum likelihood objective function in training, but models have also been presented using perceptron weight updates and maximum margin objectives. Each training method has particular strengths and such choices give us added flexibility when tackling NLP tasks.

- CRFs have a number of simple options for regularising the model, from probabilistic models with Gaussian priors to soft margins for maximum margin models. Such regularisation allows the models to be robust in the presence of many uninformative features and can be easily tuned on a development set. This stands in contrast to many generative models that require complex accounting for probability mass when smoothing.
• Using dynamic programming algorithms it is possible to tractably train and test optimal models. This allows us to have confidence that we can locate the optimal label for a sequence without suffering errors due to sub-optimal heuristic beam searches or sampling.

1.1 Aims and Contributions

The primary aim of this thesis is to demonstrate the application of CRF models to multilingual sequence classification tasks. In doing so we explore complex label and feature representations which extend existing CRF applications.

The first application we explore is word-alignment for statistical machine translation (SMT). In order to apply CRFs to word-alignment we first need to recast the alignment problem into one that is amendable to sequence labelling. As there are no labels in the traditional sense, we need to generalise the interpretation of labels to indexes from a foreign sentence into a target one. In doing so we create a CRF in which the number of labels is not fixed \textit{a priori}, and in which a label for one sentence has no correspondence with that same label applied to another sentence. Instead we derive features from what is implied by a label: aligned word pairs, rather than the label itself.

In addition to the description of the alignment model, we perform extensive feature engineering of feature types ranging from real valued features derived from generative alignment models, to binary features derived from syntactic information such as part-of-speech tags. In particular, the inclusion of features based on probabilities output from unsupervised models trained on a large number of sentence aligned examples allows our approach to be viewed as a novel and effective semi-supervised model, deriving its modelling power from a small amount of supervised data and a large volume of unsupervised data. Finally, using our model we are able to report results with f-scores far in excess of the benchmark generative system and show that the improved alignments produced by our model lead to improved translations when incorporated into an end-to-end translation system.
The second task on which we demonstrate the modelling flexibility of CRFs is multilingual supertagging for lexicalised grammars. This task requires a different approach to finding the CRF weights as the standard dynamic programming algorithm scales quadratically in the number of supertags, while the very large supertag sets require linear scaling in order to be practical. We apply a pseudo-likelihood inspired method for setting the model parameters that breaks apart the sequence graph and trains each node individually. In addition we show how this approach subsumes the existing maximum entropy Markov model approach and improves upon it by removing its left to right bias.

A significant contribution of this work is a simple and generic feature set which we demonstrate can achieve high performance across different languages and lexicalised grammars. We evaluate this feature set on a novel application of supertagging to deep lexical acquisition, which aims to identify new lexical types for insertion into a grammar, and parse search space reduction, which presents an automatic parser with a reduced set of categories for each word in order to improve parsing efficiency. A particular strength of this work is the multilingual and formalism neutral character of the proposed CRF.

In addition to the above two applications, an underlying theme of this work is to reinforce the separation of the CRF model structure and the algorithm chosen to set the feature weights. Where practical we present results using both the standard log-linear conditional probability CRF formulation alongside a maximum margin based approach. In doing so we emphasise that the training criteria can be selected in accordance with the demands of the task performed, be it hard classification, a strength of the maximum margin approach, or a distribution over labels offered by the conditional model.

1.2 Thesis Structure

This thesis is organised into three distinct parts. Part 1 introduces linear classification (Chapter 2) and the structured sequence models we utilise in this thesis (Chapter 3). In Part 2 we describe the application of structured classification to word-alignment for machine translation (Chapter 4), and evaluate these alignment models on real translation tasks (Chapter 5). Part 3 includes an introduction to supertagging for highly lexicalised
grammars (Chapter 6), an application of supertagging to multilingual deep lexical acquisition (Chapter 7), and the use of supertagging to improve automatic parsing performance and scalability (Chapter 8).

Below we summarise each chapter of this thesis:

**Chapter 2: Linear classification** In this chapter we introduce linear classification as the argmax of a dot product between a weight vector and the features active for an observation. We describe three criteria upon which we can base a choice of the weight vector. The first is to seek the best discriminator by maximising the classification margin between the desired class and the rest. Secondly we describe the generative approach of explicitly modelling the joint probability of an observation and a class label. The third is to model the conditional probability of a class label given an observation using a log-linear model, avoiding explicitly modelling the probability of an observation as is done by the generative approach, but still estimating a probability. We conclude this chapter by drawing attention to the similarities and differences between the three approaches.

**Chapter 3: Structured sequence classification** We introduce the CRF sequence models that we use throughout this thesis. We first describe the hidden Markov model (HMM) which is the most popular generative approach to sequence modelling in NLP. We then describe the clique structure of CRFs and the dynamic programming algorithms that can be used to calculate optimal labels and marginal distributions over graph nodes. We present two methods for setting the features weights: the maximum a posteriori (MAP) approach which casts the CRF as a log-linear conditional probability model and finds the most probable parameters given the training sample and a Gaussian prior distribution over those parameters; and the maximum margin approach which seeks the parameters that maximise the classification margin of the model on the training sample. In this presentation we reinforce the separation of the model structure and the criteria used to set the feature weights.

**Chapter 4: CRFs for word-alignment** Word-alignment is a core task in most modern statistical machine translation (SMT) systems. In this chapter we present a novel
application of CRFs to word-alignments. We show how word-alignment can be cast as structured sequence classification by interpreting the sequence states as indexes into an aligned sentence and deriving features from the word pairing implied by this indexing, rather than the sequence state itself. Using this model we present a series of feature engineering experiments designed to select the best feature set from the diverse range of features supported by the discriminative approach. We then present test results and analysis of our model on a French-English alignment test set.

Chapter 5: End-to-end translation with discriminative word-alignments Having described our alignment models, which achieve state-of-the-art performance when intrinsically evaluated on an aligned test set, we show in this chapter that they also lead to improved translation performance when integrated with a phrase-based SMT decoder. Additionally, we present experiments and analysis aimed at detecting if any of the intrinsic word-alignment evaluation metrics are correlated with extrinsic translation quality metrics.

Chapter 6: CRFs for supertagging We introduce the task of supertagging for heavily lexicalised grammars. We show how this task can be viewed as structured sequence classification, where lexical category assignments for words which exhibit long range dependencies can be significantly disambiguated with local sequence information. We describe the pseudo-likelihood variant of the CRF which is able to scale to the large label set and number of training instances required for supertagging.

Chapter 7: Supertagging for deep lexical acquisition We describe how supertagging can be used to acquire novel lexical types for a precision grammar. By tagging words in context with lexical types, novel items can be presented to a grammar developer for incorporation into the grammar. In addition we show how a simple threshold on the marginal probabilities of the conditional model can be employed to control the precision of the suggested type-set passed on to the grammar developer.

Chapter 8: Parse search space reduction This chapter presents results of applying our supertagging model to the established task of restricting the search space of automatic syntactic parsers. By using the pseudo-likelihood CRF as a multi-tagger we can
suggest a restricted set of lexical categories for each word in a sentence, even words which do not appear in the grammar. In addition we show how the pseudo-likelihood approach improves upon previous maximum entropy Markov model approaches by removing their left-to-right bias.

Chapter 9: Conclusion Finally we conclude the thesis by reflecting upon our application of structured classification. We discuss future directions for research into applying state-of-the-art machine learning algorithms to increasingly more difficult NLP tasks.
Part I

Models
Chapter 2

Linear Classification

In this chapter we provide background for our subsequent structured classification models by introducing and comparing a number of linear classification algorithms. This presentation separates the structure of the classifier, a weighted sum of features derived from the input, from the algorithm used to set the feature weights. By doing so we can compare the popular training algorithms of maximum margin and maximum entropy classification and characterise their utility and complexity in various classification scenarios.

In natural language processing (NLP) we are often confronted with tasks that require assigning a label, from a finite set of labels, to an input characterised by a set of attributes. We may wish to assign a part-of-speech (PoS) tag to a word observed in context, or to label a question with the type of answer that would satisfy it (number, person’s name, etc.). Such problems can be modelled as instances of classification, where we seek a function \( h : \mathcal{X} \rightarrow \mathcal{Y} \) which maps an input \( x \in \mathcal{X} \) to a class label \( y \in \mathcal{Y} \).

This thesis is concerned with applications of linear classification, in which we assume that the input and label pairs \((x, y)\) are characterised by a set of features, which are extracted by feature functions \( f : (\mathcal{X}, \mathcal{Y}) \rightarrow \mathbb{R}^k \). A label is assigned to an input by maximising a weighted sum of the features extracted by \( f \):

\[
h_w(x) = \arg \max_{y \in \mathcal{Y}} \sum_{i=0}^{k} w_i f_i(x, y)
= \arg \max_{y \in \mathcal{Y}} \langle w, f(x, y) \rangle
\]  

(2.1)
where the classification function $h_w$ is parameterised by the weight vector $w$, and $<\cdot, \cdot>$ is the inner product.

A primary task when building a linear classifier for a language processing application is the choice of feature functions. Machine learning for language applications is characterised by the highly heterogeneous set of feature functions that are commonly used. While other domains can often use a homogeneous feature representation, such as lists of pixel values in image recognition, NLP systems employ a diverse range of annotations (parse trees, collocation statistics, morphology etc.) as sources for features. For example, in a system for classifying an occurrence of the word *bank* in context from a set of predefined word senses we may define the following binary valued feature function:

$$f_i(x, y) = \begin{cases} 1, & \text{if } (\text{river} \in x) \land (y = \text{river\_sense}) \\ 0, & \text{otherwise} \end{cases}$$

where the input $x$ is a set of words found ‘near’ the occurrence of *bank*, and *river\_sense* is the class corresponding to the *bank of a river* sense. Here we make use of a wide variety of feature functions, from binary ($f_i : (X, Y) \mapsto \{0, 1\}$) to real valued ($f_i : (X, Y) \mapsto \mathbb{R}$).

The process of setting the feature weights $w$ is referred to as training, or model estimation. In the statistical approach to classification we assume a sample of $n$ training inputs $D = (x^i, y^i)$ which have been drawn i.i.d (independent and identically distributed) from an underlying distribution. The weights are set in order to optimise an objective function over the training sample, chosen according to a particular theory of classification. This is referred to as inductive learning where the objective function aims to find weights on the training sample which will generalise to good performance on the global distribution. The two theories that we utilise in this thesis are: maximum entropy (logistic) and maximum margin (support vector machine) classification. Section 2.1 describes maximum margin classification in which the aim is to find the classifier that maximises the difference between the value of the product in Equation (2.1) for correct and incorrect labels, for all example inputs in the training sample. Section 2.2 introduces maximum entropy classification, which assigns a probabilistic interpretation to classification and seeks the conditional distribution with maximum entropy which fits the training sample.
2.1 The Support Vector Machine

Support Vector Machines, introduced by Vapnik, 1995, choose the weights of Equation (2.1) that maximise the **margin** between the correct class, and the incorrect class with the highest value of the $\langle w, f(x, y) \rangle$ product.\(^1\) The intuition behind such a criterion is simple: we want to choose the classifier that is maximally confident in separating the correct from the incorrect classes on the training sample $D$.

We define the margin for training instance $i$ as:

$$\gamma_i = \langle w, f(x^i, y^i) \rangle - \max_{y \in Y \backslash y^i} \langle w, f(x^i, y) \rangle$$

(2.2)

Assuming a classifier exists that can correctly classify all training inputs (i.e. the training sample is linearly separable), we can express maximum margin training as finding the maximum $\gamma$ such that all of the individual margins $\gamma_i$ for the training inputs are greater than this value. This **hard-margin** objective is described by the following constrained optimisation problem:

$$\max_{\gamma, w : \|w\| = 1} \gamma$$

s.t. $\forall i, \forall y \in Y \backslash y^i : \langle w, \Delta f_i(y) \rangle \geq \gamma$

where

$$\Delta f_i(y) = f(x^i, y^i) - f(x^i, y)$$

Note that in this equation we fix the norm of the weight vector to be equal to one ($\|w\| = 1$), in order to make maximising the margin well posed. Without this restriction we could maximise Equation (2.2) in degenerate cases by simply making the weights of the correct classes arbitrarily large.

We can rewrite Equation (2.3) as a standard convex quadratic program by equivalently minimising the size of the weight vector $\|w\|$ while defining the size of the margin to be fixed to one ($\gamma = 1$). This margin formulation is graphically depicted in Figure 2.1.

\(^1\)Here we follow the multi-class SVM formulation of Crammer and Singer, 2001
Figure 2.1. Multiclass classification margin scenarios for a training input. The vertical axis represents the size of the classification inner product and each point depicts a labelling of the instance. The gold standard is shaded. (a) shows the case where the margin between the correct and incorrect class is greater than, or equal to, one, and no constraints are violated. (b) shows a correct classification, but within the margin and thus the margin constraint is violated. In (c) the classification is incorrect, the margin is negative and the constraints are violated.

The assumption that the training sample is linearly separable may not be true. We can account for this by introducing slack variables $\xi_i$ which allow for violations of the margin:

$$
\min_{\mathbf{w}, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{n} \sum_i \xi_i
$$

subject to

$$
\forall i, \forall y \in \mathcal{Y} \setminus y^i : \langle \mathbf{w}, \Delta f_i(y) \rangle \geq 1 - \xi_i
$$

The slack variables $\xi_i$ have the effect of penalising the misclassification of training examples in the case that $\mathcal{D}$ is not linearly separable, while the user defined parameter $C$ (scaled by the number of training inputs $n$) sets how strong this penalty should be. Thus the optimisation now balances finding a large margin and minimising misclassified training inputs, referred to as regularisation. This formulation is known as the soft-margin SVM.
Most approaches to solving this optimisation operate on the **dual** equation of Equation (2.4), derived using the method of Lagrange multipliers. A dual \( \lambda_{iy} \) variable is added for each margin constraint (\( \langle w, \Delta f_i(y) \rangle \geq 1 - \xi_i \)) in Equation (2.4):

\[
L(w, \Xi, \Lambda) = \frac{1}{2} \|w\|^2 + \frac{C}{n} \sum_i \xi_i
\]

\[
+ \sum_{i, y \in \mathcal{Y} \setminus y'} \lambda_{iy} \left( \langle w, \Delta f_i(y) \rangle - (1 - \xi_i) \right)
\]

\[
\text{s.t. } \forall i, \forall y \in \mathcal{Y} \setminus y' : \lambda_{iy} \geq 0.
\]

where the capital Greek letters \( \Lambda \) and \( \Xi \) represent the set of \( \lambda_{iy} \) and \( \xi_i \) variables respectively. This equation is referred to as the Lagrangian and we seek its saddle point, found at the minimum with respect to the primal variables \( (w, \Xi) \), and the maximum w.r.t the dual variables \( (\Lambda) \). By setting \( \frac{\partial L}{\partial \xi_i} = 0 \) and \( \frac{\partial L}{\partial w} = 0 \) we derive:

\[
w^* = \sum_{i, y \in \mathcal{Y} \setminus y'} \lambda_{iy} \Delta f_i(y)
\]

This equation states that the optimal weights \( (w^*) \) of the maximum margin classifier are a linear combination of the dual variables and the training inputs.

Substituting the above derivations into Equation (2.5) we eventually arrive at the dual objective function:

\[
\Theta_{\text{SVM}}(\Lambda) = - \frac{1}{2} \sum_{i, y \neq y'} \sum_{j, \bar{y} \neq y'} \lambda_{iy} \lambda_{j\bar{y}} \langle \Delta f_i(y), \Delta f_j(\bar{y}) \rangle + \sum_{i, y \neq y'} \lambda_{iy}
\]

Thus training a SVM can be achieved by finding the parameters \( \Lambda^* \) which maximise the dual objective function \( \Theta_{\text{SVM}} \), subject to the constraints introduced by the slack variables:

\[
\Lambda^* = \arg \max_\Lambda \Theta_{\text{SVM}}(\Lambda)
\]

\[
\text{s.t. } \forall i : \lambda_i \geq 0, \sum_{y \neq y'} \lambda_{iy} \leq \frac{C}{n}
\]

\[\text{2The inner product } \langle \Delta f_i(y), \Delta f_j(\bar{y}) \rangle \text{ is referred to as the kernel function } (k(\cdot, \cdot)) \text{ and, for the linear feature representations employed in this thesis, is simply the dot product:}
\]

\[\kappa(a, b) = \langle a, b \rangle\]

Although discussions of SVMs are often linked to discussions of the various non-linear kernels that they may employ, in this thesis we focus on the comparison of the maximum margin criteria versus maximum likelihood models and thus restrict our discussion to linear kernels. By decoupling the optimisation objective function from the feature representation the effect of each can be assessed more clearly. In addition, it is possible to train linear kernel SVMs in linear time, rather than the standard quadratic training time of arbitrary kernels (Joachims, 2006).
This quadratic program can be solved using standard constrained convex optimisation
techniques. However, Crammer and Singer, 2001 describe a more efficient approach,
suitable for this multiclass SVM formulation, which makes use of the fact that many of
the $\lambda_{iy}$ parameters will be zero. Training focuses on optimising the weights of the most
violated constraints first, as the zero weights mostly correspond to the instances, as shown
in Figure 2.1 (a), where the margin constraints are not violated. The training input/label
pairs for which the corresponding $\lambda_{iy}$ are non-zero are referred to as support vectors, as
these inputs are all that are needed to define the linear classifier according to Equation (2.6).
Solutions to the SVM optimisation which have many zero dual variables are referred to as
sparse.

2.2 The Maximum Entropy Classifier

The SVM aims to learn a classifier by maximally separating the correct classes from
the incorrect. A different approach to classification is to learn a probability distribution of a
class given an input $p(y|x)$, and classify a new input by identifying its most probably class.
These conditional classification models take the form:

$$h_w(x) = \arg \max_{y \in Y} p(y|x)$$

The model estimation problem then becomes: given the observed training sample $D = (x^i, y^i)$, of all the conditional distributions $p$ that agree with $D$, which one should we
choose?

A particularly popular criterion amongst the language processing community is to
choose the distribution having the maximum entropy (Jaynes, 1957). Research by Ratnaparkhi, 1996 and Berger et al., 1996 established maximum entropy classifiers as particularly
effective for a range of NLP tasks where a diverse set of feature functions could be defined.
The intuition behind the maximum entropy criterion is that, given a set of probability
functions that fit $D$, we should choose the one that assumes as little as possible about
what is not in $D$. That is, choose the most uniform distribution, using entropy as a measure
of uniformity. An extreme counter example to this would be to choose a distribution that
simply memorised the classes of the training inputs and for all unseen inputs assigned the
same class with probability one. Such a choice would be assuming a lot about the unseen inputs, namely that they all share the same class, and the distribution chosen would have a low entropy and would likely generalise very poorly.

Finding the maximum entropy model is an example of constrained optimisation and has two components: calculating the conditional entropy objective function of the distribution which we wish to maximise, and matching the constraints imposed by the training sample, i.e. the predictions of the distribution must equal those observed in the training sample. The conditional entropy of a probability distribution $p$ is defined as:

$$H(p) = - \sum_{x \in X, y \in Y} p(y, x) \log_2(p(y|x))$$

$$= - \sum_{x \in X, y \in Y} p(x)p(y|x) \log_2(p(y|x))$$

(2.9)

As the sum over every possible input $x$ is impractical, we approximate the probability of an input by its empirical probability $\tilde{p}(x)$. The empirical probability of an input is equal to the number of times it was observed in training sample:

$$H(p) \approx - \sum_{x \in X, y \in Y} \tilde{p}(x)p(y|x) \log_2(p(y|x))$$

$$= - \sum_{i=0}^{n} \sum_{y \in Y} p(y|x^i) \log_2(p(y|x^i))$$

(2.10)

where $n$ is the number of training inputs.

We encode the constraints that the distribution chosen must agree with $D$ by stating that for each feature function $f_i$, its empirical expectation on $D$ (the number of times it occurred in the training data) must equal the expectation predicted by the distribution (the number of times the model would predict it to occur):

$$E_p[f_i] = E_{\tilde{p}}[f_i] : 0 \leq i \leq k$$

(2.11)

where the predicted expectation of a feature is:

$$E_p[f_i] = \sum_{j=0}^{n} \sum_{y \in Y} p(y|x^j) f_i(x^j, y)$$
and the empirical expectation is simply the sum of the value of the feature for all the training inputs:

\[ E_\tilde{p}[f_i] = \sum_{j=0}^{n} f_i(x^j, y^j) \quad (2.12) \]

Given the definition of conditional entropy, and the feature expectation constraints, we can formulate the maximum entropy criteria as an instance of constrained optimisation:

\[
 p^* = \arg \max_p - \sum_{i=0}^{n} \sum_{y \in Y} p(y|x^i) \log_2(p(y|x^i)) \quad (2.13)
\]

\[
 \text{s.t.} \quad \forall j : E_p[f_j] = E_\tilde{p}[f_j]
\]

Using the method of Lagrange multipliers we introduce a dual variable \( \lambda_i \) for each feature constraint, reformulating this constrained optimisation problem as one of unconstrained optimisation:

\[
 p^*_\Lambda = \arg \max_p H(p) + \sum_{i=0}^{k} \lambda_i(E_p[f_i] - E_\tilde{p}[f_i]) \quad (2.14)
\]

where \( k \) is the number of feature functions. By setting the derivative of this equation to zero and solving for \( p \) we find that the maximum entropy distribution has the form:

\[
 p_{\Lambda}(y|x) = \frac{1}{Z_{\Lambda}(x)} e^{\sum_{i=0}^{k} \lambda_i f_i(x, y)} \quad (2.15)
\]

\[
 Z_{\Lambda}(x) = \sum_{y \in Y} e^{\sum_{i=0}^{k} \lambda_i f_i(x, y)}
\]

At this point it can be observed that if we take the \( \arg \max \) of the logarithm of Equation (2.15), the maximum entropy classifier fits Equation (2.1) with the Lagrange multipliers equivalent to the weights (\( w = \Lambda \)). This illustrates why the maximum entropy model is also known as a log-linear model.

The weights \( w \) can be found as the solution of the Lagrangian dual equation:

\[
 \Theta_{MLE}(w) = \sum_{i=0}^{n} \left( - \log Z_w(x^i) + \sum_{j=0}^{k} w_j f_j(x^i, y^i) \right) \quad (2.16)
\]

It is easy to verify that Equation (2.16) is also the log-likelihood of the training sample, thus the maximum entropy model is also the maximum likelihood model.
Maximum likelihood models have a tendency to overfit the training sample, and thus generalise poorly to unseen data, especially when the training sample is small. A small sample may not give a good representation of the whole distribution, and thus fitting the constraints exactly could be too trusting. In order to address this issue we regularise the model by assuming a prior probability distribution over the weights, this has the effect of allowing the model to approximately fit the constraints. We define the probability distribution of weight vector values to be proportional to a Gaussian distribution with zero mean \( p(w) \propto e^{-\frac{\|w\|^2}{2\sigma^2}} \) and take the maximum a posteriori (MAP) estimate. The MAP estimate chooses the most probable weights given the prior probability of weights and the training set. This has the effect of penalising extreme weight values:

\[
    w^* = \arg \max_w \Theta^{MLE}(w) - \frac{\|w\|^2}{2\sigma^2}
\]  

(2.17)

For example, if our part-of-speech tagging training set only contained \textit{saw} as a verb the unregularised model would be inclined to assign a very high weight to this case, making an alternate labelling very unlikely. In the case of the regularised model, the weight assigned would be constrained, thus leaving the possibility that if \textit{saw} was encountered as a noun in testing, features from the surrounding context suggesting this label may overcome the preference for a verb.

Equation (2.17) is convex and can be solved using standard convex optimisation techniques. Malouf, 2002 provides a comprehensive comparison of the performance of such techniques and shows that off-the-shelf convex optimisation approaches such as quasi-Newtonian methods work well.

### 2.3 Discussion

Having introduced two powerful discriminative linear classifiers, it is of interest to compare and contrast the objective functions they use for estimating the feature weights, and the effect this has on the classification tasks they will be most applicable to. We
can write the objective functions for both classifiers in a way which emphasises their similarities:

\[
\text{MaxEnt} : \arg \max_w -\frac{1}{2\sigma^2} \|w\|^2 - \sum_{i=0}^{n} \left( \log Z_w(x_i) - \langle w, f(x_i, y_i) \rangle \right)
\]

\[
\text{SVM} : \arg \max_w -\frac{1}{2} \|w\|^2 - \frac{C}{n} \sum_{i=0}^{n} \left( \max_{y \neq y_i} [1 - \langle w, \Delta f_i(y) \rangle] \right)
\]

Both formulations minimise the norm of the weight vector for regularisation, and some measure of loss on the training sample. We see that the parameters \(C\) and \(\frac{1}{\sigma^2}\) control the tradeoff between fitting the training sample and regularisation, thus the only difference between the maximum entropy and the SVM models selection criteria is their choice of loss functions. The maximum entropy model uses the log-loss: \(\log Z_w(x_i) + \langle w, f(x_i, y_i) \rangle\), which leads to a dense feature weight vector, but allows the output of the classifier to be interpreted as a conditional probability. The SVM employs the hinge-loss: \(\max_{y \neq y_i} [1 - \langle w, \Delta f_i(y) \rangle]\), which implies that only the relationship between the correct class and the highest ranked incorrect class affects the estimation of the model weights. Thus the SVM tends to produce sparse weight vectors which can make it more efficient than the maximum entropy model as only non-zero features need to be processed. However, the trade-off for this is that the output of a SVM cannot be easily interpreted as a conditional probability.\(^3\) Conditional probabilities are often useful in NLP processing task where a particular classifier is part of a pipeline of processing and the ability to avoid hard decisions by providing confidence estimates can improve robustness. In particular, in many NLP tasks human annotators have difficulty agreeing on the correct labels to assign, so choosing a classifier which produces a distribution over possible labelings may often be well motivated. Curran et al., 2006 demonstrated that by passing probabilities from a log-linear PoS tagger to a parser, rather than a single tag, the parsing performance could be improved. However, it is also possible to acquire confidence estimates from a SVM, as demonstrated by Tong and Koller, 2001.

A further point of consideration is the ability of the models to incorporate non-linear kernels. The SVM is often employed with non-linear kernels, which can enable the

---

\(^3\)Platt, 1999 presents a comparison of methods for directly estimating probabilities with SVMs. Although some of these models are shown to adequately model posterior probabilities, the complexity of the approaches is high and they have little to recommend them over simply using a log-linear model.
efficient modelling of extremely large feature spaces. However, although less common, the Representer theorem can also be used to formulate log-linear models as kernel machines.

From this comparison we can formulate the guideline that when we simply wish to classify inputs, the SVM is the most appropriate choice. If we require confidence estimates for our classifications in the form of conditional probabilities, or to return a set of most probable labels, then the maximum entropy model is a better choice.

## 2.4 Summary

In this chapter we have introduced a selection of linear classifiers that act as a primer to our further discussion of structured classification. We described a SVM classifier which assigns a class label to an input, and a conditional log-linear model which estimates a conditional probability distribution for each label. Both models balance fitting the training data against minimising the absolute size of the features weights. The maximum entropy model trades the sparse weight vector solutions of the SVM for conditional probabilities.
Chapter 3

Structured Classification with Graphical Models

Natural language processing is replete with machine learning applications which are most naturally expressed as learning a mapping from an input to an output, where both possess internal structure which we wish to exploit. For example, sentences (whose structure consists of a sequence of words) are frequently the input to NLP models which map them to a label consisting of a sequence of tags (e.g. PoS tagging, named entity recognition, shallow ‘chunk’ parsing). The applications explored in this thesis are framed as instances of this **structured modelling** and are solved by positing a graphical structure on the predicted output labels.

In Chapter 2 we introduced classification as the task of learning a mapping from an input observation characterised by a set of features, to an atomic label. But many tasks involving language can have extremely large label sets. Figure 3.1 illustrates two such tasks: Penn Treebank style part-of-speech (PoS) tagging and syntactic parsing. In the PoS tagging example each input sentence is classified with a label consisting of the sequence of PoS tags corresponding to each word in the sentence, while in the parsing example the label consists of a syntactic parse tree. If we treat these tasks as simple classification we would have to deal with extremely large label sets. For PoS tagging, where we may have fifty or more possible tags, then our set of possible labels for a $T$ word sentence would be $50^T$. For parsing, the space of possible syntax trees could be much larger. Clearly these
Chapter 3: Structured Classification with Graphical Models

Figure 3.1. Examples of classification tasks in which both the input observation and the output label exhibit a graphical structure.

Tasks cannot be tractably treated as simple classification where we enumerate and score all possible labels. However, if we exploit the structure in the inputs and outputs we can create tractable models for a large number of these structured classification problems.

In an extension to linear classification (Chapter 2), we can model structured classification as seeking a function \( h : \mathcal{X} \rightarrow \mathcal{Y} \) which maps a structured input observation \( x \in \mathcal{X} \) to a structured label \( y \in \mathcal{Y} \). Despite their complex and varied derivations, all the models we present in this chapter can be viewed as structured linear classifiers of the form:

\[
h_w(x) = \arg \max_{y \in \mathcal{Y}} w^T f(x, y)
\]  

(3.1)

where the only difference from Equation (2.1) is that now the labels \( y \) are assumed to have a structure which the feature functions can exploit.

In this chapter we explore a number of approaches to structured modelling which we group into generative (Section 3.1) and discriminative models (Section 3.2). Although
there are alternate ways of viewing these models, we explicitly introduce and discuss them from the point of view of structured classification. We aim to illustrate that these diverse models, despite their varied histories and independent motivations and derivations, can all be viewed as instances of structured classification, and the differences between them lie in the way they decompose the observation and label structure, and the objective function they optimise to estimate their parameters.

The classification tasks focused upon in this thesis are ones in which the label can be represented as a sequence of states. Formally, each label is a $T$ dimensional sequence of states, indexed by $t$, and drawn from a finite state space $S$: $y \in S^T$. We can also refer to the maximal cliques of the graph, whether directed or undirected, formed by linking each state in the label with its neighbours:

$$C(y) = \{(y_{t-1}, y_t) : 1 \leq t < T\}$$

(3.2)

The prototypical generative model we explore is the hidden Markov model (HMM), which has a long history of successful application in language processing (Rabiner, 1990). We introduce the HMM in Section 3.1.1 as a probabilistic sequence model, describe how it can be trained from a corpus of examples, and introduce the Viterbi dynamic programming algorithm as a way of efficiently using the model to assign a label to an input sequence. In Section 3.2 we explore models based upon undirected graphical structures which allow additional flexibility through the incorporation of diverse features derived from the input sequence and output graph, and form the basis of the algorithms applied later in this thesis. We first introduce the historical forerunner to these models, the maximum entropy Markov model (MEMM) (McCallum et al., 2000), and describe its limitations. We then move on to conditional random fields (CRFs) in Section 3.2.2, which provide a principled framework for the discriminative modelling of structured classification tasks which can be decomposed using a Markov assumption (Lafferty et al., 2001). In Section 3.3 we explore how dynamic programming algorithms make these models efficient to implement and in Section 3.4 we describe two methods for deriving the parameters for these models: maximum a posteriori and maximum margin. We also describe the pseudo-likelihood approximation of CRFs that can be used to efficiently estimate the model parameters with a reduced complexity.
3.1 Generative Models

In this section we introduce generative structured classification, using the example of the hidden Markov model. Generative models describe a process by which the structure of the input is generated from the structure of the output. In the case of the PoS tagging example in Figure 3.1 we can think of each PoS tag generating each word in the input according to a probability distribution over all words. Such generative models estimate the joint probability of an observation and its label \( p(x, y) \). Classification can be performed by using Bayes’ rule to calculate the conditional probability \( p(y|x) \).

3.1.1 Hidden Markov Models

The HMM is a powerful statistical tool for modelling generative sequences that can be characterised by an underlying state process generating a sequence of observations. HMMs have found application in many areas interested in signal processing, and in particular speech processing, but have also been applied with success to low level NLP tasks such as PoS tagging, phrase chunking, and named entity recognition. Andrei Markov gave his name to the mathematical theory of Markov processes in the early twentieth century (Markov, 1913), but it was Baum and his colleagues that developed the theory of HMMs in the 1960s (Baum et al., 1970). Here we loosely follow Rabiner, 1990 which provides an excellent introduction to the topic.

Figure 3.2 illustrates a simple named entity recognition task which serves as an example of structured classification. In this example the label consists of a state tag for each word in the input sentence. These tags declare whether the word is the start of a named entity.
Figure 3.3. Example Markov process.

(person, company, place etc.) (B), the continuation of a named entity (I), or neither (O). Using this example we can introduce Markov and hidden Markov processes:

**Markov Processes** Figure 3.3 depicts an example of a Markov process. The model presented describes a simple model for our named entity recognition example. The model has three states, B, I, and O, corresponding to the label tags. Each state can emit a word from its own observation set, where the same word cannot be observed from two different states. The model is a finite state automaton, with probabilistic transitions between states. Given a sequence of observations, e.g. *Bill Gates Buys*, we can easily verify that the state sequence that produced those observations was: B—I—O, and the probability of the sequence is simply the product of the probability of starting in state B (given by the $\pi$ array) with the transition probabilities, in this case $0.2 \times 0.6 \times 0.6$.

**Hidden Markov Models** The Markov model described includes the unrealistic restriction that a particular word can only be emitted from one state. Obviously there are many words that can be both part of a named entity and not part of one, for example *Apple*. Figure 3.4 shows an example of how the previous model can be extended into an HMM. The new model now allows all observation words to be emitted from multiple
states with a finite probability. This change makes the model much more expressive and able to better represent our intuition, in this case that Apple can refer to the computer company as part of a named entity, but is more likely to be a fruit. The key difference is that now if we have the observation sequence Million Apple Stock then we cannot say exactly what state sequence produced these observations and thus the state sequence is ‘hidden’. However, we can calculate the probability that the model produced the sequence, as well as the state sequence that was most likely to have produced the observations. The next three sections describe the common calculations that we can perform on a HMM.

**Definition**

The formal definition of a HMM is as follows:

$$\Lambda = (A, B, \pi)$$

(3.3)
$S$ is our state alphabet set, and $V$ is the observation alphabet set:

$$S = (s_1, s_2, \ldots, s_q) \quad (3.4)$$
$$V = (v_1, v_2, \ldots, v_m) \quad (3.5)$$

We define the label $y$ to be a fixed state sequence of length $T$, and corresponding observations $x$:

$$y = y_1, y_2, \ldots, y_T \quad (3.6)$$
$$x = x_1, x_2, \ldots, x_T \quad (3.7)$$

$A$ is a transition array, storing the probability of state $j$ following state $i$. Note the state transition probabilities are independent of time:

$$A = [a_{ij}], \quad a_{ij} = p(y_t = s_j | y_{t-1} = s_i) \quad (3.8)$$

$B$ is the observation array, storing the probability of observation $k$ being produced from the state $i$, independent of $t$:

$$B = [b_i(k)], \quad b_i(k) = p(x_t = v_k | y_t = s_i) \quad (3.9)$$

$\pi$ is the initial probability array:

$$\pi = [\pi_i], \quad \pi_i = p(y_1 = s_i) \quad (3.10)$$

Two assumptions are made by the model. The first, called the Markov assumption, states that the current state is dependent only on the previous state, this represents the memory of the model:

$$p(y_t | y_1^{t-1}) = p(y_t | y_{t-1}) \quad (3.11)$$

The independence assumption states that the output observation at time $t$ is dependent only on the current state, it is independent of previous observations and states:

$$p(x_t | x_1^{t-1}, y_1^t) = p(x_t | y_t) \quad (3.12)$$

These two assumptions are of key importance in allowing efficient model estimation and inference using dynamic programming.
Figure 3.5. A trellis algorithm. The vertical axis represents model states and the horizontal axis records time steps through the input. The calculation for each node at time $t$ is a function of the values at all of the nodes at $t-1$.

Evaluation

Given a HMM $(\Lambda = (A, B, \pi))$, and a sequence of observations, we’d like to be able to compute $p_\Lambda(x)$, the probability of the observation sequence given a model parameterised by $\Lambda$. This problem could be viewed as one of evaluating how well a model predicts a given observation sequence, and thus allow us to choose the most appropriate model from a set.

The probability of the observations $x$ for a specific label $y$ is given by the product of the probabilities of each observation at time $t$ being emitted by the state at time $t$:

$$p_\Lambda(x|y) = \prod_{t=1}^{T} p_\Lambda(x_t|y_t) = b_{y_1}(x_1) \times b_{y_2}(x_2) \times \cdots \times b_{y_T}(x_T)$$  \hspace{1cm} (3.13)$$

and the probability of the state sequence is given by the product of the probabilities of each state:

$$p_\Lambda(y) = \pi_{y_1} a_{y_1 y_2} a_{y_2 y_3} \cdots a_{y_{T-1} y_T}$$  \hspace{1cm} (3.14)$$
So we can calculate the probability of the observations given the model by the product of the probability of transitioning into a particular state and emitting the observation, for all possible labels:

\[
p_{\Lambda}(x) = \sum_{y} p_{\Lambda}(x|y)p_{\Lambda}(y) = \sum_{y} \pi_{y_1}b_{y_1}(x_1)a_{y_1y_2}b_{y_2}(x_2)\ldots a_{y_{T-1}y_T}b_{y_T}(x_T)
\]

This equation allows the evaluation of the probability of \( x \), but to evaluate it directly would be exponential in \( T \) as we would need to enumerate every label of length \( T \).

A better approach is to recognise that many redundant calculations would be made by directly evaluating Equation (3.16), and therefore caching calculations can lead to reduced complexity. We implement the cache as a trellis of states at each time step, calculating the cached valued (called \( \alpha \)) for each state as a sum over all states at the previous time step. \( \alpha \) is the probability of the partial observation sequence \( x_1, x_2 \ldots x_t \) and state \( s_i \) at time \( t \). Figure 3.5 depicts this process.

We define the forward probability variable:

\[
\alpha_t(i) = p_{\Lambda}(x_1, x_2 \ldots x_t, y_t = s_i)
\]

so if we work through the trellis filling in the values of \( \alpha \) the sum of the final column of the trellis will equal the probability of the observation sequence. The algorithm for this process is called the forward algorithm and is as follows:

1. Initialisation:

\[
\alpha_1(i) = \pi_i \times b_i(x_1), \quad 1 \leq i \leq q
\]

2. Induction:

\[
\alpha_{t+1}(j) = \left( \sum_{i=1}^{q} \alpha_t(i) a_{ij} \right) \times b_j(x_{t+1}), \quad 1 \leq t \leq T-1, \quad 1 \leq j \leq q
\]

3. Termination:

\[
p_{\Lambda}(x) = \sum_{i=1}^{q} \alpha_T(i)
\]
For each state $s_j$, $\alpha_j(t)$ stores the probability of arriving in that state having observed the observation sequence up until time $t$.

It is apparent that by caching $\alpha$ values the forward algorithm reduces the complexity of calculations involved to $O(q2T)$ rather than $O(2^T q T)$. We can also define an analogous backwards algorithm which is the exact reverse of the forwards algorithm with the backwards variable:

$$\beta_t(i) = p_A(x_{t+1} x_{t+2} \ldots x_T | y_t = s_i)$$  \hspace{1cm} (3.20)

as the probability of the partial observation sequence from $t + 1$ to $T$, starting in state $s_i$.

**Decoding**

The aim of decoding is to discover the hidden state sequence that was most likely to have produced a given observation sequence. One solution to this problem is to use the Viterbi algorithm to find the single best state sequence for an observation sequence. The Viterbi algorithm is another trellis algorithm which is very similar to the forward algorithm, except that the transition probabilities are maximised at each step, instead of summed. First we define:

$$\delta_t(i) = \max_{y_1, y_2, \ldots, y_{t-1}} p_A(y_1, y_2, \ldots, y_t = s_i, x_1, x_2 \ldots x_t)$$  \hspace{1cm} (3.21)

as the probability of the most probable state path for the partial observation sequence.

The Viterbi algorithm for HMMs is as follows:

1. **Initialisation:**

   $$\delta_1(i) = \pi_i b_i(x_1), \quad 1 \leq i \leq q, \quad \psi_1(i) = 0$$  \hspace{1cm} (3.22)

2. **Recursion:**

   $$\delta_t(j) = \max_{1 \leq i \leq q} \left( \delta_{t-1}(i) a_{ij} \right) b_j(x_t), \quad 2 \leq t \leq T, \quad 1 \leq j \leq q$$  \hspace{1cm} (3.23)

   $$\psi_t(j) = \arg \max_{1 \leq i \leq q} \left( \delta_{t-1}(i) a_{ij} \right) b_j(x_t), \quad 2 \leq t \leq T, \quad 1 \leq j \leq q$$  \hspace{1cm} (3.24)
3. Termination

\[ P^* = \max_{1 \leq i \leq q} \left( \delta_T(i) \right) \]  
(3.25)

\[ y^*_T = \arg \max_{1 \leq i \leq q} \left( \delta_T(i) \right) \]  
(3.26)

4. Optimal state sequence backtracking:

\[ y^*_t = \psi_{t+1}(y^*_{t+1}), \quad t = T - 1, T - 2, \ldots, 1 \]  
(3.27)

The main difference with the forward algorithm in the recursion step is that we are maximising, rather than summing, and storing the state that was chosen as the maximum for use as a back-pointer. The backtracking allows the best state sequence to be found from the back pointers stored in the recursion step, but it should be noted that to find the second best state sequence a second back pointer would need to be stored at every trellis cell.

**Estimation**

Given a set of examples from a process, we would like to be able to estimate the model parameters \( \Lambda = (A, B, \pi) \) that best describe that process. There are two standard approaches to this task, dependent on the form of the examples, which will be referred to here as supervised and unsupervised training. If the training examples contain both the
inputs and outputs of a process, we can perform supervised training by equating inputs to observations, and outputs to states. If only the inputs are provided in the training data, then we must use unsupervised training to estimate a model that may have produced those observations. In this section we will discuss the supervised approach to training, which corresponds to standard classification, for a discussion of the Baum-Welch algorithm for unsupervised training see Baum et al., 1970.

The easiest solution for creating a model, $\Lambda$, is to have a large sample of training examples, $D$, each annotated with the correct classification. In Figure 3.6 the HMM configuration for our named entity detection example is depicted. In this case we define two sets:

- $t_1 \ldots t_3$ is the set of tags $B-I-O$, which we equate to the HMM state set $s_1 \ldots s_3$
- $w_1 \ldots w_m$ is the set of words, which we equate to the HMM observation set $v_1 \ldots v_m$

With this model we can frame named entity detection as decoding the most probable hidden state sequence of $B-I-O$ tags given an observation sequence of words. To determine the model parameters $\Lambda$, we can use maximum likelihood estimates (MLE) from a corpus containing sentences tagged with their correct named entities. For the transition matrix we use:

$$a_{ij} = p(t_i|t_j) = \frac{\text{Count}(t_i, t_j)}{\text{Count}(t_i)}$$

(3.28)

where $\text{Count}(t_i, t_j)$ is the number of times $t_j$ followed $t_i$ in the training data. For the observation matrix:

$$b_j(k) = p(w_k|t_j) = \frac{\text{Count}(w_k, t_j)}{\text{Count}(t_j)}$$

(3.29)

where $\text{Count}(w_k, t_j)$ is the number of times $w_k$ was tagged $t_j$ in the training data. And lastly the initial probability distribution:

$$\pi_i = p(q_1 = t_i) = \frac{\text{Count}(y_1 = t_i)}{\text{Count}(y_1)}$$

(3.30)

In practice when estimating a HMM from counts it is normally necessary to apply smoothing in order to avoid zero counts and improve the performance of the model on data not appearing in the training set.
3.1.2 Summary

In this section we have introduced the hidden Markov model as an example of a generative structured classification model. The HMM generative story describes a sequence of states emitting a sequence of observation symbols. The model is parameterised by the transition probabilities of one state following another, and the emission probabilities of an observation being emitted from a particular state. Using the forward-backward and Viterbi dynamic programming algorithms it is possible to efficiently calculate the probability of a model producing an observation sequence (evaluation) and the most likely state sequence given an observation sequence (decoding). Training a HMM given a full observed training set is particularly simple as it is possible to calculate the maximum likelihood parameters by simply counting observation and state occurrences in the training data.

3.2 Discriminative Models

Discriminative classifiers attempt to maximise the predictive ability of a model trained on a training set applied to a test set. As such these models can avoid explicitly modelling the probability of the observations \( p(x) \), making it considerably easier to incorporate expressive feature representations of the inputs. In this section we introduce two discriminative graphical models, the maximum entropy Markov model, in order to provide historical context, and the conditional random field models that we utilise in this thesis. Extending on the presentation in Chapter 2 we present conditional probabilistic and maximum margin methods for estimating the graphical model parameters and include a discussion of a pseudo-likelihood approximation which can be used to reduce training complexity.

3.2.1 Maximum Entropy Markov Models

The maximum entropy Markov model (MEMM) is an application of the maximum entropy classifier (Section 2.2) to sequence labelling (Ratnaparkhi, 1996). The structure

\[ \text{MEMM} \]

\[ \text{MEMM} \] was introduced by McCallum et al., 2000 for a model that is actually a restricted form of the model described in Ratnaparkhi, 1996, here we use MEMM to refer to the original model.
Chapter 3: Structured Classification with Graphical Models

of the MEMM is inspired by the HMM: each state in the label is conditioned on the previous state and the observation, as shown in Figure 3.7. Thus the global probability of the sequence is calculated as the product of the individual state distributions:

$$p(y|x) = \prod_t p(y_t|y_{t-1}, x, t)$$  \hspace{1cm} (3.31)

The probability of a state given the previous state, and the observation, is modelled using a maximum entropy classifier:

$$p_w(y_t|y_{t-1}, x, t) = \frac{1}{Z_w(y_{t-1}, x, t)} e^{\sum_{i=0}^k w_i f_i(y, y_{t-1}, x, t)}$$  \hspace{1cm} (3.32)

$$Z_w(y_{t-1}, x, t) = \sum_{y \in S} e^{\sum_{i=0}^k w_i f_i(y, y_{t-1}, x, t)}$$

The only difference here with a standard log-linear classifier is that the feature functions can now incorporate the previous state in the sequence, as well as the current state. An example feature function from Figure 3.7 that mirrors the HMM observation probability might be:

$$f_i(y_t, y_{t-1}, x, t) = \begin{cases} 
1, & \text{if } (x_t = \text{Apple}) \land (y_t = \text{B}) \\
0, & \text{otherwise}
\end{cases}$$

While a feature function modelling the HMM state transition probability could be:

$$f_i(y_t, y_{t-1}, x, t) = \begin{cases} 
1, & \text{if } (y_t = \text{O}) \land (y_{t-1} = \text{B}) \\
0, & \text{otherwise}
\end{cases}$$
However, with the MEMM we’re not limited to the above two features, we can arbitrarily combine properties of the observation, and the previous and current states, to get features such as:

\[
f_i(y_t, y_{t-1}, x, t) = \begin{cases} 
1, & \text{if } (x_t = \text{Gates}) \land (x_{t-1} = \text{Bill}) \\
\land (y_t = \text{I}) \land (y_{t-1} = \text{B}) \\
0, & \text{otherwise}
\end{cases}
\]

In addition we are not limited to using only the words from the observations, we could instead define features that use the suffix of a word combined with the current and/or previous state, for example.

We can train this model in the same way that we train a standard maximum entropy classifier, conceptually treating each state in each training label as a training instance and maximising the posterior probability of the training states given the observations and a Gaussian prior weight probability:

\[
w^* = \arg \max_w \sum_{i=0}^n \sum_{t=0}^{s_i} \left(- \log Z_w(y_{i-1}^i, x_i^t, t) + \sum_{j=0}^k w_j f_j(y_i^i, y_{i-1}^i, x_i^t) \right) - \frac{||w||^2}{2\sigma^2} \tag{3.33}
\]

This is a straight application the regularised log-linear model described in Section 2.2 to each clique in the input.

Once we have a trained model, we can use the Viterbi algorithm in a similar fashion to a HMM to decode the optimal label for a test observations. The application of the Viterbi algorithm to discriminative feature based models is described in more detail in Section 3.3.1.

A criticism of locally conditioned models such as the MEMM is that they can exhibit label bias (Lafferty et al., 2001). This problem occurs when a state has a low entropy next state distribution, an extreme case being when there is only one state that can follow that state. In the extreme case, the one state will have a probability of 1 of following the initial state, rendering any evidence from the input \( x \) irrelevant. This is an example of explaining away, observed in conditional models (Pearl, 1988). Although the possibility for label bias in the MEMM was cited by Lafferty et al., 2001 as motivation for the CRF, there is little evidence that it is actually a problem in practice. MEMM tagging models are usually induced from data, their next state distributions are fully connected,
and smoothing is employed. In addition a range of label and input features are defined, including features which incorporate the input and next state, but not the previous state. Such configurations mean that it is extremely unlikely that a single state transition could monopolise the probability mass. Klein and Manning, 2002 investigated this issue and also found label bias to be of limited concern. They did however suggest that the reverse problem of observation bias was more likely in tagging models. In this bias a word may only have been observed with a single tag, thus this tag may overwhelm any counter evidence from the previous state. Again, if smoothing and a range of features are employed, it is unlikely for this bias to become a problem.

3.2.2 Conditional Random Fields

Although the MEMM can deal with a number of the limitations of the HMM, in particular the ability to employ a diverse feature set and easy regularisation through the use of a Gaussian prior, it does not directly model the probability of a label given an observation \( p(y|x) \). Instead the MEMM estimates a conditional distribution for each state in the label, given the previous state. **Conditional random fields** (Lafferty et al., 2001) are a globally normalised model that directly models \( p(y|x) \). By performing a global normalisation for the probabilities of a state sequence the CRF’s features can be estimated more accurately, as all the features are trained in competition with each other. This global, rather than local, competition between features means that CRFs are not subject to the label
or observation bias problems. Thus CRFs are undirected graphical models which can be seen as the discriminative equivalent of generative directed HMMs, while we may view the MEMM as an approximation to the CRF. Sutton and McCallum, 2007a provides a useful introduction and discussion of CRFs.

The structure of a CRF for our PoS classification example is shown in Figure 3.8. We can see that, as for the MEMM, each state in the label is conditioned on the entire observation, however for the CRF the state sequence forms an undirected graph.

The form of the conditional probability model for the CRF is:

\[
p_w(y|x) = \frac{\exp \sum_t \sum_k w_k f_k(y_{t-1}, y_t, x, t)}{Z_w(x)}
\]  

(3.34)

Here \( t \) ranges over the indices of the observation sequence \( (x) \), \( k \) ranges over the model’s features, and \( w = \{w_k\} \) are the model parameters (linear weights for their corresponding features). The distribution is globally normalised by the partition function, \( Z_w(x) \), which sums out the numerator in (3.34) for every possible label:

\[
Z_w(x) = \sum_y \exp \sum_t \sum_k w_k f_k(y_{t-1}, y_t, x, t)
\]

Although these equations look similar to those of the MEMM, it is important to note that for CRFs the normalisation function \( Z(\cdot) \) is outside of the sum over \( t \), and thus the normalisation ranges over the entire set of features for a sequence, rather than a single state in the sequence.

As for our linear classifiers, the parameters of the CRF are estimated from a fully observed training sample \( \mathcal{D} \), by maximising an objective function on this sample. In Section 3.4 we describe both probabilistic and margin based objectives for training CRFs. The probabilistic methods require the use of the forward-backward algorithm, described in Section 3.3.2. The Viterbi algorithm is used to find the maximum posterior probability alignment for test sentences, \( y^* = \arg \max_y p_w(y|x) \). Both the forward-backward and Viterbi algorithm are dynamic programs which make use of the Markov assumption to calculate efficiently the exact marginal distributions.
3.3 Inference

The undirected graphical models discussed so far provide tools for decomposing the feature representations of structured classification tasks through independence assumptions. In order to use these decompositions for classification we need a method for finding the optimal label for an input given a set of weights, a task we refer to as decoding. In addition, for training models which optimise a probabilistic objective function we require the ability to calculate both marginal distributions over node states, and the normalised probability of a label given an observation, referred to as marginalisation. In Section 3.3.1 we describe how the Viterbi algorithm can be used to solve the decoding task, and in Section 3.3.2 we introduce the forward-backward algorithm for calculating the marginal distributions. Both these dynamic programming algorithms are instances of the belief propagation algorithm for graphical models as described by Pearl, 1988.

3.3.1 Viterbi algorithm

The Viterbi algorithm is an instance of the max-product algorithm and its application to HMMs was previously described in Section 3.1. Here we describe the application of this model to feature based linear classifiers, independent of whether they have probabilistic semantics or not. The aim of Viterbi decoding is to compute the label which has the maximum score for a given observation, i.e. to solve Equation (3.1).

The algorithm is depicted in Figure 3.9 and proceeds as follows:

1. **Initialisation**

   First we define the activation of a maximal clique having state assignment \((s_a, s_b)\):
   \[
   \Psi(s_a, s_b, x, t) = \langle w, f(s_a, s_b, x, t) \rangle
   \]

   Using this, and allowing an initial null state \(\emptyset\), we initialise the \(\delta\) variables for time \(t = 0\). The \(\delta_t\) variables store the feature weights product for the highest scoring state sequence from time \(0 \rightarrow t\).

   \[
   \forall s \in S, \quad \delta_0(s) = \Psi(\emptyset, s, x, t)
   \]
2. Recursion

The $\delta_t(s)$ variables are recursively updated by taking the maximum of the sum of the previous $\delta_{t-1}(s_{t-1})$ variables and the features product for the clique state pair $(s_{t-1}, s)$:

$$\forall s \in S, 1 \leq t < T, \quad \delta_t(s) = \max_{s_{t-1} \in S} \left( \delta_{t-1}(s_{t-1}) + \Psi(s_{t-1}, s, x, t) \right)$$

We use the matrix $b_t(s)$ to record which previous state $s_{t-1}$ lead to the calculated $\delta_t$ value:

$$\forall s \in S, 1 \leq t < T, \quad b_t(s) = \arg \max_{s_{t-1} \in S} \left( \delta_{t-1}(s_{t-1}) + \Psi(s_{t-1}, s, x, t) \right)$$

3. Termination & backtracking
Figure 3.10. The backtracking step of the Viterbi algorithm

The highest $\delta_{T-1}$ variable in the final column of the trellis indicates the finishing state of the maximising label:

$$q^*_{T-1} = \arg \max_{s \in S} \delta_{T-1}(s)$$

By following the $b_t(s)$ back-pointers we can recover the whole label:

$$0 \leq t < T - 1, \quad q^*_t = b_{t+1}(q^*_{t+1})$$

Again we note that there is nothing in the Viterbi algorithm that assumes that the model we are decoding has a probabilistic interpretation. In addition, nothing is assumed about how we arrived at the model weights $w$: we could have used a maximum-likelihood, maximum margin, or some other objective function. The algorithm is merely an efficient way of calculating the optimal label for an observation that exploits the decomposition of features across cliques, and avoids enumerating the exponential number of possible labels.

### 3.3.2 Forward-backward algorithm

When our model does have probabilistic semantics, such as a HMM or CRF, we may wish to calculate the **marginal probability**: $p(y_t = s|x, t)$, the probability that at time $t$ the label will have state $s$. In contrast with the Viterbi algorithm, which finds the single most
probable state sequence for the entire graph, the forward-backward algorithm can find the most probably state for a given graph node considering all possible state sequences passing through it. This calculation provides a distribution over states for each node in the input, useful for returning confidence estimates in a tagging task, or to return multiple ‘n-best’ tags for a node (multi-tagging). These probabilities are also required by the probabilistic training algorithms described in Section 3.4. We can calculate the marginal probabilities using the forward-backward algorithm, which is a specialisation of the sum-product algorithm for sequences (Pearl, 1988).
The algorithm is illustrated in Figure 3.11. We first define the recursive **forward probability** variables, $\alpha_t(s)$, which store the unnormalised probability of all partial labels starting at $t = 0$ and ending in state $s$ at time $t$:

$$\forall s \in S, \quad \alpha_0(s) = 1$$

$$\forall s \in S, 1 \leq t \leq T, \quad \alpha_t(s) = \sum_{s_{t-1} \in S} \left( \alpha_{t-1}(s_{t-1}) \times e^{\Psi(s_{t-1},s,x,t)} \right)$$

The similarity with the $\delta_t(s)$ variable of the Viterbi algorithm is clear, however in this case we sum, rather than maximise, at each time step.

We next define the **backward probability** variables that are recursively calculated from the end to the start of the sequence:

$$\forall s \in S, \quad \beta_{T-1}(s) = 1$$

$$\forall s \in S, T - 1 > t \geq 0, \quad \beta_t(s) = \sum_{s_{t+1} \in S} \left( \beta_{t+1}(s_{t+1}) \times e^{\Psi(s,s_{t+1},x,t)} \right)$$

These variables store the probability of all partial labels starting in state $s$ at time $t$ and finishing at the end of the sequence.

Given the definitions of the forward and backward probabilities we can calculate the probability of all labels that have state $s$ at time $t$:

$$p(y_t = s|x,t) = \frac{\alpha_t(s) \times \beta_t(s)}{Z(x)}$$

In addition we can also easily calculate the probability of a label having state $s_a$ at time $t - 1$ and state $s_b$ at time $t$:

$$p(y_{t-1} = s_a,y_t = s_b|x,t) = \frac{\alpha_{t-1}(s_a) \times e^{\Psi(s_a,s_b,x,t)} \times \beta_t(s_b)}{Z(x)}$$

where the normalisation function $Z(x)$ can be calculated from the sum of either the first column of $\beta$ variables, or the last column of $\alpha$ variables:

$$Z(x) = \sum_{s \in S} \beta_0(s) = \sum_{s \in S} \alpha_{T-1}(s)$$

The intuition for this calculation is that each state entry in the first column of the $\beta$ variables is the unnormalised probability of all labels starting with that state, thus the sum of these variables is the unnormalised probability of all labels.
3.4 Estimation

In Chapter 2 we discussed a number of approaches for training the weights of a linear classifier and these equally apply to the structured classification models described in this chapter. In this section we describe probabilistic maximum likelihood, maximum a posteriori, and maximum margin training, for conditional random field models. Additionally we describe a pseudo-likelihood approximation technique which decomposes the CRF graph into locally normalised nodes, reducing the complexity of training for both MAP and maximum margin optimisation methods.

3.4.1 Maximum Likelihood

Perhaps the most obvious training objective for CRFs is to find the weights $w$ which maximise the conditional log-likelihood of the training sample $D$. We refer to this as the maximum likelihood estimate ($w_{MLE}$), and note that, as described in Section 2.2, this is also the maximum entropy model.

The likelihood of the training sample given a weight vector is:

$$p(D|w) = \prod_{i=1}^{n} p(y^i|x^i, w)$$

thus we can express the log-likelihood of the sample as:

$$L_{MLE} = \log p(D|w)$$

$$= \sum_{i=1}^{n} \left( -\log Z_w(x^i) + \sum_{1 \leq t < |x^i|} \langle w, f(y^i_{t-1}, y^i_{t}, x^i, t) \rangle \right)$$

and the maximum likelihood weights are then found as the maximiser:

$$w_{MLE} = \arg \max_w L_{MLE}$$

If we compare Equation (3.41) and Equation (2.16), we see that the only difference is that in the case of the CRF we have decomposed the linear feature sum across the internal clique structure of the classification instance. This decomposition has the principle effect of making the calculation of the normalising function $Z_w(x^i)$ tractable through the forward-backward algorithm.
Equation 3.42 is convex and standard convex optimisation techniques can be used to solve it efficiently. Malouf, 2002 provides a comprehensive comparison of the performance of such techniques. Most of these techniques, in particular the popular quasi-Newtonian ones, require the calculation of the gradient of $L_{MLE}$ with respect to the model weights for each optimisation iteration. This gradient can be expressed as:

$$\frac{\partial L_{MLE}}{\partial w_k} = \sum_{i=1}^{n} \left[ \sum_{1 \leq t < |x_i|} f_k(y_{t-1}, y_t, x^i, t) - \sum_{y \in \mathcal{Y}} p(y|x^i, w) \sum_{1 \leq t < |x_i|} f_k(y_{t-1}, y_t, x^i, t) \right]$$

$$= E_{\tilde{p}}[f_k] - E_p[f_k] \quad (3.43)$$

As in Section 2.2 we see that, by setting the gradient equal to zero, the optimal weights are located at the point where the empirical feature expectations, counted from the training sample, equal the estimated feature expectations predicted by the model on that sample. The main difficulty with directly evaluating Equation (3.43) is the sum over all possible labels in the calculation of the feature expectations. However, by rearranging the order of summation we can express this sum as:

$$\sum_{y \in \mathcal{Y}} p(y|x^i, w) \sum_{1 \leq t < |x_i|} f_k(y_{t-1}, y_t, x^i, t) = \sum_{1 \leq t < |x_i|} \sum_{(s_a, s_b) \in S^2} p(y_{t-1} = s_a, y_t = s_b|x^i, w) f_k(s_a, s_b, x^i, t) \quad (3.44)$$

Using the forward-backward algorithm, and Equation (3.44), we can efficiently calculate the marginal probabilities $p(y_{t-1} = s_a, y_t = s_b|x, w)$, thus making the gradient calculation tractable.

### 3.4.2 Maximum a posteriori

As discussed in relation to the maximum entropy classifier (Section 2.2), maximum likelihood estimates have a tendency to over-fit the training sample. This over-fitting is especially common in NLP applications where the number of training observations is often small in comparison to the number of feature weights to be estimated.

In order to regularise the model we assume that the prior probability of the model weights is described by a zero mean Gaussian distribution:

$$p(w) \propto e^{-\frac{|w|^2}{2\sigma^2}} \quad (3.45)$$
This has the effect of penalising weights which are much greater, or much less than zero. We can now derive the maximum a posteriori (MAP) log-likelihood:\footnote{The true Bayesian MAP estimate would require the sum over the probability of all possible weight vectors. Here we approximate this distribution by its mode.}

\[
\mathcal{L}^{\text{MAP}} = \log p(D|w) + \log p(w) = \mathcal{L}^{\text{MLE}} - \frac{\|w\|^2}{2\sigma^2}
\]

Given this definition of $\mathcal{L}^{\text{MAP}}$ we can restate the regularised feature weight gradient as:

\[
\frac{\partial \mathcal{L}^{\text{MAP}}}{\partial w_k} = \frac{\partial \mathcal{L}^{\text{MLE}}}{\partial w_k} - \frac{w_k}{\sigma^2}
\]

which can be solved using the same approach as for $\mathcal{L}^{\text{MLE}}$, with the same complexity.

### 3.4.3 Pseudo-Likelihood

As described, the parameters of CRFs are usually estimated from a fully observed training sample, by maximising the likelihood of the labels given the observations. I.e. $w^{\text{MLE}} = \arg\max_w p_w(D)$, where $D = \{(y^i, x^i)\}$ is the complete set of training data.

However, as calculating $Z_w(x)$ for a CRF has complexity that is quadratic in the number of labels, it could prove useful to be able to approximate $p_w(y|x)$ in order to scale our model to hundreds of states and tens-of-thousands of training sequences. Here we describe the pseudo-likelihood approximation $p^{\text{PL}}_w$ (Besag, 1975; Li, 1994) in which the marginals for a node at time $t$ are calculated with its neighbour nodes’ labels fixed to those observed in the training data:

\[
U^{\text{PL}}_w(s, x, t) = \sum_k w_k(f_k(\hat{y}_{t-1}, s, x, t) + f_k(s, \hat{y}_{t+1}, x, t))
\]

\[
p^{\text{PL}}_w(y|x) = \prod_t \frac{\exp(U^{\text{PL}}_w(y_t, x, t))}{\sum_{s\in S} U^{\text{PL}}_w(s, x, t)}
\]

where $y_t$ is the state observed in the training data and $s$ ranges over the state set. This approximation removes the need to calculate the normalising partition function over the whole sequence, thus reducing the complexity to be linear in the number of states and
training instances. Intuitively, instead of normalising using the sum over every possible label sequence, we step through the graph nodes normalising each one over all states that node could take, given the nodes gold standard neighbour states. This approximation is closely related to the MEMM (Section 3.2.1), however in the pseudo-likelihood approach we symmetrically fix both the left and right neighbours of each node, rather than just the left as for the MEMM.

As for the other log-linear models, a zero-mean Gaussian prior $(p_0(w_k) \propto \exp \left(-\frac{w_k^2}{2\sigma_k^2}\right))$ is used for regularisation. This yields a log-pseudo-likelihood objective function of:

$$L_{PL} = \sum_{(y,x) \in D} \log p_{w}^{PL}(y|x) + \sum_k \log p_0(w_k)$$

As we cannot observe label values for the test data we must use $p_w(y|x)$ when decoding. The Viterbi algorithm is used to find the maximum posterior probability alignment for test sentences, $y^* = \arg \max_y p_w(y|x)$.

An alternative method of breaking apart the inference graph is piecewise training (Sutton and McCallum, 2005). Piecewise training is mostly applicable to non-tree (including sequence) structured graphs in which exact inference is intractable. As it has the same complexity as full CRF training it is not suitable for the large tagging problems presented in this thesis. Sutton and McCallum, 2007b presents a piecewise version of pseudo-likelihood conditioning which scales linearly with the number of labels and thus could be applied to large problems. In the sequence tagging case, piecewise pseudo-likelihood conditions separately on the previous and next tags of the graph node in question, rather than both as is the case for pseudo-likelihood, or just the previous, as is the case for the MEMM. We could expect this model to produce results very similar to the standard pseudo-likelihood model.

### 3.4.4 Maximum Margin

Although CRFs have most often been treated as probabilistic models, this is not inherent in their formulation. As described in Section 2.1, a popular alternative to maximum likelihood training is maximum margin training, as exemplified by the SVM. Taskar et al., 2003 showed that by decomposing the loss function across cliques, in the same manner as
the feature functions are decomposed, it is possible to tractably train CRF-like undirected graphical models with maximum margin objectives. Here we will describe the application of the Tsochantaridis et al., 2005 margin re-scaling structured classification formulation to CRFs. This formulation is a generalisation of that used by Taskar et al., 2003 to a wider range of loss functions.

Restating the maximum margin constrained optimisation objective, with the addition that for structured classification we now consider the label \( y \) to be a vector, rather than an integer:

\[
\min_{\mathbf{w}, \xi} \frac{1}{2} \| \mathbf{w} \|^2 + \frac{C}{n} \sum_i \xi_i \tag{3.49}
\]

s.t. \( \forall i, \forall y \in Y \setminus y^i : \mathbf{w}^T \Delta f_i(y) \geq 1 - \xi_i \)

where

\[
\Delta f_i(y) = f(x^i, y^i) - f(x^i, y) \tag{3.50}
\]

Inherent in this formulation is the assumption of a zero-one (\( l_{01} \)) loss function where the loss is zero if the label is correctly predicted, and one otherwise:

\[
l_{01}(y^i, y) = \begin{cases} 
1, & \text{if } y^i \neq y \\
0, & \text{otherwise} 
\end{cases} \tag{3.51}
\]

Such a loss function is not practical for most structured classification tasks. If we consider our PoS tagging example, we would like our loss function to recognise that if an incorrect label differs from the correct label by only a small number of predicted PoS tags it should incur less loss than a label which shares no PoS tags with the correct label. The \( l_{01} \) loss for an incorrect label is always 1, thus there is no differentiation between small and large differences in labels. A popular loss for sequence labelling tasks which exhibits this property, and was used by Taskar et al., 2003, is the Hamming loss:

\[
l_{Ham}(y^i, y) = \sum_{1 \leq t < |y|} l_{01}((y^i_{t-1}, y^i_t), (y_{t-1}, y_t)) \tag{3.52}
\]

This loss counts the matching cliques in the label Markov sequence. In the case of CRFs, we can employ any loss function that can be calculated from the decomposed sum of
functions of the sequence cliques. For the field of NLP, one such popular loss function that we could optimise is the f-score. Formally, loss functions \( l : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R} \) are assumed to assigned zero loss to the correct label \( l(y^i, y^i) = 0 \), and non-zero loss otherwise. In addition it is assumed that the loss is bounded for every possible \( y \in \mathcal{Y} \).

We can explicitly incorporate a non-zero/one loss function into the objective function by rescaling the margin according to the loss incurred:

\[
\min_{w, \xi} \frac{1}{2} \|w\|^2 + \frac{C}{n} \sum_i \xi_i \quad (3.53)
\]

s.t. \( \forall i, \forall y \in \mathcal{Y} \setminus y^i : \langle w, \Delta f_i(y) \rangle \geq l(y^i, y) - \xi_i \)

This has the effect of penalising the violation of a margin constraint that has a high loss more than one with a lower loss.\(^3\)

Using the margin re-scaling formulation we can derive a similar dual objective function to the one in Section 2.1:

\[
\Theta^{MM}(\Lambda) = -\frac{1}{2} \sum_{i, y \neq y^i} \sum_{j, y \neq y^i} \lambda_{iy} \lambda_{jy} k(\Delta f_i(y), \Delta f_j(y)) + \sum_{i, y \neq y^i} \lambda_{iy} l(y^i, y) \quad (3.54)
\]

where there is a \( \lambda_{iy} \) multiplier for each of the \( \langle w, \Delta f_i(y) \rangle \geq l(y^i, y) - \xi_i \) margin constraints.

As for the SVM, training is achieved by finding the parameters \( \Lambda^* \) which maximise the dual objective function \( \Theta^{MM} \), subject to the constraints introduced by the slack variables which now include the loss scaling:

\[
\Lambda^* = \arg \max_{\Lambda} \Theta^{MM}(\Lambda) \quad (3.55)
\]

s.t. \( \forall i : \lambda_i \geq 0, \sum_{y \neq y^i} \lambda_{iy} \leq \frac{C}{n} \)

\(^3\)An alternative to margin rescaling is to rescale the slack variables:

\[
\forall i, \forall y \in \mathcal{Y} : \langle w, \Delta f_i(y) \rangle \geq 1 - \frac{\xi_i}{l(y^i, y)}
\]

This method has the advantage that it is invariant under scaling the loss function. However many loss functions, including the Hamming loss, don’t decompose easily for this approach, making it more complex to implement.
The optimal $\Lambda^*$ from the dual equation can be used to recover the linear classifier weights $w$ using a weighted linear combination of training examples:

$$w^* = \sum_i \sum_{y \neq y_i} \lambda_{iy} \Delta f_i(y)$$

At this point we note that the maximum margin formulation described has not assumed anything about the structure of the classification task to which it is to be applied, except that we can calculate the loss for a predicted label. However, the parameterisation of the objective in terms of the $\lambda_{iy}$ variables is problematic as it implies a sum over (a possibly exponential number of) all labels $y \in Y$. As a result it is not practical to apply standard constrained quadratic optimisation solvers to this objective and it is necessary to use an optimisation strategy tailored to this structured classification problem. This contrasts with maximum likelihood training in which off-the-shelf convex optimisation solvers are efficient.

Tsochantaridis et al., 2005 describe a cutting plane algorithm that solves the quadratic program by iteratively finding the most violated constraint $\lambda_{iy}$ for each training instance, adding the constraint to a working set, and solving a reduced optimisation problem with respect to this working set. The core of this algorithm is the cost function which is computed for each training instance on each iteration:

$$H(y) = l(y_1, y) - w^T \Delta f_i(y)$$

(3.56)

The maximiser of this equation determines the most violated constraint:

$$\hat{y} = \arg \max_{y \in Y} H(y)$$

(3.57)

If the constraint is violated by more than a threshold, it is then added to the working set for this training instance and the dual equation is optimised with respect to this reduced set.

---

4 The equivalent cost function for the slack re-scaling formulation is:

$$H(y) = (1 - w^T \Delta f_i(y))l(y_1, y)$$
using standard quadratic optimisation. This procedure has the effect of reducing the global optimisation into a series of simpler relaxed optimisations.

In order to implement a structured classifier using the formulation described, we need to be able to efficiently compute Equation (3.56). In the case of CRFs, for loss functions which decompose across cliques, we can use a version of the Viterbi algorithm (Section 3.3.1), modified to incorporate the loss scaling at each dynamic programming step.

It is also possible to use a maximum margin objective with the pseudo-likelihood approximation described in Section 3.4.3. In that case training the model is equivalent to the simple multi-class SVM described in Section 2.1, with the clique features folded into each states feature representation. Decoding such a model is performed identically to the maximum likelihood model with the Viterbi algorithm.
3.5 Discussion

Having described a number of approaches to structured sequence classification it is useful to compare these approaches to highlight their similarities and differences.

First we assume a CRF with the simple HMM-inspired feature decomposition shown in Figure 3.12. Using this feature decomposition we can view both the HMM and MEMM as instances of this CRF, where each model defines a different training regime for the CRF parameters. Once the parameters are set by training, all the models essentially share the same Viterbi decoding strategy when used for classification.

It is apparent that the features in Figure 3.12 mirror the emission and transition parameters of the HMM. The HMM makes independence assumptions about the model parameters, which enable their maximum likelihood estimate to be calculated analytically by simply counting occurrences in the training sample. In contrast, the CRF only assumes that the features can be decomposed into cliques, meaning that there is not an analytic solution for training the CRF and iterative algorithms must be used. On the other hand it allows the CRF a much greater freedom in the choice of features. The trained parameters for the HMM can be used in the linear classification framework, where the emission and transition log-probabilities can be viewed as the linear feature weights $w$. However, these parameters have the added benefit that they are actual probabilities, unlike the CRF parameters, thus they don’t need normalisation and can be used for additional modelling such as calculating the probability of an observation. This generative capability of the HMM models allow it to be used for language modelling applications and makes it suitable for unsupervised training.

From the discussion of pseudo-likelihood approximation for CRFs (Section 3.4.3), it is apparent that the MEMM can also be viewed as an approximation to full CRF training. While the pseudo-likelihood and MEMM training regimes are similar, both breaking the globally normalised CRF model into individual locally normalised models, the difference is that the MEMM only includes clique features from each states left neighbour, while the pseudo-likelihood model includes features from both neighbours. This imbalance in the MEMM is a result of its historical motivation to create a discriminative version of the directed graphs of the HMM. Working backwards from the undirected graphs of
CRFs, it is clear that there is little motivation for the leftward bias of the MEMM and the pseudo-likelihood model provides a more principled approximation to full CRF training. In Section 8.4 we compare the performance of pseudo-likelihood and MEMM models for the task of supertagging.

As a final note, all the models described in this section, and used subsequently in this thesis were implemented in C++, using MPI for distributed training on a cluster. Additionally, the maximum-margin models used the SVMSTRUCT\(^5\) library for optimising the quadratic program at the core of the parameter estimation algorithms.

\(^5\)http://svmlight.joachims.org/svm_struct.html
Part II

Word Alignment
Chapter 4

CRFs for Word Alignment

4.1 Introduction

In this chapter we explore the application of conditional random fields (CRFs) to the task of word-alignment for statistical machine translation (SMT).

Research in machine translation (MT) aims to develop systems which can automate, or assist, the translation of spoken or written documents from one language into another. With the dramatic increase in availability of multilingual information sources produced by the web, such technologies are of great interest to those faced with the task of finding information in documents in a foreign language. Potential users of machine translation range from governmental intelligence organisations that receive a large number of foreign documents and are unable to find enough human translators, to individuals searching for information on the world wide web written in languages other than their own.

SMT aims to automatically induce MT systems by extracting translation statistics from parallel corpora. Such corpora contain semantically similar pairs of sentences in both the source and target languages. Typically, SMT systems attempt to recognise common translation patterns, whether they be simple source/target ngram translation pairs (e.g. phrase based SMT) or more structured rules (e.g. syntax based SMT). Other approaches to MT include hand defining translation rules, often according to particular theories of grammar and semantics (e.g. transfer based MT), or extracting example translations from corpora (e.g. example based MT). Although historically separate, example based MT
Figure 4.1. Word alignment with a CRF: each word in the English source sentence is assigned an index into the target French sentence. The indexes assigned in this example imply the alignment displayed on the right.

(EBMT) and phrase based SMT exhibit a strong similarity. EBMT has tended to take a more direct approach, looking for analogous sub-translations of a required translation in a parallel corpus, where SMT makes use of a variety of statistics derived from the corpus, plus additional sources such as statistical language models.

Almost all modern SMT systems require a parallel corpus in which the words of each sentence pair are aligned. Such alignments loosely indicate which words, or ngrams, are translations of each other. From a word-aligned parallel corpus it is possible to create a translation table which is used by a decoder to map a foreign sentence \( f \) to its English translation \( e \).\(^1\) In the case of phrase based SMT, this translation table contains source/target ngram pairs, while for syntax based SMT it might contain, possibly incomplete, syntactic tree fragment pairs. Thus alignment of words within sentence-aligned parallel corpora is a core task in SMT systems.

Most current SMT systems (Och and Ney, 2004; Koehn et al., 2003) use a generative model for word-alignment such as the freely available GIZA++ (Och and Ney, 2003), an implementation of the IBM alignment models (Brown et al., 1993). These models treat word-alignment as a hidden process, and maximise the probability of the observed \((e, f)\) sentence pairs using the expectation maximisation (EM) algorithm (see Section 4.3 for a

\(^1\) We adopt the standard notation of \( e \) and \( f \) to denote the target (English) and source (foreign) sentences, respectively.
detailed description of these models). In this chapter we propose a discriminative word-alignment model based on CRFs (introduced in Section 3.2.2). We treat word-alignment as a sequence tagging structured classification task in which we tag each source word with the index of the target word to which it aligns (or a null tag for no alignment). This configuration is displayed in Figure 4.1. By training two such models, one for each source/target translation direction, and combining the output, we are able to achieve state-of-the-art alignment results. By using CRFs we are able to easily use many feature functions which are much more difficult to use in a generative model, and even incorporate features derived from the generative models. For instance, as well as detecting that a source word is aligned to a given target word, we would also like to encode syntactic and lexical features of the word pair, such as their parts-of-speech, affixes, lemmas, etc. Features such as these would allow for more effective use of sparse data and result in a model which is more robust in the presence of unseen words. Adding these non-independent features to a generative model requires that the features’ inter-dependence be modelled explicitly, which often complicates the model (e.g. Toutanova et al., 2002).

In this chapter we first give a brief overview of SMT research, from the word based IBM models to their phrase based successors (Section 4.2). We conclude this section with some comments on recent syntax based SMT research. In Section 4.3 we describe previous and current approaches to word-alignment, including the popular GIZA++ generative system and the most recent discriminative models. This section also includes the definition of a number of evaluation metrics used to assess the quality of word-alignments. We describe our experimental setup in Section 4.4, including the parallel corpus we have used. The structure of the CRF alignment model, and the feature functions employed, is described in Section 4.5. The proposed feature functions are systematically assessed on a development corpus in order to distill a subset of the most effective features for evaluation on the test set. The test set alignment results are presented in Section 4.6 and an analysis of these results and the efficacy of the feature functions chosen is contained in Section 4.7. Finally, we summarise and conclude the chapter in Section 4.9.
4.2 Statistical Machine Translation

The field of statistical machine translation (SMT) aims to develop systems that can extract translation models from large sentence-aligned parallel corpora and apply these models to translate novel sentence from one language to another. Given a foreign source sentence \( f = f_1 \ldots f_J \) we can formulate the SMT task as the maximisation:

\[
\hat{e} = \arg \max_e P(e|f)
\]  

(4.1)

where \( e = e_1 \ldots e_I \) is the target translation. Note that although this equation is most commonly stated as maximising the conditional probability of the English sentence given the foreign, there is no inherent need to model the translation probabilistically. The translation task can be viewed more generally as another instance of structured classification (Equation (3.1)), where we seek to maximise some translation scoring function.

In this section we give an overview of SMT and highlight the importance of the word-alignment task to the models discussed.

**IBM models** The work of Brown et al., 1990 forms the foundation of most modern work in SMT and introduced what have become known as the IBM Models. Their insight was to apply the noisy channel model, which had previously proved effective in the field of speech recognition, to machine translation. In this model we re-write Equation (4.1) using Bayes’ rule:

\[
\hat{e} = \arg \max_e P(e) \times P(f|e)
\]  

(4.2)

allowing the introduction of a target language model \( P(e) \), and having the effect of reversing the translation probabilities \( P(f|e) \). The use of a target language model is particularly an advantage when translating into languages for which large text corpora are available, such as English.

In order to tractably estimate the parameters of the language and translation models, they need to be decomposed by making a number of independence assumptions. Language modelling (LM) had previously been studied extensively in fields such as speech recognition (Stolcke, 2002). A number of LM systems, most based on ngrams which
The IBM translation models decompose the translation process into a number of steps which constitute a ‘generative story’ of how an observed source sentence is generated from an unobserved target sentence. The principle assumption made by these models is that words are translated independently of their context. The steps used in this decomposition are: (1) word duplication (fertility), (2) word insertion, (3) word translation, and (4) word reordering. The differences between the five IBM models lie in the way the probability distributions for these steps are conditioned. Figure 4.2 shows the decomposition of the translation process used in IBM Model 3.

As part of the decomposition used by these models the concept of a word-alignment is introduced as a hidden variable in the translation probability:

\[
\hat{e} = \arg \max_e P(e) \times \sum_a P(f, a|e) \quad (4.3)
\]

\[
\approx \arg \max_e P(e) \times \max_a P(f, a|e) \quad (4.4)
\]

where \(a_j = i\) is a mapping from source to target words and the so-called maximum approximation is used such that the sum over all alignments is approximated by its mode.

---

(1) words are first duplicated to allow for one-to-many translations, (2) null symbols are inserted that allow for the generation of words that aren’t direct translations from the target, (3) each word is then translated, (4) finally the resulting sequence is reordered to form the source sentence.

http://www.speech.sri.com/projects/srilm/
Therefore, word-alignments are used in the IBM models as a way of decomposing the translation process, rather than an end in themselves. Figure 4.3 shows a simple alignment example for a French-English sentence pair. This example is typical of alignments for closely related languages in that the alignments closely follow the diagonal.

The decoding problem of finding the English target sentence which maximises Equation (4.2) has been shown to be NP-complete for the IBM models (Knight, 1999). Therefore, in order to decode source sentences it is necessary to resort to heuristic approximations or greedy hill climbing based techniques (Germann, 2003).

Although the IBM models are obsolete as translation models, they have become the principle method for inducing word-alignments for other translation models and in Section 4.3.2 we describe the alignment component of these models in more detail.

**Phrase based translation** The main limitation of the IBM models is that they treat word translations as independent and do not allow multiple source words to translate into multiple target words. Och, 2002 presented an alternate approach to translating with the noisy channel model which uses a phrase translation and reordering model instead of IBMs word based model. In this model the units of translation are contiguous spans of words (or word classes) from the source and target sentences, called phrases or alignment templates.
Figure 4.4. The phrase based translation process: (1) words are grouped into phrases, (2) the phrases are reordered, (3) each phrase generates a foreign string. The arrow directions indicate that the English output sentence is modelled as generating the foreign input sentence.

These phrases do not necessarily correspond to constituents in any syntactic theory and are simply groups of words that occurred together. The generative translation process first segments the target sentence into phrases, reorders these phrases, and then translates them into source phrases. This process is illustrated in Figure 4.4.

A key step in phrase based translation systems is the acquisition of a phrase translation probability table. The alignment template model, as implemented by Och, uses heuristics to extract phrase translation pairs from a word-aligned corpus. The phrase extraction heuristic is illustrated in Figure 4.5. This heuristic extracts all phrases whose words are either not aligned, or aligned with only other words in the same phrase. The phrase translation probabilities are then calculated using a maximum likelihood estimation. Thus we can expect that the quality of the phrase translation table would be directly dependent on the accuracy of the word-aligned corpus it was extracted from.

The phrase based approach to SMT is currently the most popular translation method and has dominated public evaluations. Part of the popularity of this approach is the public availability of high quality translation decoders such as Pharaoh.

Marcu and Wong, 2002 proposed an alternate phrase translation model in which phrase-alignments are directly induced by the model, in much the same manner as the IBM word  

[^3]: http://www.nist.gov/speech/tests/mt/  
[^4]: http://www.isi.edu/publications/licensed-sw/pharaoh/
Figure 4.5. Translation phrases extracted from a word-aligned sentence pair according to the Pharoah phrase extraction heuristic.

based models. However, the complexity of aligning phrases makes it difficult to scale this direct approach to large training corpora.

Syntax based translation Phrase based SMT systems place no restriction on the phrases used in translation corresponding to syntactic constituents. Koehn et al., 2003 found that restricting a phrase based SMT system to only using linguistically-motivated phrases degrades translation performance. However, researchers continue to investigate whether syntax can improve SMT. In particular, by putting syntax at the core of the translation process the grammatical acceptability of the output should improve, something that phrase translation systems currently have difficulty with. Another advantage of this approach is that by basing translations on the syntactic structure of the source and or target sentences it is possible to reduce the decoding search space of possible re-orderings.
Wu, 1996 proposed a model that uses an Inversion Transduction Grammar to build synchronous parse trees of the source and target sentences. This model allows limited word re-orderings restricted to switching the order of children at binary non-terminal nodes. Yamada and Knight, 2002 employed syntax only on the target side, treating translation as the process of parsing a source sentence into the syntactic parse tree of the target sentence. He also included some syntactic phrase translations in his model and trained on a large sentence-aligned corpus using an EM-like algorithm and a cluster implementation.

The one sided syntactic approach to SMT has been extended recently to use larger translation units, analogous to the move from word to phrase based SMT systems. Galley et al., 2004 presents an approach for extracting translation rules from a word-aligned corpus which is parsed on the target side. These rules have been used for translation in Galley et al., 2006 and Marcu et al., 2006 with promising results. These systems currently represent the state-of-the-art for syntactic SMT.

An alternate syntactic SMT approach is presented by Chiang, 2005. He uses a synchronous parsing translation model that avoids any particular grammar formalism and attempts to learn weights for combining phrases in a hierarchical fashion. The hierarchical phrase translation rules are extracted from a word-aligned corpus with an algorithm which is an extension of the phrase extraction methods used in standard phrase based translation systems.

### 4.3 Word Alignment

Word-alignment is the task of finding a many-to-many mapping from the words of a source translation sentence to a target sentence:

\[ a \subseteq \{(j, i) : 0 \leq j \leq |f|, 0 \leq i \leq |e|\} \]  \hspace{1cm} (4.5)

Source or target words that don’t map to any other word are referred to as unaligned, or null aligned.

All of the current state-of-the-art approaches to SMT (Och, 2002; Koehn et al., 2003; Marcu et al., 2006; Chiang, 2005) rely on an automatically word-aligned corpus in order to extract a phrase translation table or translation rules. It is not only the SMT approach
to MT that can make use of word-alignments, EBMT systems such as Brown, 1997 also employ them. Besides MT, there are many other applications for word alignment, including bilingual lexicon extraction (Melamed, 2000), word sense disambiguation (Diab, 2000), and training part-of-speech taggers via multilingual projection (Yarowsky and Ngai, 2001; Drabek and Yarowsky, 2005).

In this section we review the many and varied approaches to automatic word-alignment, from heuristics to machine learning, and highlight the limitations and issues present in these models.

### 4.3.1 Heuristics

One of the simplest methods of computing word-alignments for parallel texts is to define a similarity function between source and target words and then use a heuristic to choose a suitable set of high scoring alignments. The most popular method of defining similarity is to use co-occurrence statistics over the parallel corpus to calculate an association score for all word pairs. One such association score that has been used is the Dice coefficient (Dice, 1945; Ker and Chang, 1997):

$$\text{Dice}(e, f) = \frac{2 \times C_{EF}(e, f)}{C_E(e) + C_F(f)}$$

where $C_X(y)$ denotes the count of the word $y$ in sentences of language $X$, and $C_{EF}(e, f)$ denotes the count of co-occurrences for the target word $f$ and the source word $e$. Once the association scores have been calculated for all word pairings in an aligned sentence pair we can apply a heuristic to choose an alignment. A simple heuristic is to pick the alignment with the highest association score for each source word:

$$a_j = \arg \max_i \{\text{Dice}(e_i, f_j)\}$$ (4.6)

A more sophisticated approach is to allow competition amongst source words for particular alignments. Melamed, 2000 defined a competitive linking algorithm which creates one-to-one alignments by iteratively choosing the alignment link with the highest score, then removing the words in this alignment from consideration. This algorithm terminates when either all words are aligned or the remaining association scores fall below some threshold.
Although these heuristic methods provide a simple and quick method of alignment, the association measures chosen lack sophistication, are often limited to predicting one-to-one alignments, and are not necessarily motivated by any probabilistic interpretation. Such heuristic methods were shown by Och and Ney, 2003 to be inferior to more sophisticated methods such as the IBM models. Although it would be possible to define more complex heuristics that addressed some of these shortcomings, the complexity would grow quickly, making machine learning approaches which seek to learn the model parameters more appealing.

4.3.2 The IBM models for alignment: GIZA++

As mentioned in Section 4.2 the IBM models can be used purely to align words as they incorporate a hidden variable over word-alignments. GIZA++ is a popular publicly available implementation of these models for word-alignment, plus an additional hidden Markov model (HMM) that wasn’t part of the original IBM formulation (Vogel et al., 1996).

The alignment model can be expressed as a HMM (Section 3.1.1) by making a first order Markov assumption (Figure 4.6). This assumption states that the probability of an alignment index depends only on the previous alignment index, and word translations are independent of context:

$$P(f, a|e) = p(J|I) \times \prod_{j=1}^{J} (p(a_j|a_{j-1}, I) \times p(f_j|e_{a_j}))$$ (4.7)

where $J$ and $I$ are the lengths of the source and target sentences respectively. This HMM implies that first we generate $J$ source word positions, given the target sentence ($p(J|I)$), then each source position generates an alignment index into the target sequence ($p(a_j|a_{j-1}, I)$) and the target word indexed generates a source word ($p(f_j|e_{a_j})$).

If we assume that alignment indexes are independent (a zero-order Markov assumption) and have a uniform distribution $p(a_j|a_{j-1}, I) = 1/(I + 1)$ we obtain Model 1:

$$P(f, a|e) = \frac{p(J|I)}{(I + 1)^J} \times \prod_{j=1}^{J} p(f_j|e_{a_j})$$ (4.8)

---

5 http://www.fjoch.com/GIZA++.html
Figure 4.6. The generative HMM alignment process: first \( J \) alignment positions are generated given the length of the target sentence \( I \), then each position generates a source index given the index of the previous position, and finally the English word indexed in each alignment position generates a source word.

However, alignments have a tendency to exhibit locality: words in the source sentence more often align with words with a similar index in the target sentence. In contrast to Model 1, Model 2 assumes each alignment index depends on the indexes of the words being aligned \( p(a_j | a_{j-1}, I) = p(a_j | j, I) \). This allows the model to learn a preference for aligning words in similar positions in the source and target sentences:

\[
P(f, a | e) = p(J|I) \times \prod_{j=1}^{J} \left( p(a_j | j, I, J) \times p(f_j | e_{a_j}) \right) \tag{4.9}
\]

Using dynamic programming algorithms it is possible to find maximum probability alignments from Models 1&2 and the HMM, allowing tractable training and decoding. This is an attractive property as it gives us some assurance that the probability distribution estimated will be useful, and that the decoded alignments are optimal with respect to the model. This contrasts to the later models which must employ significant amounts of heuristic approximation in order to perform training and decoding.

A limitation of Models 1 and 2 is that they do not explicitly model whether target words have a tendency to generate more than one source word. Models 3 and 4 introduce
the concept of word fertility \( \phi \), where the fertility of a word \( e_i \) is defined as the number of source words to which it aligns:

\[
\phi_i = \sum_j \delta(a_h, i)
\]  

(4.10)

where \( \delta \) is the Kronecker delta function which evaluates to one if its arguments are equal, and zero otherwise. In Figure 4.2 we see that \( \text{slap} \) translates into \( \text{daba una bofetada} \), thus in this example \( \text{slap} \) has fertility \( \phi_4 = 3 \). The fertility based models reverse the alignment direction, introducing the hidden inverted-alignment variable \( B \) which defines a mapping from target to source. Each target word may be aligned to several adjacent source words and every source word must be aligned to a single target, or \textbf{null} \( (B_0) \):

\[
B : i \rightarrow B_i \subset \{1, \ldots, j, \ldots, J\}
\]  

(4.11)

If we assume that a target word depends only on the English word that generated it, these models have the rather complex form:

\[
P(f, B|e) = P(B_0|B_1^i) \times \prod_{i=1}^{I} \prod_{k=1}^{B_i} p(f_{B_{ik}}|e_i) \times P(B_{ik}|B_{i-1}^i, e_{i, B_{i-1}^i}, f_{B_{i-1}^i}^{B_{i-1}^i}, \cdots)
\]  

(4.12)

where the \textbf{null} \( (B_0) \) alignments are generated last. The difference in how the alignment probabilities \( P(B_{ik}|\ldots) \) are estimated differentiates the fertility based models. Model 3 makes a zero-order Markov assumption, i.e. alignment sets \( B_i \) are generated independently of each other. Model 4 includes a first-order Markov dependency in which the probability of an alignment set \( B_i \) is dependent on the last previous non-empty alignment set. Models 3 and 4 are referred to as deficient because the probabilities of all valid alignments do not sum to one. A non-deficient fifth model was proposed, but as it has considerably more parameters than Model 4 and, as deficiency has not proved to be a problem in practice, it has not gained popularity.

The IBM models are trained using a large sentence-aligned corpus and the expectation maximisation (EM) algorithm. For the HMM and models 1&2 it is possible to perform exact training using dynamic programming algorithms. For models 3&4 it is not possible to calculate the normalising partition function necessary for calculating the expectations.
so a heuristic optimisation approach is used. The fertility models approximate the Viterbi alignment by first calculating the Viterbi alignment of a simpler model, such as the HMM, and then performing hill-climbing with respect to the fertility model by permuting this initial alignment. As the EM algorithm only finds a local maximum, there is no guarantee that the parameters found in training are optimal. In addition, the number of EM iterations must in practice be chosen carefully to avoid poor solutions. For a complete mathematical description of the IBM models as they are implemented in GIZA++ see Och and Ney, 2003.

As mentioned earlier, many current systems use the GIZA++ implementation of the IBM alignment models. While GIZA++ gives good results when trained on large sentence-aligned corpora, its generative models have a number of limitations. Firstly, they impose strong independence assumptions between features, making it very difficult to incorporate non-independent features over the sentence pairs. For instance, as well as detecting that a source word is aligned to a given target word, we would also like to encode syntactic and lexical features of the word pair, such as their parts-of-speech, affixes, lemmas, etc. Features such as these would allow for more effective use of sparse data and result in a model which is more robust in the presence of unseen words. Adding these non-independent features to a generative model requires that the features’ inter-dependence be modelled explicitly, which often complicates the model (e.g. Toutanova et al., 2002). Secondly, the later IBM models, such as Model 4, have to resort to heuristic search techniques to approximate forward-backward and Viterbi inference. These heuristics sacrifice optimality for tractability, but still require considerable time to train, often taking many days. Finally, these models require very large parallel corpora in order to achieve reasonable performance, which is an unreasonable assumption for many language pairs.

4.3.3 Discriminative approaches

A particular limitation of generative models for word-alignment is the difficulty of incorporating features. As mentioned in Section 4.3.2, there are many examples of compelling features we may wish to include in an alignment model. Toutanova et al., 2002 showed that significant mathematical effort is required in order to incorporate just PoS tags into the hidden Markov alignment model. They introduced additional lexicon probabilities
for PoS tags and then extended the generative story of the model in order to incorporate this additional source. However this becomes more difficult with each additional information source. Recently, a number of discriminative word-alignment models have been proposed which are motivated by their ability to model such features; these models have demonstrated performance superior to generative models.

A standard form for such discriminative models is to search for the alignment, $\hat{a}$, that maximises a weighted ($w$) linear combination of real-valued feature functions ($h(\cdot)$):

$$\hat{a} = \arg \max_a \sum_{k=0}^{K} w_k h_k(a, e, f) \quad (4.13)$$

This is an example of structured linear classification as outlined in Section 3. The current models are differentiated by two factors:

1. whether the $\arg \max$ search is decomposed via dynamic programming, and the manner in which it is decomposed;

2. the optimisation method employed to find the weights $w$, the current methods being log-linear, large-margin or heuristic.

Liu et al., 2005 used a conditional log-linear model with similar features to those we employ in this thesis. They formulated a global model, without making a Markovian assumption, which meant that it was not possible to decompose the calculation of the partition function via a dynamic programming algorithm. Instead a heuristic beam search was employed for finding an N-best list of alignments. Their search employs the algorithm shown in Figure 1. Using this search algorithm the authors computed the partition function by creating an N-best list for a sentence, updating the model weights using iterative scaling (Darroch and Ratcliff, 1972), then iterating this procedure by adding the N-best lists together until there is no change in the list. As the space of possible alignments searched by this algorithm is huge, it is unlikely that the beam search will find the optimal alignment, or that the parameter optimisation strategy will accurately locate the correct conditional distribution. In contrast, the model we present here decomposes the alignments into cliques which enables us to use a dynamic programming algorithm which efficiently finds the optimal alignment, and calculates the partition function exactly.
Algorithm 1: Alignment search algorithm used by Liu et al., 2005

1: \( a = \emptyset \) \hspace{1cm} \triangleright \text{Start with the empty alignment}
2: loop
3: \( \text{max_gain} \leftarrow 0 \)
4: for alignment link \( l = (i, j) \notin a \) do
5: \( \triangleright \text{gain} \) is a measure of the improvement in the alignment by adding the link \( l \)
6: if \( \text{max_gain} < \text{gain}(a, l) \) then
7: \( \text{max_gain} \leftarrow \text{gain}(a, l) \)
8: end if
9: end for
10: if \( \text{max_gain} \leq \text{threshold} \) then
11: return
12: end if
13: \( a \leftarrow a \cup \text{max_gain} \) \hspace{1cm} \triangleright \text{Add the alignment link with the highest gain}
14: end loop

A discriminative word-alignment system trained on a substantial number of word-aligned sentences was presented by Ittycheriah and Roukos, 2005. This work was able to use a training corpus of ten thousand word-aligned Arabic-English sentences, and thus a fairer comparison with a generative model trained on hundreds of thousands of sentence-alignments was possible. Their discriminative model was able to significantly outperform the \textsc{Giza++} baseline. As with other approaches, they proposed a model which didn’t allow a tractably optimal solution and thus had to resort to a heuristic beam search. However, they didn’t use the form in Equation (4.13), instead employing a log-linear model to learn ‘observation’ probabilities, while using a fixed ‘transition’ distribution:

\[
p(a|e, f) = \frac{1}{Z} \left\{ \prod_{i=0}^{I} p(a_i|a_{i-1})^{\frac{1}{2}} \times p(a_i|e_i, f_{i}^1, a_{i-1})^{\frac{1}{2}} \right\} \quad (4.14)
\]

The transition distribution is fixed to:

\[
p(a_i|a_{i-1}) = \frac{1}{Z(a_{i-1})} \left\{ \frac{1}{\text{dist}(a_i, a_{i-1})} + \frac{1}{\text{ns}(a_i)} \right\} \quad (4.15)
\]
where $dist$ is the integer difference between the current alignment index and the previous rightmost alignment index assigned, and $ns$ is a fertility penalty for many-to-one alignments. Both of these quantities require non-local alignment information, thus disallowing any efficient dynamic programming solution. A log-linear model is used to model the ‘observation’ distribution:

$$p(a_i | e_i, f_i, a_{i-1}) = \frac{1}{Z(e_i, f_i, a_{i-1})} \exp \sum_k w_k h_k(f_a, e_i, f_i, a_{i-1})$$

(4.16)

where the feature functions $h(\cdot)$ are dependent on both sentences, and the previous and current alignment decisions, again including non-local information. It is not exactly clear what motivates this model decomposition as both the transition and observation distribution allow for global feature dependencies. In addition they appear to overlap by both considering the sequence of alignment predictions. In order to train their model the authors used a beam search which predicts in sequence an Arabic alignment for each English word. As non-local features are used, the search algorithm must maintain the whole of each hypothesis within the beam. Therefore, for any sentence of significant length, the beam width required would be extremely large in order to have any chance of exploring the state space of feature configurations. The authors do not state the size of the beam used, beyond saying that it was ‘large’. In comparison, our proposed CRF model (Section 4.5.2) allows both the observation and transition components of the model to be jointly optimised from the corpus, and uses only local features which allows the optimisation problem to be solved exactly.

Taskar et al., 2005 presented a word matching model for discriminative alignment which they formulated as a min-flow problem and were able to solve optimally using standard combinatorial algorithms. Their original model was limited to only providing one-to-one alignments and didn’t include features on label sequences, although the recent extension by Lacoste-Julien et al., 2006 includes limited many-to-many (2x2) alignments and sequence features. The maximum-margin formulation of the model is expressed as an integer linear program (ILP), but can also be solved with combinatorial techniques when viewed as a min-cost flow problem (Figure 4.7). Like our approach, this model requires significantly less heuristics than the other models presented in this section and its optimal inference algorithms avoid the use of beam search. This model reports an alignment error.
Figure 4.7. Maximum weight bipartite matching min-flow graph. Alignment is modelled as finding the edge configuration which minimises the flow cost from the source to the sink, where the cost of an edge is proportional to the product of the model weights and features active for that edge.

rate (Section 4.3.4) of 3.8, the lowest to date on the French-English Canadian Hansards corpus (Section 4.4) when using other alignment models, IBM Models 2&4 and the model of Liang et al., 2006, as features. Our use of a CRF sequence model allows the natural modelling of one-to-many alignments and sequences without the need for a complex and costly ILP solver. In addition, we can choose to optimise our CRF according to either a probabilistic MAP, or maximum-margin, objective function. The word matching model cannot be easily optimised probabilistically due to the difficulty of calculating the normalising partition function.

Moore, 2005 implemented another model that attempted to directly optimise Equation (4.13) without making independence assumptions. This work was then extended by
Moore et al., 2006 with the addition of more features and refinements to the training algorithm. They explored two association measures estimated from a large sentence-aligned corpus, the first being a log-likelihood ratio statistic:

\[
llr(f, e) = \sum_{x \in \{f, \neg f\}} \sum_{y \in \{f, \neg f\}} C(x, y) \log \frac{p(x|y)}{p(x)}
\]

where \(p(\cdot)\) is a maximum likelihood estimate. The second measure is called conditional link probability which is derived from word-alignments predicted by a simpler model:

\[
lp_d(f, e) = \frac{\text{links}_1(f, e) - d}{C(f, e)}
\]

where \(\text{links}_1(\cdot, \cdot)\) is the number of times the two words were aligned by a simpler model, and \(d\) is a fixed discount. The beam search algorithm they employed was similar to that described in Algorithm 1, however significantly more sophisticated heuristics were used to augment, and permute, the alignments in the beam. Using this search algorithm the model weights were trained using a variant of the averaged perceptron algorithm (Collins and Duffy, 2002). In order to improve the stability of the perceptron algorithm with the heuristic search a large number of modifications and tuning parameters were required. This model eventually equalled the performance of Lacoste-Julien et al., 2006 on the Canadian Hansards data set.

### 4.3.4 Evaluation measures

Many measures of word-alignment performance have been proposed and there is little agreement as to which is the most pertinent. Here we describe the standard measures of precision, recall and f-score. We also include the popular alignment error rate (AER) measure and the recently proposed consistent phrase error rate (CPER) (Ayan and Dorr, 2006).

**Precision, recall and f-score** A number of word-aligned parallel corpora don’t simply mark whether two words are aligned, but differentiate between strong and weak alignments. The most common annotation scheme is to mark the alignments as either sure (\(S\)) or possible (\(P\)) (\(S \subseteq P\); (Och and Ney, 2003)), where the possible alignments indicate
they are constrained by limits which are imposed in order to ensure that the freedom of one person does not violate that of another.

Figure 4.8. A word-aligned example from the Canadian Hansards French-English corpus. Hollow squares represent gold standard sure alignments, circles are gold possible alignments, and filled squares are predictions output from an automatic alignment system.

ambiguous or idiomatic alignments. Using this annotation scheme, we can define the following performance metrics:

\[
\text{recall} = \frac{|A \cap S|}{|S|} \quad \text{precision} = \frac{|A \cap P|}{|A|}
\]

In addition, we can use the standard definition of the f-score as the harmonic mean of the precision and recall:

\[
f_1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}
\]

Figure 4.8 shows an example French-English alignment where both sure and possibles are marked, as well as predictions made by an automatic alignment system. Evaluating this example we see that: \(|A \cap S| = 19, |A \cap P| = 20, |S| = 20, |P| = 36, \text{ and } |A| = 20\). Therefore, recall = 0.95, precision = 1.0, and \(f_1 = 0.97\)
Alignment error rate  Most previous works have used alignment error rate (AER), as the primary measure of model performance:

\[ AER(a, s, p) = 100 \times \left(1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|}\right) \]

where \( A \) is the set of predicted alignments. Returning to Figure 4.8 we can calculate the AER as: \( 100 \times \left(1 - \frac{19+20}{20+20}\right) = 2.5 \). Note that the possible alignments missed by the automatic predictions are not heavily penalised by AER. Thus, when using this metric, possibles are treated as bonus points.

When a data set is only annotated with sure alignments the AER reduces to one minus the f-score. When possibles are annotated it is often the case that they are predominately used to indicate alignments with fertility greater than one, which means the AER metric has a tendency to over-emphasise one-to-one alignments. This property has lead researchers to question the relevance of AER and recent work has found that f-score correlates with end-to-end translation performance significantly better than AER (Fraser and Marcu, 2006a). As a result, in this chapter we emphasise f-score results when tuning our models.

Consistent phrase error rate  Ayan and Dorr, 2006 introduced the CPER measure in an attempt to counter some of the limitations of AER. CPER evaluates the translations that would be extracted from a predicted alignment versus those that would be extracted from a reference alignment, thus attempting to directly evaluate alignments in respect to the purpose that they are used in phrase-based translation systems. If \( P_A \) and \( P_G \) are the sets of phrases extracted from the predicted and reference alignments respectively, the precision and recall are defined as:

\[
\begin{align*}
P_{\text{recall}} &= \frac{|P_A \cap P_G|}{|P_G|} \\
P_{\text{precision}} &= \frac{|P_A \cap P_G|}{|P_A|}
\end{align*}
\]

and the CPER is defined as one minus the extracted phrase f-score:

\[ CPER = 1 - \frac{2 \times P_{\text{precision}} \times P_{\text{recall}}}{P_{\text{precision}} + P_{\text{recall}}} \]

Figure 4.9 shows how the precision of the alignments predicted by a model can affect the phrases that are extracted by a phrase-based translation system such as Pharaoh. By making the incorrect alignment \( \text{the} \leftrightarrow \text{de} \) five phrases would be missed by the phrase extraction
Figure 4.9. The effect of alignment errors on phrases extracted by a translation system. The phrases with a broken outline would not be extracted if the erroneous grey alignment point was predicted.

heuristic. Such a mistake would be penalised much more heavily using CPER than the other metrics discussed here.

4.4 Experimental Setup

In order to train and evaluate our discriminative word-alignment model we need a corpus of both sentence-aligned, and word-aligned sentences. As discriminative approaches to alignment are relatively new there are not a great number of word-aligned parallel corpora, and those that do exist are quite small. We have chosen to evaluate our model on a corpus made available from a NAACL shared task: the French-English Hansards data set. Later in this chapter we present additional results using other language pairs (Section 4.8).
4.4.1 Data

Canadian Hansards

Our main data set is the English-French Canadian Hansards corpus, which consists of 1.1 million aligned sentences and 484 word-aligned sentences. This corpus was compiled from transcripts (Hansards) of the proceedings of the 36th Canadian Parliament. It includes debates in both the House and the Senate, and has been tokenised and sentence aligned. We have a used version of this corpus made available for the 2003 NAACL shared task (Mihalcea and Pedersen, 2003), where the word-aligned sentences were split into a 37 sentence trial set and a 447 sentence testing set.

Unlike the unsupervised entrants in the 2003 task, we require word-aligned training data, and therefore must cannibalise the test set for this purpose. We follow Taskar et al., 2005 in using the first 100 test sentences for training and the remaining 347 for testing. This means that our results should not be directly compared to systems that participated in that shared task, other than in an approximate manner. We used the original 37 sentence trial set for feature engineering and for fitting a Gaussian prior.

The word-aligned data are annotated with both sure ($S$) and possible ($P$) alignments (Section 4.3.4). In the case of this corpus, it would seem that the $P$ alignments are most often used for alignments with fertility greater than 1, even when the alignment might be considered unambiguous (e.g. seront ↔ will be).

---

Table 4.1. Statistics for the parallel corpus.

<table>
<thead>
<tr>
<th></th>
<th>Canadian Hansards</th>
</tr>
</thead>
<tbody>
<tr>
<td>aligned sentences</td>
<td>1.1M</td>
</tr>
<tr>
<td>word-aligned sentences</td>
<td>484</td>
</tr>
<tr>
<td>sure alignment points</td>
<td>4376</td>
</tr>
<tr>
<td>possible alignment points</td>
<td>14846</td>
</tr>
<tr>
<td>one-to-one alignments</td>
<td>3959</td>
</tr>
<tr>
<td>one-to-one (sure-only)</td>
<td>4076</td>
</tr>
<tr>
<td>one-to-many alignment points</td>
<td>573</td>
</tr>
<tr>
<td>many-to-one alignment points</td>
<td>1075</td>
</tr>
<tr>
<td>many-to-many alignment points</td>
<td>13615</td>
</tr>
</tbody>
</table>

6http://www.isi.edu/natural-language/download/hansard/
4.5 A CRF for Word Alignment

In Section 4.3.3 we discussed the limitations of generative word-alignment models such as GIZA++, in particular their limited ability to use features to integrate varied information sources. In this section we present our alternative discriminative method for word-alignment. We propose modelling word-alignment with CRFs (Chapter 3) and demonstrate how the attributes of these models make them particularly suitable for this task.

CRFs are discriminative models that allow for the use of arbitrary and overlapping features of a structured input, and can be trained to estimate a conditional probability distribution over a structured output. For the word-alignment model the input to the CRF is the source and target sentences to be aligned, and the output is a sequence of alignment indexes from source to target. The sequence models we present here have optimal and tractable training and decoding algorithms, a significant advantage over previously proposed discriminative alignment models that require heuristics for training and decoding. Furthermore, the models allow regularisation, using either a prior over the parameters in the case of the conditional model, or the soft-margin in the maximum margin model. Both are effective and simple methods for limiting over-fitting.

We use a similar graphical structure to the directed hidden Markov model (HMM) from GIZA++ (Och and Ney, 2003). This models one-to-many alignments, where each target word is aligned with zero or more source words. Many-to-many alignments are recoverable using the standard heuristics for superimposing predicted alignments in both translation directions.

We demonstrate the effectiveness of the CRF word-alignment method on the English-French data described in Section 4.4, evaluated using precision, recall, f-score and AER (Section 4.3.4).

4.5.1 Conditional random fields for word-alignment

We use a CRF to model many-to-one word-alignments, where each source word is aligned with zero or one target words, and therefore each target word can be aligned with
many source words. Each source word is labelled with the index of its aligned target, or the special value null, denoting no alignment. In this way the alignment task can be viewed as tagging each source word with a target index. An example word-alignment is shown in Figure 4.8, where the hollow squares and circles indicate the correct sure and possible alignments respectively. In this example the French words une and autre would both be assigned the index 24 – for the English word another – when French is the source language. When the source language is English, another could be assigned either index 25 or 26; in these ambiguous situations we take the first index.

To train the model we have a choice between optimising either the likelihood of the training data, or the classification margin of the correct alignments over the incorrect alignments. In this chapter we use the probabilistic maximum likelihood trained CRF to evaluate features and tune our model, however we give final alignment results for the test set using both models. In Chapter 5 we investigate how the abilities of the maximum margin formulation can allow us to optimise specific loss functions, and the impact that these optimisations have on end-to-end translation performance.

Recapping Chapter 3, the joint probability density of the alignment, $a$ (a vector of target indices), conditioned on the source and target sentences, $e$ and $f$, is given by:

$$p_{w}(a|e, f) = \frac{\exp \sum_{t} \sum_{k} w_{k} h_{k}(t, a_{t-1}, a_{t}, e, f)}{Z_{w}(e, f)}$$

(4.19)

where we make a first order Markov assumption over the alignment sequence. Here $t$ ranges over the indices of the source sentence ($f$), $k$ ranges over the model’s features, and $w = \{ w_{k} \}$ are the model parameters (weights for their corresponding features). The feature functions $h_{k}$ are predefined real-valued functions over the source and target sentences coupled with the alignment labels over adjacent times (source sentence locations), $t$. These feature functions are unconstrained, and may represent overlapping and non-independent features of the data. Note that the alignment CRF is a function of two input sequences, $e$ and $f$, and that the label set differs with the length of each target sentence. These differences generalise the standard application of CRFs from labelling a single input sequence with a fixed label set.
The distribution is globally normalised by the partition function, $Z_w(e, f)$, which sums out the numerator in (4.19) for every possible alignment:

$$Z_w(e, f) = \sum_a \exp \left( \sum_t \sum_k w_k h_k(t, a_{t-1}, a_t, e, f) \right)$$

We use a linear chain CRF, which is encoded in the feature functions of Equation (4.19). Here the alignment, $a$, is a vector of indices into the target sentence, with an entry for each word of the source sentence.

In the probabilistic case (MAP-CRF), the parameters are estimated from a fully observed word-aligned training sample, $D = \{(a, e, f)\}$, by defining a prior distribution over the model parameters and deriving a maximum a posteriori (MAP) estimate: $w^{MAP} = \arg \max_w p_w(D)p(w)$.

Using a zero-mean Gaussian prior, with the probability density function $p_0(w_k) \propto \exp\left( -\frac{w_k^2}{2\sigma_k^2} \right)$, this yields a log-likelihood objective function of:

$$L^{MAP} = \sum_{(a, e, f) \in D} \log p_w(a|e, f) + \sum_k \log p_0(w_k)$$

$$= \sum_{(a, e, f) \in D} \sum_t \sum_k w_k h_k(t, a_{t-1}, a_t, e, f)$$

$$- \log Z_w(e, f) - \sum_k \frac{w_k^2}{2\sigma_k^2} + \text{const.} \quad (4.20)$$

As discussed in Chapter 3 we can optimise this convex function using standard numerical methods. For our model we use L-BFGS, an iterative quasi-Newton optimisation method, which performs well for training log-linear models (Malouf, 2002; Sha and Pereira, 2003). The expectations required by this method are calculated using forward-backward inference (Section 3.3.2), which yields the partition function, $Z_w(e, f)$, required for the log-likelihood, and the pairwise marginals, $p_w(a_{t-1}, a_t|e, f)$, required for its derivatives. The Viterbi algorithm (Section 3.3.1) is used to find the maximum posterior probability alignment for test sentences, $a^* = \arg \max_a p_w(a|e, f)$.

In the case of the maximum margin formulation (MM-CRF), we optimise the dual of Equation (3.53) from Chapter 3, using the clique wise feature functions $h_k(t, a_{t-1}, a_t, e, f)$
and the cutting plane algorithm of Tsochantaridis et al., 2005. For this algorithm the main issue is calculating the maximiser of the cost function \( \hat{y} = \arg \max_{y \in \mathcal{Y}} H(y) \), where:

\[
H(y) = l(y_1, y) - \mathbf{w}^T \Delta f_i(y)
\]

(4.21)

The intuition behind the use of this maximisation is that we need to be able to identify the incorrect label with the highest classification score in order to maximise the margin between it and the correct label. For our alignment CRF, with a Hamming loss (Equation (3.52)), this calculation decomposes across the cliques of the graph and can be efficiently calculated using the Viterbi algorithm with the following modification to the recursion step (Section 3.3.1):

\[
\delta_t(s) = \max_{s_{t-1} \in \mathcal{S}} \left( \delta_{t-1}(s_{t-1}) + l^{0 \rightarrow 1}(y_{it}, s) - \Delta \Psi_{it}(s_{t-1}, s) \right)
\]

\[
\Delta \Psi_{it}(s_{t-1}, s) = \Psi(s_{t-1}, y_{it}, \mathbf{x}, t) - \Psi(s_{t-1}, s, \mathbf{x}, t)
\]

where \( y_{it} \) is the alignment index at source index \( t \) in the gold standard label \( y_i \), for the \( i \)th training instance. We can decode test instances using the standard Viterbi algorithm, in the same manner as for the MAP-CRF.

Both the MAP-CRF and the MM-CRF require a dynamic programming step for processing each training instance in their optimisation loops. Therefore, the complexity of these models is dominated by the square of the number of target sentence indices. However, as the MM-CRF attempts to find a sparse weight vector, it often requires less evaluations for each training instance making it faster to converge in practice.

### 4.5.2 The alignment model

Our model takes a sequence of target and source words, and assigns each source word the index of a target word. This allows the modelling of many-to-one relationships where a target word may be aligned to more than one source word: for example, we would like to be able to align both the French words \( \text{ne} \) and \( \text{pas} \) with the same English \( \text{not} \). Such an alignment would be possible with French as the source language, but not with English as we could only assign a single target index for \( \text{not} \).

Before we can apply our CRF alignment model, we must first specify the feature set – the functions \( h_k \) in Equation (4.19). Typically CRFs use binary indicator functions as
features; these functions are only active when the observations meet some criteria and
the label \( a_t \) (or label pair, \((a_{t-1}, a_t)\)) matches a pre-specified label (pair). However, in our
model the labelings are word indices in the target sentence and cannot be compared readily
to labelings at other sites in the same sentence, or in other sentences with a different length.
Such naive features would only be active for one labelling, and therefore this model would
suffer from serious sparse data problems.

We instead define features which are functions of the source-target word match implied
by a labelling, rather than the labelling itself. For example, from the sentence in Figure 4.8
for the labelling of \( f_{24} = de \) with \( a_{24} = 16 \) (for \( e_{16} = of \)) we might detect the following
feature:

\[
h(t, a_{t-1}, a_t, f, e) = \begin{cases} 
1, & \text{if } e_{a_t} = '\text{of}' \land f_t = '\text{de}' \\
0, & \text{otherwise}
\end{cases}
\]

Note that it is the target word indexed by \( a_t \), rather than the index itself, which determines
whether the feature is active, and thus the sparsity of the index label set is not an issue.
In order to train our sequence model we must have gold standard alignment indexes for each source sentence word. As alignment data sets mark many-to-many alignments, which our model can’t directly represent, we must decide how to handle these alignments. In this work we have employed the simple heuristic of taking the first alignment index that is encountered for a source word, and ignoring subsequent indexes. A further complication is that some data sets (in our case the Hansards corpus) are annotated with both sure and possible alignments. As most many-to-many alignments are marked with possibles, our models ignore the possible alignments and only extract gold standard alignments from the sure alignments. We revisit this issue in Section 4.5.6 to investigate whether we can gain useful gold alignments from these possibles.

### 4.5.3 Symmetrisation

In order to produce many-to-many alignments we combine the outputs of two models, one for each translation direction. A number of heuristics for combining the output of directional many-to-one alignment models have been proposed. Here we use heuristics available in the Pharaoh SMT decoder.\(^7\)

The default heuristic in Pharaoh is the grow-diag-final algorithm which, starting from the intersection of the alignments predicted by the directional models, adds additional points from the union which are neighbours of the points in the intersection. The pseudo-code for this heuristic is displayed in Algorithm 2 (Koehn, 2004a). We present symmetrised results using this heuristic.

### 4.5.4 Features

One of the main advantages of using a discriminative model is the ability to explore a diverse range of features engineered for a specific task. In our CRF model we employ two main types of features: those defined on a candidate aligned pair of words; and Markov features defined on the alignment sequence predicted by the model. In this section we propose a range of feature types and evaluate their performance on the 37 sentence trial.

---

\(^7\)http://www.isi.edu/publications/licensed-sw/pharaoh/
Algorithm 2 The grow-diag-final heuristic used in the Pharaoh toolkit

1: neighbour ← ((−1, 0), (0, −1), (1, 0), (0, 1), (−1, −1), (−1, 1), (1, −1), (1, 1))
2: alignment ← eng2for ∩ for2eng
3: procedure GROW-DIAG-FINAL(eng2for, for2eng)
4:    GROW-DIAG(); FINAL(eng2for), FINAL(for2eng);
5: end procedure
6: procedure GROW-DIAG
    ▶ Grow from points in the intersection to their unaligned neighbours
7:    repeat
8:        for English word e ← 0 . . . en do
9:            for Foreign word f ← 0 . . . fn do
10:               if (e, f) ∈ alignment then
11:                   for each neighbouring point (ê, ˆf) do
12:                                   if ê /∈ alignment ∧ ˆf /∈ alignment ∧
13:                                       (ê, ˆf) ∈ (eng2for ∪ for2eng) then
14:                                           alignment = alignment ∪ (ê, ˆf)
15:                                   end if
16:                   end if
17:               end for
18:            end for
19:        end repeat
20:    end procedure
21: procedure FINAL(a)
    ▶ Add points from a if they haven’t already been aligned
22:        for English word ˆe = 0 . . . en do
23:            for Foreign word ˆf = 0 . . . fn do
24:               if (ˆe /∈ alignment ∨ ˆf /∈ alignment) ∧ (ê, ˆf) ∈ a then
25:                  alignment = alignment ∪ (ê, ˆf)
26:               end if
27:         end for
28:     end for
29: end procedure
### Table 4.2

Results comparing Dice and Model 1 features. Feature results in bold have a positive impact on f-score.

<table>
<thead>
<tr>
<th>feature type</th>
<th>Trial f-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dice</td>
<td>0.614</td>
</tr>
<tr>
<td>+CrossDice</td>
<td><strong>0.716</strong></td>
</tr>
<tr>
<td>+MaxDice</td>
<td><strong>0.628</strong></td>
</tr>
<tr>
<td>+NullMaxDice</td>
<td>0.552</td>
</tr>
<tr>
<td>+NullSumDice</td>
<td><strong>0.616</strong></td>
</tr>
<tr>
<td>+CrossDice +MaxDice</td>
<td><strong>0.692</strong></td>
</tr>
<tr>
<td>+NullSumDice</td>
<td></td>
</tr>
<tr>
<td>Model1</td>
<td>0.659</td>
</tr>
<tr>
<td>+CrossModel1</td>
<td><strong>0.705</strong></td>
</tr>
<tr>
<td>+MaxModel1</td>
<td><strong>0.710</strong></td>
</tr>
<tr>
<td>+NullMaxModel1</td>
<td><strong>0.659</strong></td>
</tr>
<tr>
<td>+NullSumModel1</td>
<td><strong>0.740</strong></td>
</tr>
<tr>
<td>+CrossModel1 +MaxModel1</td>
<td><strong>0.787</strong></td>
</tr>
<tr>
<td>+NullSumModel1 (Base)</td>
<td></td>
</tr>
<tr>
<td>+CrossModel1 +MaxModel1</td>
<td>0.765</td>
</tr>
<tr>
<td>+NullSumModel1 +CrossDice</td>
<td></td>
</tr>
<tr>
<td>+MaxDice +Null SumDice</td>
<td></td>
</tr>
</tbody>
</table>

set. We select the features which appear to give a positive contribution for this set and incorporate them into a complete reference model for evaluation on the test set.

**Dice and Model 1**

As we have access to only a small amount of word-aligned data, we wish to be able to incorporate information about word association from any sentence-aligned data available. As mention in Section 4.3, a common measure of word association is the Dice coefficient (Dice, 1945):

\[
\text{Dice}(e, f) = \frac{2 \times C_{EF}(e, f)}{C_E(e) + C_F(f)}
\]

where \(C_E\) and \(C_F\) are counts of the occurrences of the words \(e\) and \(f\) in the corpus, while \(C_{EF}\) is their co-occurrence count. We treat these Dice values as translation scores: a high (low) value indicates that the word pair is a good (poor) candidate translation.

However, the Dice score often over-estimates the association between common words. For instance, the words *the* and *of* both score highly when combined with either *le* or...
de, simply because these common words frequently co-occur. The GIZA++ models can be used to provide better translation scores, as they enforce competition for alignment between the words. For this reason, we investigate the translation probability distribution from Model 1 in addition to the Dice scores. Model 1 is a simple position independent model which can be trained quickly and is often used to bootstrap parameters for more complex models. It models the conditional probability distribution:

\[
p(f, a | e) = p(|f||e|) \times \prod_{t=1}^{|f|} p(f_t | e_a_t)
\]

where \(p(f | e)\) are the word translation probabilities.

Figure 4.2 displays the results of evaluating a variety of Dice and Model 1 based features on the trial set. We use both the Dice value and the Model 1 translation probability as real-valued features for each candidate pair \(\text{Dice, Model1}\), as well as a normalised score over all possible candidate alignments for each target word \(\text{CrossDice, CrossModel1}\). We derive a feature from both the Dice and Model 1 translation scores to allow competition between source words for a particular target alignment \(\text{MaxDice, MaxModel1}\). This feature indicates whether a given alignment has the highest translation score of all the candidate alignments for a given target word. For the example in Figure 4.8, the words la, de and une all receive a high translation score when paired with the. To discourage all of these French words from aligning with the, the best of these (la) is flagged as the best candidate. This allows for competition between source words which would otherwise not occur.

We also used these translation scores to develop features for null alignments. We use the maximum score for a particular source word in order to indicate whether a strong match alignment is present and dissuade a null alignment in this case \(\text{NullMaxDice, NullMaxModel1}\). We also use a sum of the translation scores for each source word to indicate how much probability mass was present \(\text{NullDiceSum, NullModel1Sum}\). In the case of Model 1 probabilities we can interpret this sum as the probability that a source word is aligned to any of the target words. Figure 4.11 shows the effect of adding NullM1Sum. With just the Model 1 feature the model is forced to predict an alignment for every source word. In Figure 4.11 we see that common words, such as the English the and French de,
they are constrained by limits which are imposed in order to ensure that the freedom of one person does not violate that of another.

Figure 4.11. Adding the NullM1Sum feature allows the model to align words to Null and avoid proposing weak alignments.

tend to act as default alignments for words that don’t strongly align with anything. Once we include NullM1Sum the model can effectively learn a threshold for these alignments and avoid over-prediction.

In Figure 4.2 we first evaluate the Dice and Model 1 based features independently. In this initial evaluation we can see that Model 1 appears to offer a better translation score than Dice, thus confirming that the probabilistic model is more useful than the heuristic co-occurrence score. Starting from just the one translation feature (Dice or Model1), we then add each of the additional features to the base feature and evaluate its contribution. For both translation score types we see that the CrossX features perform the best, allowing the model to evaluate alignment competition in both dimensions, across source words and across target words. Next we group all the features that had a positive contribution for each base feature and evaluate them together. Finally we group the Dice and Model 1 positively contributing features in one model and evaluate that. We see that the Model 1 based group actually outperforms the combined group. From this we conclude that it is better to use Model 1 alone and discard Dice translation scores from our future evaluations. We use
Table 4.3. Results comparing contributions of sentence position features.

<table>
<thead>
<tr>
<th>feature type</th>
<th>Trial f-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>+Base</td>
<td>0.787</td>
</tr>
<tr>
<td>+RelPosition</td>
<td>0.859</td>
</tr>
<tr>
<td>+AbsoluteDiff</td>
<td>0.835</td>
</tr>
<tr>
<td>+M1*RelPosition</td>
<td>0.811</td>
</tr>
</tbody>
</table>

Relative sentence position It is easy to observe that for closely related language pairs, such as English and French, words have a tendency to align with other words in a similar part of their respective sentences. A feature for the absolute difference in relative sentence position \( \text{abs}(\frac{a_i}{|e|} - \frac{t}{|f|}) \) allows the model to learn a preference for aligning words close to the alignment matrix diagonal (RelPosition). We also test a conjunction feature for the relative sentence position multiplied by the Model 1 translation probability \( (M1 \times \text{RelPosition}) \) and the raw absolute difference in sentence position \( \text{abs}(a_i - t) \) (AbsoluteDiff). From Table 4.3 we see that each of these three features make a positive contribution to our base features set.

Figure 4.12 shows that by adding the RelPosition feature we bias the model towards alignments along the diagonal, thus correcting the errors made on the French words: sont, que, and de.

Orthographic features Features based on string overlap allow our model to recognise cognates and orthographically similar translation pairs, which are particularly common between European languages. Here we employ a number of string matching features inspired by similar features in Taskar et al., 2005.

Table 4.4 shows the results for the orthographic features tested on the trial data set. We use an indicator feature for every possible source-target word pair in the training data (SourceTarget). This feature allows the model to learn a weight for an individual source-target translation and can be seen as providing a supervised version of the Model...
they are constrained by limits which are imposed in order to ensure that the freedom of one person does not violate that of another.

Figure 4.12. The effect of adding the RelPosition feature.

1 translation scores. Where Model 1 is trained in an unsupervised way, and any co-occurrence of a translation pair in the same sentence pair adds to their translation score, the supervised SourceTarget feature only counts pairs which are annotated as aligned in the word-aligned corpus. Thus we can see this feature as providing a high precision translation score, at the expense of coverage, as the word-aligned corpus is a fraction of the size of the sentence-aligned corpus used to train Model 1. Although this feature is very sparse considering our small training set, we still see a positive contribution by being able to learn weights over specific alignment pairs. We could hypothesise that given a more substantial training set these features would be more significant, possibly rivalling the effectiveness of the Model 1 translation features.

One feature which is particularly compelling for a discriminative model is exact string match (ExactMatch). While this obvious feature is trivial to include in our model, it would take significant effort to include it in the IBM Models. This feature should be of particular use for aligning names and numbers. Names are most often translated unchanged, for languages sharing the same alphabet, and should therefore be easy to align. However, we
Table 4.4. Results comparing contributions of orthographic features on the trial set.

<table>
<thead>
<tr>
<th>feature type</th>
<th>Trial f-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>+Base</td>
<td>0.787</td>
</tr>
<tr>
<td>+SourceTarget</td>
<td><strong>0.800</strong></td>
</tr>
<tr>
<td>+Dictionary</td>
<td>0.783</td>
</tr>
<tr>
<td>+Prefix</td>
<td><strong>0.790</strong></td>
</tr>
<tr>
<td>+Suffix</td>
<td><strong>0.788</strong></td>
</tr>
<tr>
<td>+PrefixMatch</td>
<td>0.787</td>
</tr>
<tr>
<td>+SuffixMatch</td>
<td>0.786</td>
</tr>
<tr>
<td>+EditDistance</td>
<td>0.787</td>
</tr>
<tr>
<td>+BothShort</td>
<td>0.781</td>
</tr>
<tr>
<td>+ExactMatch</td>
<td><strong>0.796</strong></td>
</tr>
<tr>
<td>+LenDiff</td>
<td>0.784</td>
</tr>
<tr>
<td>+NullSource</td>
<td>0.768</td>
</tr>
</tbody>
</table>

would expect to encounter many names in a test set that have not been observed in training. In this situation GIZA++ would assign a low alignment probability as it would not have any co-occurrence counts for the translation pair, while by incorporating the exact match feature our model can recognise that the pair has a strong alignment probability. Here we see that this feature does have a positive contribution on the trial set. Figure 4.13 shows how this feature is also effective in aligning matching numbers (25,000) that the base model doesn’t have Model 1 probabilities for.

Many aligned translation pairs are close, rather than exact, matches (e.g. the English *person* and French *personne* in Figure 4.13). One way of modelling the closeness of the orthography of the translation pair is to calculate the edit-distance between them. We tried a thresholded edit-distance (set to three in the reported case) between the source and target words as used as a binary feature (*EditDistance*). We tried a variety of threshold values and normalising the edit-distance by the word lengths, however this feature wasn’t effective on the trial set, possibly because of the difficulty in comparing edit distances across different word lengths.

Another method for capturing closeness of word orthography, which has proved effective in tagging tasks such as PoS tagging and named entity recognition, is to compare the prefixes and suffixes of the words. We used indicator features to test for matching prefixes
Figure 4.13. The ExactMatch feature allows the model to align numbers which are particularly sparse in the training data.

and suffixes of length three (PrefixMatch, SuffixMatch), as well as alignment suffix-prefix pairs (Prefix, Suffix). The matching pre/suffix features proved useful, possibly providing a better method of detecting words that have very similar spellings than the edit distance. The affix pair features provide a less sparse version of the SourceTarget feature by only considering the start/end of the translation pair in question. However these features didn’t contribute anything to the model, suggesting that the approximation was too coarse.

As stated earlier, the translation score feature often erroneously rewards alignments with common words. In order to address this problem, we include the absolute difference in word length as a real-valued feature (LenDiff), and an indicator feature which tests whether both words are shorter than 4 characters (BothShort) as these shorter words tend to occur more frequently. We hoped that together these features would allow the model to disprefer alignments between words with very different lengths – i.e. aligning rare (long) words with frequent (short) determiners, verbs etc. However in our evaluation they didn’t improve performance.
### Table 4.5.

<table>
<thead>
<tr>
<th>feature type</th>
<th>Trial f-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>+Base</td>
<td>0.787</td>
</tr>
<tr>
<td>+SPoSTPoS</td>
<td><strong>0.791</strong></td>
</tr>
<tr>
<td>+SourceTPoS</td>
<td>0.798</td>
</tr>
<tr>
<td>+SPoSTarget</td>
<td>0.792</td>
</tr>
<tr>
<td>+NullPoS</td>
<td><strong>0.811</strong></td>
</tr>
</tbody>
</table>

Results comparing contributions of part-of-speech tag features.

**Bilingual dictionary**  
Dictionaries are another source of information for word-alignment. We use a single indicator feature which detects when the source and target words appear as a translation pair in a bilingual dictionary (*Dictionary*). For the English-French dictionary we used FreeDict,\(^8\) which contains 8,799 English words, or 5,344 translation pairs once multi-word translations were removed. In Table 4.4 we see that the dictionary feature has a negative impact on performance. We believe this is mostly due to the small size of the dictionary, and that the Model 1 features with their much greater coverage provide a better measure of translation equivalence.

**PoS tags**  
Part-of-speech tags are an effective method for addressing the sparsity of the lexical features. For example, when the noun-adjective pair *Canadian experts* aligns with the adjective-noun pair *spécialistes canadiens*, the alignment exactly matches the parts-of-speech. Access to the words’ PoS tags will allow simple modelling of such effects. PoS tags could also be of use for less closely related language pairs, such as English and Japanese, where English determiners rarely align; and nor do Japanese case markers.

For our French-English language pair we PoS tagged the source and target sentences with TreeTagger.\(^9\) We created indicator features over the PoS tags of each candidate source and target word pair (*SPoS*), as well as over the source word and target PoS (and vice-versa) (*SourceTPoS, SPoSTarget*). Additionally we include the PoS tag of the source word as a feature for null alignments (*NullPoS*) with the idea that certain parts of speech may be more likely to be dropped in translation than others.

\(^8\)http://www.freedict.de  
\(^9\)http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger
### Table 4.6. Results comparing contributions of Markov features.

<table>
<thead>
<tr>
<th>feature type</th>
<th>Trial f-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>+Base</td>
<td>0.787</td>
</tr>
<tr>
<td>+AbsLabelDiff</td>
<td>0.862</td>
</tr>
<tr>
<td>+PositiveLDiff +NegativeLDiff</td>
<td>0.861</td>
</tr>
<tr>
<td>+FromNull +ToNull +BetweenNull</td>
<td>0.797</td>
</tr>
<tr>
<td>+ForwardLDiff +BackwardLDiff +NeutralLDiff</td>
<td>0.858</td>
</tr>
</tbody>
</table>

In Table 4.5 we see that these features all increase the model performance on the trial set.

**Markov features** Features defined over adjacent alignment labels allow our model to reflect the tendency for monotonic alignments between European languages and are a natural feature type to include in a CRF. We define a real-valued alignment index jump width feature ($\text{AbsLabelDiff}$):

$$jump\_width(t - 1, t) = \text{abs}(a_t - a_{t-1} - 1)$$

This feature has a value of 0 if the alignment labels follow the downward sloping diagonal, and is positive otherwise. This differs from the GIZA++ hidden Markov model which has individual parameters for each different jump width (Och and Ney, 2003; Vogel et al., 1996). We also tried separate features for positive and negative jump widths ($\text{PositiveLDiff}$, $\text{NegativeLDiff}$), as well as binary indicator features for the direction of the alignment difference ($\text{ForwardLDiff}$, $\text{BackwardLDiff}$ and $\text{NeutralLDiff}$). Figure 4.14 shows that by adding the $\text{AbsLDiff}$ feature the two mistakes made by the base feature set are corrected by the weight of the local alignment sequence.

As the jump width features are only valid when both the previous and current alignments are non-null, we also defined three indicator features over null transitions to allow the modelling of the probability of transition between, to, and from null labels ($\text{BetweenNull}$, $\text{ToNull}$ and $\text{FromNull}$ respectively).

We evaluate these features in Table 4.6, grouping the complementary features as they don’t make much sense independently, and see that they all contribute positively to the model. In particular the $\text{AbsLabelDiff}$ feature leads to the highest f-score of any of the
so far, around $250,000 have been issued by the institute in grants for research or public education activity.

**Figure 4.14.** The effect of adding the AbsLDiff feature.

<table>
<thead>
<tr>
<th>feature type</th>
<th>Trial f-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>+Base</td>
<td>0.787</td>
</tr>
<tr>
<td>+Null</td>
<td>0.797</td>
</tr>
</tbody>
</table>

**Table 4.7.** Adding a global null feature increases performance.

features we evaluated with the Base feature set, confirming that the ability to include sequence features using a CRF model is of benefit in word-alignment.

**Null** Finally we tried a single Null feature that is active for all null alignments. This feature allows the model to learn a global weight on aligning words to null. It would also be possible to modify the weight of this feature by hand in order to increase/decrease the recall of the model at the expense of precision. Table 4.7 shows that this feature is able to improve performance over our baseline feature set.
### 4.5.5 All positive features

In order to create a model for evaluation on the test data set we combine all features that had a positive contribution (shown in bold in the results tables) with our Base feature set and refer to this model as Best. This combined model achieves an f-score of 0.906 on the trial set.

### 4.5.6 Training with possible alignments

Up to this point our Hansards model has been trained using only the sure (S) alignments. As the data set contains many possible (P) alignments, we would like to use these to improve our model. Most of the possible alignments flag blocks of ambiguous or idiomatic (or just difficult) phrase level alignments. These many-to-many alignments cannot be modelled with our many-to-one setup. However, a number of possibles flag one-to-one or many-to-one alignments: for this experiment we used these possibles in training to investigate their empirical benefit on recall. We found no benefit in using many-to-many possible alignments as they added a significant amount of noise to the data.

Table 4.8 shows the effect of adding possibilities to training on many-to-many performance. In this table we measure precision, recall and f-score on the one-to-many and many-to-one alignments which our model is capable of capturing. Using these additional alignments with the Best feature set results in an f-score of 0.905 on the trial data set, which is not significantly different to training without them. However, we see a large increase in recall on the fertile alignments indicating that this training scheme does increase our models ability to handle these more difficult alignments.

<table>
<thead>
<tr>
<th>model</th>
<th>precision</th>
<th>recall</th>
<th>f-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best</td>
<td>0.353</td>
<td>0.038</td>
<td>0.148</td>
</tr>
<tr>
<td>Best +possibles</td>
<td>0.375</td>
<td>0.234</td>
<td>0.288</td>
</tr>
</tbody>
</table>

Table 4.8. The effect that using possible alignments in training has on many-to-many alignments for the trial set.
4.5.7 Model 4 as a feature

Previous work has demonstrated that by including the output of Model 4 as a feature, it is possible to achieve a significant decrease in AER, at the expense of having to train Model 4 (Taskar et al., 2005). We trained Model 4 in both directions on the two language pairs. We added two indicator features (one for each direction) and an intersection feature to our CRF which were active if a given word pair was aligned in the Model 4 output. Table 4.9 displays the results when these additional features are used. This produces a large increase in performance, and when including the possibles, increases the f-score to 0.945.

A significant disadvantage of using features derived from Model 4 is the large amount of time it takes to train GIZA++. If training time was an issue, an alternative would be to use one of the earlier, and faster to train, models such as Model 2 or 3, although we would expect this to have a negative impact on performance. At the opposite extreme, if training time was not an issue, we could incorporate features from all these models plus any other alignment systems we had access to. This ability to combine information from a variety of sources is one of the great strengths of the discriminative modelling approach.

4.6 Results

Table 4.10 shows the results obtained on the 347 sentence test set using our Best feature set, training with Possibles, and IBM Model 4 features. We report the results for both translation directions and when combined using the grow-diag-final heuristic method.

For each model type the best precision, recall and f-score is highlighted in the table. We see that the highest precision is obtained when we don’t use possibles, and when we do we trade some precision for increases in recall and f-score. With and without possible training
Table 4.10. CRF results on the English-French Hansards test data using the Best feature set. The best score for each metric is listed in bold.

<table>
<thead>
<tr>
<th>Feature Set</th>
<th>MAP-CRF</th>
<th>MM-CRF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prec</td>
<td>Rec</td>
</tr>
<tr>
<td>Best</td>
<td></td>
<td></td>
</tr>
<tr>
<td>French → English</td>
<td>0.970</td>
<td>0.857</td>
</tr>
<tr>
<td>English → French</td>
<td>0.975</td>
<td>0.834</td>
</tr>
<tr>
<td>English ↔ French</td>
<td>0.960</td>
<td>0.900</td>
</tr>
<tr>
<td>Best +Model4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>French → English</td>
<td>0.982</td>
<td>0.884</td>
</tr>
<tr>
<td>English → French</td>
<td>0.982</td>
<td>0.872</td>
</tr>
<tr>
<td>English ↔ French</td>
<td>0.977</td>
<td>0.914</td>
</tr>
<tr>
<td>Best +Possibles</td>
<td></td>
<td></td>
</tr>
<tr>
<td>French → English</td>
<td>0.950</td>
<td>0.885</td>
</tr>
<tr>
<td>English → French</td>
<td>0.965</td>
<td>0.832</td>
</tr>
<tr>
<td>English ↔ French</td>
<td>0.939</td>
<td>0.920</td>
</tr>
<tr>
<td>Best +Possibles +Model4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>French → English</td>
<td>0.968</td>
<td>0.913</td>
</tr>
<tr>
<td>English → French</td>
<td>0.974</td>
<td>0.880</td>
</tr>
<tr>
<td>English ↔ French</td>
<td>0.961</td>
<td>0.937</td>
</tr>
</tbody>
</table>

our model outperforms the GIZA++ baseline on both f-score and AER. When we include the GIZA++ output as feature for our model we see a large increase in performance over the GIZA++ baseline.

4.7 Analysis

On the Hansards task both our models achieves an AER better than the state-of-the-art GIZA++ Model 4 refined result of 7.0 (Och and Ney, 2003).\(^\text{10}\) Where Model 4 refined alignments have significantly higher recall than precision, our model has the opposite behaviour of higher precision than recall. This leads to the GIZA++ model achieving its highest f-score using the intersection for symmetrisation, while our model performs best with the more union like growing heuristic. Relative to previous research, our best AER of

\(^{10}\)The GIZA++ result is on the full test set, include the 100 sentences we used for training.
Chapter 4: CRFs for Word Alignment

<table>
<thead>
<tr>
<th></th>
<th>Prec</th>
<th>Rec</th>
<th>AER</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>English ↔ French (refined)</td>
<td>0.874</td>
<td>0.951</td>
<td>9.81</td>
<td>0.911</td>
</tr>
<tr>
<td>English ↔ French (intersection)</td>
<td>0.979</td>
<td>0.860</td>
<td>7.42</td>
<td>0.916</td>
</tr>
</tbody>
</table>

Table 4.11. GIZA++ baseline on the English-French Hansards test data.

<table>
<thead>
<tr>
<th></th>
<th>Prec</th>
<th>Rec</th>
<th>AER</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best baseline</td>
<td>0.960</td>
<td>0.900</td>
<td>6.84</td>
<td>0.930</td>
</tr>
<tr>
<td>- All Model 1 features</td>
<td>0.894</td>
<td>0.658</td>
<td>23.21</td>
<td>0.776</td>
</tr>
<tr>
<td>- All PoS features</td>
<td>0.961</td>
<td>0.878</td>
<td>7.81</td>
<td>0.920</td>
</tr>
<tr>
<td>- All Markov features</td>
<td>0.922</td>
<td>0.862</td>
<td>10.6</td>
<td>0.892</td>
</tr>
</tbody>
</table>

Table 4.12. The resulting F1 after removing groups of features from the Best feature set.

4.93 is better than the result of 5.4 achieved by Taskar et al., 2005 using a word-matching approach, while the quadratic assignment model of Lacoste-Julien et al., 2006 managed an AER of 4.5. When Model 4 is not used, our best AER of 6.84 is significantly better than the 10.7 of the word-matching approach, but slightly below the 6.2 of the quadratic assignment model.

When comparing the MAP and MM-CRFs we see that the MAP model generally outperforms the MM model on the f-score metric. This is particularly noticeable for the Best feature set. As we tuned the MM-CRFs C parameter only for the Best +Possibles +Model4 feature set, and the feature sets themselves were tuned for the MAP-CRF, this model may have been at a disadvantage in some experiments. However, the MM model appears to achieve a better balance between recall and precision (at least when Model 4 features are used), where the MAP model tends to favour precision. In Chapter 5 we explore how the recall vs. precision properties of our models effect end-to-end translation performance and assess the statistical significance of our alignment results in the context of a full translation system.
4.7.1 Feature contributions

Table 4.12 shows the effect of removing groups of features from the MAP-CRF model with the Best feature set. The most useful features are those derived from Model 1 values which allow the model to incorporate translation probabilities from the large sentence-aligned corpora, removing these leads to large decreases in both precision and recall. This is to be expected as the amount of word-aligned data is extremely small, and therefore the model can estimate translation probabilities for only a fraction of the lexicon. We would expect the dependence on sentence-aligned data to decrease as more word-aligned data becomes available.

Removing the part-of-speech derived features mostly impacts upon recall. In this case the model appears to be using the PoS tags to make alignment predictions for word pairs that were not seen in the training data, thus allowing the model to make alignment predictions which may otherwise have been aligned to null.

The effect of removing the Markov features can be seen from Figure 4.15, which shows an example from the test set. The model has learnt to prefer alignments that follow the diagonal and this allows it to capture the slightly non-literal translations of: would relieve ↔ réduirait and pressure on ↔ demande de. In these cases the Markov features serve as an extra piece of evidence for these alignments, without which the model acts conservatively and aligns the words to null.

4.7.2 Smoothing

Figure 4.16 depicts the effect of varying the Gaussian prior sigma parameter, where a higher value for sigma corresponds to less smoothing. We can see that there is not a single best value for sigma for optimising precision, recall and f-score together. Tighter smoothing leads to higher precision, possibly because this encourages the model to be more conservative and align words to null. The opposite is true of recall which peaks when less smoothing is used. As a result of this tuning graph, all the MAP-CRF models evaluated in this chapter used a sigma value of 1.0. The decline in performance as smoothing is lessened, shown on the right of the graph, indicates that the prior is having a significant positive effect on alignment results.
Figure 4.15. An example from the Hansard test set, showing the effect of removing the Markov features from the MAP-CRF model.

Figure 4.17 displays the analogous graph for the C parameter of the MM-CRF. This graph is less smooth than for the corresponding MAP-CRF graph, and there is a much starker trade-off between precision and recall. The highest f-score on the trial set was achieved with $C = 5e^{-6}$, and this is the value that we have used for all experiments in this chapter. It is of note that we could also select the point $C = 1e^{-6}$ which achieved almost the same f-score, but with the relationship between recall and precision reversed. Thus it appears that the C parameter can be effectively used to favour precision or recall in the MM-CRF model.
4.7.3 Many-to-many alignments

In order to assess the ability of our alignment models to make multi-word predictions on the test set, Table 4.13 shows the precision, recall and f-score on many-to-one and one-to-many alignments. The MAP-CRF model clearly outperforms the MM-CRF on many-to-one alignments, and both models outperform the GIZA++ f-score baseline. One explanation for this result is that by seeking a sparse solution the MM-CRF concentrates on modelling the more frequent one-to-one alignments, neglecting the difficult many-to-one alignments. In addition, as was found on the trial set, using the possibles in training gives a significant boost when predicting fertile alignments for both models.
Chapter 4: CRFs for Word Alignment

Figure 4.17. Relationship between the C soft-margin parameter of the MM-CRF and precision, recall and f-score on the trial set.

4.8 Additional Language Experiments

4.8.1 Romanian

In this section we apply our alignment model to a second data set, the Romanian-English parallel corpus from the 2005 ACL shared task (Martin et al., 2005). This consists of approximately 50,000 aligned sentences and 448 word-aligned sentences, which are split into a 248 sentence trial set and a 200 sentence test set. We used these as our training and test sets, respectively. For parameter tuning, we used the 17 sentence trial set from the Romanian-English corpus in the 2003 NAACL task (Mihalcea and Pedersen, 2003). For this task we have used the same test data as the competition entrants, and therefore can
Chapter 4: CRFs for Word Alignment

103

## MANY-TO-ONE ALIGNMENT RESULTS

<table>
<thead>
<tr>
<th></th>
<th>MAP-CRF</th>
<th></th>
<th>MM-CRF</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prec</td>
<td>Rec</td>
<td>F-score</td>
<td>Prec</td>
</tr>
<tr>
<td>Best</td>
<td>0.201</td>
<td>0.0623</td>
<td>0.0952</td>
<td>0.145</td>
</tr>
<tr>
<td>Best +possibles</td>
<td>0.341</td>
<td>0.314</td>
<td>0.327</td>
<td>0.0676</td>
</tr>
<tr>
<td>Best +Model4</td>
<td>0.342</td>
<td>0.314</td>
<td>0.327</td>
<td>0.138</td>
</tr>
<tr>
<td>Best +Model4 +possibles</td>
<td>0.451</td>
<td>0.376</td>
<td>0.410</td>
<td>0.272</td>
</tr>
<tr>
<td>GIZA++ baseline</td>
<td>0.151</td>
<td>0.228</td>
<td>0.182</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.13. The performance of each feature group on many-to-one alignment accuracy.

<table>
<thead>
<tr>
<th></th>
<th>Prec</th>
<th>Rec</th>
<th>AER</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best</td>
<td>0.824</td>
<td>0.778</td>
<td>19.96</td>
<td>0.801</td>
</tr>
<tr>
<td>Best +Dictionary</td>
<td>0.837</td>
<td>0.774</td>
<td>19.58</td>
<td>0.805</td>
</tr>
</tbody>
</table>

Table 4.14. Results evaluating the utility of the Dictionary feature for the Romanian-English trial data set.

directly compare our results. The word-alignments in this corpus were only annotated with sure (S) alignments, and therefore the AER is equivalent to $100 \times (1 - f$-score$)$.

In the shared task it was found that models which were trained on only the first four letters of each word obtained superior results to those using the full words (Martin et al., 2005). We observed the same result with our model on the trial set and thus have only used the first four letters when training the Model 1 translation probabilities. Using only the first four characters can be viewed as a rough form of stemming, ameliorating the sparse co-occurrence statistics in the small sentence-aligned corpus. This observation suggests that, in the case of this data set at least, the Romanian-English language pair could benefit from more principled morphological analysis.

With a system based on Model 4 plus additional parameters such as a dictionary, the Romanian-English result easily beats the best reported AER of 26.10 from the unlimited resources stream of the ACL shared task (Martin et al., 2005; Tuffis et al., 2005). That best model was a version of IBM Model 4 augmented with a Romanian-English dictionary extracted from an alignment of the Romanian and English WordNets. Although we haven’t evaluated our model without the dictionary in order to directly compare with the limited
resources stream, the feature tuning results from Table 4.14 indicate that the change in performance would be small and we’d still expect to easily outperform the best result of 26.55 from this stream. The standard Model 4 implementation in the shared task achieved an AER of 31.65, while when only the first 4 letters of each word were used it achieved 28.80. When comparing f-score, the CRF based models match the performance of the best GIZA++ model when no Model 4 features are used.

The Romanian-English language pair appears to offer a more difficult modelling problem than the French-English pair. With the translation score features (Model 1) removed – i.e. the sentence-aligned data is not used – the AER of the Romanian is more than twice that of the French, despite employing more word-aligned data. This could be caused by the lack of possible (P) alignment markup in the Romanian data, which provides a boost in AER on the French data set, rewarding what would otherwise be considered errors. Interestingly, without any features derived from the sentence-aligned corpus, our model achieves performance equivalent to Model 3 trained on the full corpus (Och and Ney.

---

Table 4.15. Results on the Romanian test data set with the MAP-CRF model.

<table>
<thead>
<tr>
<th>Romanian-English HANSARDS: TEST SET RESULTS</th>
<th>Prec</th>
<th>Rec</th>
<th>AER</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GIZA++ Model 4 Baseline</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Romanian ↔ English(refined)</td>
<td>0.805</td>
<td>0.641</td>
<td>28.63</td>
<td>0.714</td>
</tr>
<tr>
<td>Romanian ↔ English(intersection)</td>
<td>0.959</td>
<td>0.536</td>
<td>31.26</td>
<td>0.688</td>
</tr>
<tr>
<td><strong>Best</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Romanian → English</td>
<td>0.839</td>
<td>0.573</td>
<td>31.92</td>
<td>0.706</td>
</tr>
<tr>
<td>English → Romanian</td>
<td>0.828</td>
<td>0.575</td>
<td>32.13</td>
<td>0.702</td>
</tr>
<tr>
<td>Romanian ↔ English</td>
<td>0.777</td>
<td>0.652</td>
<td>29.10</td>
<td>0.714</td>
</tr>
<tr>
<td><strong>Best +Dictionary</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Romanian → English</td>
<td>0.841</td>
<td>0.573</td>
<td>31.84</td>
<td>0.707</td>
</tr>
<tr>
<td>English → Romanian</td>
<td>0.837</td>
<td>0.577</td>
<td>31.68</td>
<td>0.707</td>
</tr>
<tr>
<td>Romanian ↔ English</td>
<td>0.783</td>
<td>0.651</td>
<td>28.88</td>
<td>0.718</td>
</tr>
<tr>
<td><strong>Best +Dictionary +Model4</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Romanian → English</td>
<td>0.888</td>
<td>0.645</td>
<td>25.24</td>
<td>0.767</td>
</tr>
<tr>
<td>English → Romanian</td>
<td>0.878</td>
<td>0.649</td>
<td>25.38</td>
<td>0.763</td>
</tr>
<tr>
<td>Romanian ↔ English</td>
<td>0.834</td>
<td>0.703</td>
<td>23.74</td>
<td>0.768</td>
</tr>
</tbody>
</table>

---

11These results differ slightly from our Model 4 results reported in Table 4.15.
2003). This is a particularly strong result, as this model only has access to the 200 training examples, indicating that this method is ideal for data-impoverished alignment tasks.

4.9 Summary

In this chapter we have presented a novel approach for inducing word-alignments from sentence-aligned data. We showed how conditional random fields could be used for word-alignment. These models allow for the use of arbitrary and overlapping features over the source and target sentences, making the most of small supervised training sets, and can capture a limited number of many-to-many alignments. Moreover, we showed how the CRF’s inference and estimation methods allow for efficient processing without sacrificing optimality, improving on previous heuristic based approaches.

In summary:

• on both French-English and Romanian-English language pairs we showed that many highly predictive features can be easily incorporated into the CRF

• by progressive feature engineering on a trial set we have been able to identify a core set of high performing features

• with only a few hundred word-aligned training sentences, our model outperforms the generative Model 4 baseline

• we have found that Markov sequence features improve alignment quality, and that by training on the *possibles* for the French data allows for better modelling of word fertility

• when no features are extracted from the sentence-aligned corpus our model still achieves a high f-score

• dictionary based features can make a positive contribution, but only if the dictionary is large enough, as evidenced by the Romanian-English experiments

• through tuning experiments on the trial set we see that the C parameter of the MM-CRF allows us to make a precision/recall trade-off
In Chapter 4 we introduced and evaluated a new CRF based word-alignment algorithm that produces higher quality alignments than the previous benchmark GIZA++ system. Although we have demonstrated that our models offer improved alignments when evaluated intrinsically on the precision and recall of alignment points predicted, word-alignment is not an end unto itself and thus in this chapter we perform extrinsic evaluation of the impact of our improved alignments on the output of an SMT system.

There are a number of applications that can employ word-alignment as part of their processing pipeline (e.g. bilingual dictionary construction, word-sense disambiguation, text summarisation etc (Diab, 2000; Ker and Chang, 1997)). Here our primary test domain for word-alignment is SMT, and thus we seek to evaluate how the quality of our CRF alignments impact on the translations produced by an SMT system derived from them. The phrase-based SMT decoder we use as our reference system in this chapter is a large and complex piece of software and it is unclear how changing one component, the word-alignments, will effect the overall system. In particular we wish to evaluate which (if any) of our alignment metrics (precision, recall and f-score) correlate with statistical measures of translation quality.

A number of previous works have sought to establish a link between improved word-alignments and translation quality. Callison-Burch et al., 2004 performed experiments
that simulated incorporating supervised word-aligned data into the standard unsupervised \textsc{GIZA++} model. As they didn’t have access to enough manually word-aligned sentences, they used the output of a \textsc{GIZA++} system trained on a much larger data set as a quasi-gold standard word-aligned corpus. They trained a number of translation systems, using models which achieved a range of AERs on a word-aligned reference set. Their results on a German-English translation task showed that performance, as measured by the \textit{BLEU} metric (Section 5.1.4), increased as AER decreased. However, it was not stated in this work what aspects of the alignments, recall, precision or both, had lead to the decrease in AER.

The log-linear alignment model proposed by Ittycheriah and Roukos, 2005 was trained on a large number of word-aligned sentences and managed to significantly outperform a \textsc{GIZA++} baseline when evaluated with the \textit{BLEU} metric on an Arabic-English translation task (see Section 4.3.3). However, they also evaluated a reference HMM aligner which achieved a lower f-score than both \textsc{GIZA++} and the log-linear model but managed to slightly outperform \textsc{GIZA++}. This brings into question the link between improved alignment f-score and improved translation \textit{BLEU}.

Ayan and Dorr, 2006 attempted to evaluate the impact of alignment precision and recall on translation systems built for two language pairs, Arabic-English and Chinese-English. Their comparison of a number of systems with different recall/precision balances indicated that alignments with a good balance of precision and recall did best. The authors conclude that high precision alignments are preferable. However, as only a small number of systems with a limited balance of precision and recall were evaluated, this conclusion seems premature.

Fraser and Marcu, 2006b performed end-to-end evaluations for two alignment systems on Arabic-English and French-English language pairs. They found that their discriminative modification of the EM algorithm produced alignments with higher f-scores, and that this improvement carried over to higher BLEU scores for their translation system.

The range of results reported in the above cited works indicate that the link between word-alignment performance and translation metrics is far from clearcut. In this chapter we aim to specifically investigate whether our CRF alignments can improve a phrase-based translation system when compared to \textsc{GIZA++} alignments, and thus establish the utility
of our discriminative method. In addition we wish to determine whether there is any link between alignment metrics (recall, precision, f-score, AER) and translation metrics.

In Section 5.1 we describe the publicly available SMT system that we have used as a test-bed. In addition we suggest a simple maximum-margin training algorithm in Section 5.1.3 and introduce automatic translation evaluation metrics in Section 5.1.4. Section 5.2 reports the translation results using alignments produced from a selection of the models described in Chapter 4. In Section 5.3 we take advantage of the flexibility of the maximum-margin loss functions in order to train alignment models with specific recall/precision trade-offs, thus allowing us to investigate the relationship between these metrics and translation quality. Finally, in Section 5.4 we finish with a discussion of future directions for research involving word-alignment, maximum-margin methods, and SMT.

### 5.1 End-to-End Machine Translation

As mentioned in Chapter 4, word-alignment is only one step in the SMT translation pipeline. It is reasonable to expect that the effect of alignment quality on end-to-end translation will vary with the type of decoder used, be it word, phrase, or syntax based. Here we have chosen a phrase-based decoder primarily because this approach currently dominates SMT research (all of the works cited in the previous section used phrase-based decoders), and because of the availability of state-of-the-art open source phrase-based decoders. In this section we introduce the MOSES phrase-based decoder that we have used in our experiments, we then describe the standard minimum error rate training (MERT) algorithm, as well as our own maximum margin variant, and the automatic translation quality metric we have used.

#### 5.1.1 Moses Phrase-Based Decoder

In these experiments we have used the MOSES\(^1\) phrase-based decoder, which is a re-implementation and extension of the popular Pharaoh\(^2\) decoder. We use MOSES as a simple

---

\(^1\)http://www.statmt.org/moses/

\(^2\)http://www.isi.edu/publications/licensed-sw/pharaoh
phrase-based decoder, ignoring its additional capabilities, to create a phrase translation table from our CRF alignments, and to translate test sentences using a beam search.

The decoder uses a beam search that seeks to find the translation with the highest score, where a translation’s score is calculated from the weighted linear combination of features derived from the phrase translation table, language model, reordering model and word penalty. Ideally, each of these sub-model features provides a measure of the quality of the translation:

- the phrase translation table estimates the probabilities of the English and foreign phrases being good translations of each other
- the language model estimates how fluent the output English is
- the distortion model defines a cost on re-orderings between the source and target phrases, the more reordering, the higher the distortion cost
- the word penalty provides a means to penalize long, or short, translations.

Training MOSES consists of the following steps:

1. create bi-directional alignments
2. heuristically combine alignments
3. get lexical translation table
4. extract phrases
5. score phrases
6. build language model
7. run MERT training algorithm to estimate the weights of each feature

For our experiments we replace steps 1&2, which are normally performed using GIZA++, with a selection of our CRF based models. MOSES takes the word-alignments produced and first extracts a lexical translation table, which contains normalised frequency counts
for one-to-one word alignments. The phrase translation table is then extracted using the heuristic illustrated in Section 4.5. The language model is trained on the English sentences of the parallel corpus using the SRI language modelling toolkit.

5.1.2 Minimum Error Rate Training

The linear weighted feature combination described above can be formally written as a log-linear model:

\[
    p_{\Lambda}(e|f) = \frac{1}{Z(f, \Lambda)} \exp \left( \sum_{m=1}^{M} \lambda_m h_m(e, f) \right)
\]

(5.1)

\[
    Z(f, \Lambda) = \sum_{\hat{e}} \exp \left( \sum_{m=1}^{M} \lambda_m h_m(\hat{e}, f) \right)
\]

(5.2)

where the \(h_m\) function are the features (phrase translation probability, language model, etc.) and \(Z(f, \Lambda)\) is a normalising function ranging over all possible translation of the source sentence \(f\). Initially Och and Ney, 2002 suggested an algorithm for calculating the maximum entropy (maximum likelihood) model parameters for this log-linear model. Subsequently, the preferred approach has become to minimise an SMT specific error metric (e.g. word error rate or \(\text{BLEU}\) (Section 5.1.4)); this approach is referred to as minimum error rate training (MERT) (Och, 2003). The motivation for using the MERT approach is that there is only a loose relationship between maximum likelihood and translation quality, while MERT seeks to directly optimise more complex measures of translation quality.

There are two significant problems that need to be overcome in order to find the minimum error rate solution. Firstly, for phrase-based decoders with re-ordering, it is not feasible to perform the sum over all possible translations required by the \(Z\) function. In practice this problem is solved by an iterative approach which first approximates the space of all possible translations with an n-best list output by the decoder. The optimal model parameters are then found with respect to this n-best list, the decoder is then re-run with these new parameters to create a second n-best list, and the parameters are then optimised relative to the union of the two n-best lists. This procedure is iterated, adding the n-best

---

3http://www.speech.sri.com/projects/srilm/
lists at each iteration. The algorithm converges when no change is made to the global n-best list.

The second problem is that of finding the optimal parameters with respect to an n-best list, given that this optimisation space is in general non-convex and exhibits local-maxima. Och, 2003 describes a line search method which is efficiently able to find the global maximum by breaking the optimisation down into a set of intervals for each source sentence.

5.1.3 Maximum Margin Training

There are a number of problems with MERT that may impede it from finding a good set of model parameters. One issue is that by finding the model parameters which minimise the selected error metric on a development set we are liable to overfit that set, and thus the parameters may not generalise well in testing. Here we experiment with using a maximum margin training criterion which provides a more principled way of selecting model parameters and can reduce overfitting by incorporating regularisation.

We take a simple approach and use the final n-best list of MERT as a training set, reducing the problem to one of multi-class classification, where each candidate translation in the n-best list is treated as a class. The loss function we employ is based on the BLEU metric. Having run standard MERT training to acquire its final n-best list, for each sentence we find the candidate translation with the maximum BLEU with respect to the gold standard translation. This candidate is treated as the “local” gold standard and the loss of other candidates is defined as the difference in BLEU between the local gold standard and the candidate.

Using the slack-rescaling formulation from Section 3.4.4 we can identify the most violated constraint as the maximiser of:

\[
H(e) = (1 - w^T \Delta f_i(e)) (BLEU(\hat{e}) - BLEU(e))
\]  

\[
\hat{e} = \arg \max_{e \in \text{nbest}(f)} BLEU(e)
\]  

Using this cost function we run standard maximum margin training to find an alternate set of translation model parameters to those provided by MERT. This simple n-best maximum
margin approach should provide more generalisable parameters than standard MERT. However, the use of n-best lists by MERT to approximate the sum over all translations is highly questionable and our approach doesn’t address this problem. In Section 5.4 we consider how a more complete maximum margin approach could be implemented.

### 5.1.4 End-to-End Evaluation Measures

Although human evaluation is widely considered the only reliable measure of machine translation quality it is often prohibitively expensive, so a number of automatic machine translation evaluation measures have been proposed that attempt to approximate human judgements. In this chapter we employ the popular **BLEU** metric. See NIST, 2001 for a more in-depth critique of MT evaluation metrics.

**BLEU**

The **BLEU** metric is based on the computation of the geometric mean of the intersection of n-grams of various lengths in the predicted translation and a gold standard translation. This average of n-gram co-occurrences is then weighted by a brevity penalty (BP) which penalises predicted translations which are shorter than the gold standard.

\[
BLEU = BP \times \exp \left( \sum_{n=1}^{N} \frac{\log p_n}{N} \right) 
\]

\[
BP = \begin{cases} 
1, & \text{if } c > r \\
\exp(1 - \frac{c}{r}), & \text{otherwise} 
\end{cases}
\]

where \( r \) and \( c \) are the length of the gold standard and predicted translation respectively, and \( p_n \) is the precision of n-grams in the predicted translations.

**Statistical significance**

The differences in **BLEU** reported for the systems evaluated in this chapter are quite small in absolute terms. In order to clarify the impact of these results we include measures of their statistical significance. However, as the **BLEU** value for a test set cannot be calculated from a linear function of the **BLEU** values of the individual sentences standard
statistical significance tests cannot be applied. Here we use the tests described by Koehn, 2004b which are specifically tailored to machine translation and non-linear performance metrics:

**Bootstrap resampling** Given \( n \) test sentences and their correspond translations as output from a translation system, we repeatedly sample (1000 times) new virtual test sets, with replacement, of \( n \) sentences from the original set. We then calculate the **BLEU** score for each of these virtual test sets, sort the scores, and discard the bottom 2.5% and top 2.5%. The range of the remaining scores is used as the 95% confidence interval for that test set.

**Pairwise bootstrap resampling** In order to compare the performance of two systems to determine whether one is significantly better than the other we first translate the \( n \) test set with both systems. Analogously to bootstrap resampling we repeatedly sample with replacement \( n \) sentence virtual test sets from the original and calculate the **BLEU** scores of each system on each virtual test set. We then count the percentage of virtual test sets for which the first system has a higher **BLEU** score than the second. If the first systems outperforms the second on 95% of the virtual test sets then we conclude that it is better with 95% statistical significance.

In this chapter we present these statistical significance values as loose indicators only of the differences in the performances of the translation systems. The large size of the test set that we have used adds weight to the significance of any differences in **BLEU** scores.

### 5.1.5 Data

We evaluate our models using the Canadian Hansards House Debates test set.\(^4\) We take the first 10,000 sentences as our test set, and the next 2,000 sentences as the development corpus upon which the MERT and maximum margin training is performed. All models are trained using the same Canadian Hansards parallel corpus as for Chapter 4.

\(^4\)http://www.isi.edu/natural-language/download/hansard/
Chapter 5: End-to-End Translation with Discriminative Word-Alignments

Table 5.1. BLEU results from translation systems derived from a variety of alignments.

<table>
<thead>
<tr>
<th>Model</th>
<th>unigram ( (p_1) )</th>
<th>bigram ( (p_2) )</th>
<th>trigram ( (p_3) )</th>
<th>quadgram ( (p_4) )</th>
<th>ratio</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>GIZA++</td>
<td>65.4</td>
<td>40.0</td>
<td>27.2</td>
<td>18.9</td>
<td>1.010</td>
<td>34.05</td>
</tr>
<tr>
<td>MAP-CRF</td>
<td>65.1</td>
<td>40.3</td>
<td>27.8</td>
<td>19.6</td>
<td>1.013</td>
<td>34.54</td>
</tr>
<tr>
<td>MM-CRF</td>
<td>65.2</td>
<td>40.5</td>
<td>27.9</td>
<td>19.8</td>
<td>1.013</td>
<td>34.76</td>
</tr>
<tr>
<td>MM-CRF(C=1 \times 10^{-6})</td>
<td>65.4</td>
<td>40.7</td>
<td>28.1</td>
<td>20.0</td>
<td>1.011</td>
<td>34.94</td>
</tr>
</tbody>
</table>

Table 5.2. Statistical significance: 95% confidence interval and pairwise sampling of CRF models versus GIZA++. Significance is measured as described in 5.1.4, using the method of Koehn, 2004b.

<table>
<thead>
<tr>
<th>Model</th>
<th>95% confidence interval</th>
<th>vs. GIZA++</th>
</tr>
</thead>
<tbody>
<tr>
<td>GIZA++</td>
<td>33.88 – 34.91</td>
<td>-</td>
</tr>
<tr>
<td>MAP-CRF</td>
<td>34.03 – 35.10</td>
<td>( p &lt; 0.05 )</td>
</tr>
<tr>
<td>MM-CRF</td>
<td>34.20 – 35.28</td>
<td>( p &lt; 0.01 )</td>
</tr>
<tr>
<td>MM-CRF(C=1 \times 10^{-6})</td>
<td>34.50 – 35.57</td>
<td>( p &lt; 0.01 )</td>
</tr>
</tbody>
</table>

5.2 Results

Section 5.1 displays the BLEU results when MOSES is trained with alignments produced by a selection of the systems described in Chapter 4. Results with both the standard MERT algorithm and our variant maximum margin approach are presented. The MM-CRF\(C=1 \times 10^{-6}\) is a maximum margin alignment model trained with an alternate C value from the one used in Chapter 4. This model is chosen by observing that in Figure 4.17 we have a choice between choosing a C parameter that leads to high precision or high recall alignments, without much change in f-score. Thus MM-CRF\(C=1 \times 10^{-6}\) represents an alignment model with high recall in contrast to MM-CRF which is the original high precision model. The ratio column in Table 5.1 records the length of the reference translation divided by the length of the predicted translation. If this ratio is less than 1.0 a proportional word penalty is incorporated into the BLEU score (Section 5.1.4).
A first observation from Table 5.1 is that the maximum margin training gives superior \textit{BLEU} results for three of the four translation experiments. This adds weight to our hypothesis that MERT may be over-fitting, and that incorporating regularisation into the parameter estimates can be beneficial. However, the maximum margin improvements are not statistically significant at the 95% level, according to pairwise bootstrap sampling.

In terms of performance we see that our discriminative models uniformly lead to increased \textit{BLEU} scores over the \textsc{GIZA++} baseline, although the increases could be seen as small given that our models achieved much higher f-scores on the alignment evaluations. The CRF models achieve their performance gains through the higher order n-grams, suggesting that the larger phrase tables extracted from the higher precision alignments lead to greater n-gram coverage on the test translations. Table 5.2 records the results of statistical significance tests for the maximum margin trained translation systems. The translation systems based upon the two maximum margin alignment models (MM-CRF & MM-CRF\(_{C=1 \times 10^{-6}}\)) both produce statistically significant (above the 95% level) improvements when compared to the \textsc{GIZA++} based system, while the log-linear MAP-CRF alignment model falls slightly short.

\subsection{Effect on the translation model}

Table 5.3 records the phrase and lexical translation statistics for decoders built from each of the alignment models. From these statistics we can see that increasing the precision of the alignments leads to more phrases being extracted. As unaligned words lead the phrase extraction heuristic to extract all phrases consistent with them, the higher precision models, which are more conservative and often predict null alignments, license many more phrases. Whereas \textsc{GIZA++} rarely predicts null alignments and therefore more highly

\begin{table}[ht]
\centering
\begin{tabular}{lccc}
\hline
\textbf{Model} & \textbf{extracted phrases} & \textbf{unique phrases} & \textbf{lexical translations} \\
\hline
\textsc{GIZA++} & 29.1M & 5.1M & 1.9M \\
\textsc{MAP-CRF} & 52.1M & 18.3M & 555K \\
\textsc{MM-CRF} & 70.1M & 24.4M & 482K \\
\textsc{MM-CRF}_{C=1 \times 10^{-6}} & 63.0M & 21.0M & 638K \\
\hline
\end{tabular}
\caption{Alignment system effect on decoder parameters.}
\end{table}
Table 5.4. End-to-end results using alignments with varying recall/precision scores, listed in order of alignment f-score. The Spearman rank correlation between alignment recall and BLEU is 0.36, between precision and BLEU is -0.36, and f-score and BLEU is -0.63.

constrains the possible phrases. Conversely, as there are many less alignment points in the high precision CRF alignments, their corresponding lexical translation tables (one-to-one translation probabilities) are much smaller, and presumable, much less noisy. The explosion in the phrase table size of the CRF derived models is not a desirable property and is a result of the phrase extraction heuristic being tailored to the dense GIZA++ alignments. In order to more efficiently make use of the sparse alignments produced by the CRF models a heuristic specifically tailored to them would need to be developed.

5.3 Recall/Precision Trade-off

From the results presented in Table 5.1 it is difficult to draw any particular conclusions as to which alignment metrics most strongly correlate to BLEU. In order to test these results further we have investigated the use of a modified loss function for the MM-CRF model. By defining a parameterised loss function which penalises errors in recall more strongly than precision we can evaluate a number of models with a range of recall/precision characteristics.
We take the simple approach of introducing a parameter *null_penalty* which weights incorrect null alignments. The modified Hamming loss we use is defined as:

\[
l(y^i, y) = \sum_{1 \leq t < |y|} l^*(y^i_t, y_t)
\]

\[
l^*(\hat{y}, y) = \begin{cases} 
null\text{-penalty} \cdot l^0(\hat{y}, y), & \text{if } \hat{y} = \text{null\_alignment} \\
l^0(\hat{y}, y), & \text{otherwise}
\end{cases}
\]  

Using values less than 1.0 for *null\_penalty* implies that incorrectly aligning a word that should be null aligned will incur less loss than aligning a word to the wrong word. Thus, such a loss function has the effect of encouraging the model to be less conservative and guess alignments, increasing recall.

From Table 4.17 we saw that the C parameter of the MM-CRF model also had the effect of trading precision for recall. In order to select a set of models with a distribution of recall/precision values we performed a grid search over C and *null\_penalty* values. We then selected nine models which give a good range of recall/precision values and evaluate end-to-end translation models using the alignments produced by these models. Table 5.4 shows the alignment and *BLEU* results for each of the selected models, ordered by alignment f-score, using the maximum margin approach to train the translation model parameters.

It is hard to identify any strong relationship from the results in Table 5.4 and a particular alignment metric. In Figure 5.1 we graph recall, precision, and f-score versus *BLEU* and the line of best fit. Of these three graphs we see that recall is the only metric in which the line of best fit has a positive gradient. The Spearman rank correlation figures also confirm that there is a weak positive correlation between recall and *BLEU*, a weak negative one between precision and *BLEU*, and a stronger negative correlation between f-score and *BLEU*. However, clearly the results for all three metrics are not strongly linear and thus drawing any particular conclusions about correlations is difficult.

The difficulty in identifying a direct relationship between *BLEU* and an intrinsic alignment metric is most likely due to the complex nature of the heuristics required to go from an aligned corpus to a phrase translation model. Phrase translation models such as MOSES have been developed in concert with GIZA++ and thus are highly tuned to
its high recall, low precision alignment characteristics. Although we have demonstrated that our improved alignment models can lead to improved translation performance, to fully utilise these models would most probably require investigating different phrase extraction heuristics for MOSES.

5.4 Summary and Future Directions

In this chapter we have evaluated translation models built from word-alignments produced by our CRF models. These higher quality alignments lead to an increase in translation performance, when evaluated with the BLEU automatic translation evaluation metric. In particular the maximum margin trained alignment models led to statistically significant improvements at the 95% level over the baseline GIZA++ system. However, our further experiments to establish which, if any, of the word-alignment metrics were correlated with BLEU were inconclusive. An additional contribution of this work is the demonstration of an effective methodology for creating alignment models with specific precision and recall properties using weighted loss functions.

One explanation suggested for the difficulty in identifying a correlation between alignment and translation outputs is the nature of the phrase extraction heuristic. In particular it would be worthwhile investigating different methods of dealing with unaligned words when extracting phrases. One possibility would be to use multiple phrase tables, with a range of heuristics for dealing with unaligned words, from conservatively ignoring all of them, to the current approach of greedily generating every possible phrase. Each phrase table could then be added to the translation model as separate features, providing a form of back-off from high precision to high recall phrase translations.

We have also demonstrated a simple method for selecting the translation model parameters using a maximum margin criteria which incorporates regularisation, improving upon the popular MERT algorithm. It would be possible to extend this further and integrate the maximum margin training algorithm within the decoder beam search. In such an approach the beam search would seek the maximum of the maximum margin cost function
(H), in much the same way that the MM-CRF alignment model uses the modified Viterbi algorithm, thus eliminating the heuristic approach of iteratively generating n-best lists.
Figure 5.1. Relationship between alignment metrics and BLEU
Part III

Multilingual Supertagging
Chapter 6

CRFs for Supertagging

6.1 Introduction

The very large lexical type systems of strongly lexicalised grammars pose significant difficulties for both grammar developers and automatic parsers alike. The number of possible lexical types makes achieving 100% grammatical coverage and parse disambiguation particularly difficult. In this part of the thesis we investigate how structured classification models can be used to speed the creation of lexicalised grammars and improve the scalability of automatic parsing models based upon them. In particular we aim to develop models which are not specific to any one theory or language, and which can be quickly ported to a grammar developer's domain of choice. These aims inform our choice of generic feature representations and model structures.

We build and evaluate models for two grammar formalisms: Head Driven Phrase Structure Grammar (HPSG), and Combinatory Categorial Grammar (CCG). These formalisms occupy different ends of the spectrum of lexicalised grammars. Whereas HPSG is structure driven, CCG is primarily type driven. In HPSG (Pollard and Sag, 1994) lexical items are structured sets of features which describe both syntactic and semantic properties. These lexical types can be combined through unification according to a rich set of grammatical rules in order to produce an analysis of a sentence. In CCG (Steedman, 2000) the grammar is far simpler, containing a small closed set of operations that may be performed to combine or change the syntactic type and semantics of lexical categories. These categories encode
both the syntax and semantics of the lexical types by defining which other types they may combine with and in what order.

Both HPSG and CCG have found extensive use in linguistic analysis and implementations ranging from those requiring precise deep semantics, to statistical wide-coverage systems. References to the extensive bodies of research in these two areas can be found in Levine and Meurers, 2006 and Steedman and Baldridge, 2007 for HPSG and CCG respectively. We emphasise the neutrality of our approach by evaluating our statistical models on both these formalisms, demonstrating that a substantial proportion of syntactic structure can be determined locally.

The first task we look at is that of deep lexical acquisition (DLA), in which we seek to propose new lexical entries for a grammar by training a model that tags words in context with their lexical type. The second task we tackle is to find a restricted subset of lexical types for words in a sentence, which can then be used to drastically reduce the search space that an automatic parser is required to explore.

Both of these tasks are approached as instances of supertagging, which treats the task as structured sequence prediction for sentences, where we aim to assign each word with one or more tags from a very large tag-set. It was first introduced as a means of reducing parser ambiguity by Bangalore and Joshi, 1999 in the context of the LTAG formalism, and has since been applied in a similar context within the CCG formalism (Clark and Curran, 2004b). In both of these cases, supertagging provides the means to perform an argmax search over the plausible lexical items for a given string context, and ideally reduces parsing complexity without sacrificing parser accuracy. Although these tasks could be viewed as simple extensions of part-of-speech (PoS) tagging, a number of properties distinguish them for special consideration. The tags assigned in supertagging are far more precise than those used in PoS tagging, resolving a large proportion of the syntax of a sentence, and thus there must be many more of them. In addition, although we treat them as atomic, supertags contain internal structure that can influence their interaction with other tags. Such a large tag-set requires models to scale at most linearly with the number of tags, thus posing difficulties for the straight application of graphical CRF models that scale quadratically. We solve this problem using a pseudo-likelihood approximation for training our models
Investors are appealing to the Exchange Commission

NP (S\NP)/(S\NP) (S\NP)/PP PP/NP NP/N N/N N

Figure 6.1. The CRF model for supertagging an input sentence using CCG categories as tags.

which uses additional conditioning assumptions in a principled way to allow our model to scale linearly with the number of labels.

The aim of supertagging is to predict rich syntactic structure for words in context, using only local information. We can think of lexicalised grammars as projecting much of the syntactic structure onto the words, and supertagging as predicting this structure, in the form of tags, without resorting to hierarchical parsing algorithms. Obviously much of the information present in such tags is non-local, possibly with dependencies spanning entire sentences. However, previous research on supertagging (Bangalore and Joshi, 1999; Clark and Curran, 2004b) has shown that the majority of lexical information can in fact be predicted from local context. This has allowed the development of very efficient parsing algorithms for heavily lexicalised grammars, for example the work of Clark and Curran, 2004a with Combinatory Categorial Grammar.

We treat supertagging as an instance of structured sequence classification (Chapter 3). In this chapter we describe the application of structured classification to supertagging, the formulation of the model proposed and the features employed.
6.2 A Pseudo-Likelihood CRF for Supertagging

Undirected graphical sequence models such as conditional random fields (CRFs) (Chapter 3) are a natural fit for the task of supertagging. They efficiently model local dependencies between tags, while allowing the incorporation of a wide variety of contextual features from the input sentence such as surrounding words and morphology.

The structure of a CRF for a CCG supertagging model is shown in Figure 6.1. In this example the label for the input sentence consists of a first order Markov sequence of CCG category tags, one for each word. Although the tags contain non-local syntactic information, most can be seen to exhibit a strong local bias. For instance, the tagging of $\text{Exchange}_{NN} \text{Commission}_{NN}$ can be determined locally, while the tagging of $\text{are}(S\backslash NP)/(S\backslash NP)$ strongly biases the tag on the left to provide an $NP$ and the tag on the right to provide an $S\backslash NP$, both of which are correct predictions. Thus, although a locally conditioned sequence model is unlikely to be able to completely determine the labelling of a sentence, it can create a distribution over labels that can be used by other systems to restrict their search.

Although the modelling properties of CRFs make them well suited to the task of supertagging, a severe limitation of the direct application of such models is the large tag sets required. CRFs scale quadratically in the number of states (tags), and therefore with the hundreds of tags and tens of thousands of sentences required to train a practical supertagging model, the standard training algorithms are not viable. In this work we use the pseudo-likelihood CRF training approximation described in Section 3.4.3. This training criterion employs a local word normalisation, rather than the standard global sentence normalisation, thus avoiding the need to calculate a costly normalisation function $(Z)$ over the entire label set of all possible tag sequences. This allows training with scaling that is linear in the size of the tag set (number of model states), while decoding the model is unchanged from standard CRF decoding with the Viterbi algorithm, which scales quadratically.

Here we restate the pseudo-likelihood equations from Section 3.4.3. We approximate the node marginals of the full likelihood by fixing the states of the nodes’ neighbours to
those observed in the training sample, and normalising over the possible states for this node:

\[ U_{PL}^{\Lambda}(i, s, t) = \sum_{k} \lambda_k \left( f_k(\hat{y}_{t-1}, i, x, t) + f_k(i, \hat{y}_{t+1}, x, t) \right) \]

\[ p_{PL}^{\Lambda}(y|x) = \prod_t \frac{\exp(U_{PL}^{\Lambda}(y_t, x, t))}{\sum_{s \in S} \exp(U_{PL}^{\Lambda}(s, x, t))} \]  

(6.1)

where \( \hat{y}_i \) is the state observed in the training data and \( s \) ranges over the state set.

As described in Chapter 3 we can choose to optimise the likelihood of the training data in order to derive a probabilistic model (Equation (3.49)), or we can use a maximum margin training criterion which optimises the classification abilities of the model with respect to some loss function (Equation (3.55)). We present results using both training criteria in order to draw comparisons between the various strengths of the different modelling approaches. For the probabilistic model we use the maximum a posteriori (MAP) training criteria, which incorporates a Gaussian prior distribution over the model parameters. We explored a selection of values for the MAP Gaussian and maximum margin C parameters on a development set, finding little variance in performance. Thus we used the default values: for all the MAP trained supertagging models described we have used a Gaussian prior with zero mean and unit variance; for the maximum margin C parameter we used 0.1.

### 6.2.1 Generic Feature Set

One of the strengths of the CRF model is that it supports the use of a large number of non-independent and overlapping features of the input sentence. Table 6.1 lists the word context and lexical features used by the proposed CRF supertagging model (shared across both tasks). In Chapter 4 we demonstrated how specifically engineering features for a particular task could improve model performance. In this chapter we are concerned with achieving good modelling performance without engineering our feature set for a particular language or theory of grammar. By taking this approach we hope to create a tool which can assist a grammar or parser developer across a range of tasks without additional engineering overhead. Obviously, in order to achieve maximal performance for a particular grammar of language specific engineering would be required.
Chapter 6: CRFs for Supertagging

FEATURE DESCRIPTION

<table>
<thead>
<tr>
<th>WORD CONTEXT FEATURES</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{lexeme}(x_t) = l \ &amp;\ y_t = s )</td>
<td>lexeme + state</td>
</tr>
<tr>
<td>( x_t = w \ &amp;\ y_t = s )</td>
<td>word unigram + state</td>
</tr>
<tr>
<td>( x_{t-1} = w \ &amp;\ y_t = s )</td>
<td>previous word unigram + state</td>
</tr>
<tr>
<td>( x_{t+1} = w \ &amp;\ y_t = s )</td>
<td>next word unigram + state</td>
</tr>
<tr>
<td>( x_t = w \ &amp;\ x_{t-1} = v \ &amp;\ y_t = s )</td>
<td>previous word bigram + state</td>
</tr>
<tr>
<td>( x_t = w \ &amp;\ x_{t+1} = v \ &amp;\ y_t = s )</td>
<td>next word bigram + state</td>
</tr>
<tr>
<td>( y_{t-1} = s \ &amp;\ y_t = m )</td>
<td>clique state pair</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LEXICAL FEATURES</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{prefix}_n(x_t) \ &amp;\ y_t = s )</td>
<td>( n )-gram prefix + state</td>
</tr>
<tr>
<td>( \text{suffix}_n(x_t) = u \ &amp;\ y_t = s )</td>
<td>( n )-gram suffix + state</td>
</tr>
<tr>
<td>( \text{contains}(x_t, C_i) \ &amp;\ y_t = s )</td>
<td>word contains element of character set ( C_i ) + state</td>
</tr>
</tbody>
</table>

| Table 6.1. Extracted feature types for the CRF model. |

Word context features were extracted from the words and lexemes of the sentence to be labelled, combined with a proposed label. A clique state pair feature was also used to model sequences of lexical types. The actual set of instantiated features is defined by those features attested by examples in the training set. Only attribute/state, or clique state pair, features that are actually observed in the training data are used.

For the lexical features, we generate a feature for the unigram, bigram and trigram prefixes and suffixes of each word (e.g. for *bottles*, we would generate the prefixes *b*, *bo* and *bot*, and the suffixes *s*, *es* and *les*); for words in the test data, we generate a feature only if that feature-value is attested in the training data. We additionally test each word for the existence of one or more elements of a range of character sets \( C_i \). In the case of English, we focus on five character sets: upper case letters, lower case letters, numbers, punctuation and hyphens. For the Japanese data, we employ six character sets: Roman letters, hiragana, katakana, kanji, (Arabic) numerals and punctuation. For example, *w* “mouldy”ould be flagged as containing katakana character(s), kanji character(s) and hiragana character(s) only. Figure 6.2 shows example realisations of these features for English and Japanese. Note that the only language-dependent component of the lexical features is the character sets, which requires little or no specialist knowledge of the language. Note also that for languages with inflexing, such as Tagalog, we may want to include \( n \)-gram infixes in
addition to \( n \)-gram prefixes and suffixes. Here again, however, the decision about what range of affixes is appropriate for a given language requires only superficial knowledge of its morphology.

In Figure 6.3 we list the feature functions which are active for the graph node corresponding to the word *to* with the state *PP/NP*. Note that clique features are active which incorporate the gold state for both the previous and following graph nodes, as described for the pseudo-likelihood model. Only the lower case character feature is active for the word *to* as it doesn’t include characters from the other four English character sets. The pseudo-likelihood training extracts a set of features for every possible state assignment for the node, given the gold states of its neighbours, and aims to maximise the weights for the correct state.

### 6.3 Part-of-Speech Tagging

In order to demonstrate the effectiveness of the pseudo-likelihood CRF approximation, we compare its performance with full CRF training on the task of part-of-speech (PoS) tagging. PoS tagging is the process of assigning syntactic function tags (*noun*, *verb* etc.), to words in context. Such taggings are often employed as a starting point for more complex processing, such as parsing or named entity detection. A direct comparison of CRF training
Figure 6.3. The features extracted for the training instance associated with the node for *to*.

algorithms for supertagging would not be computationally tractable due to the large number of lexical types modelled. However, typical PoS tagsets are much smaller than those employed in supertagging, making it feasible to train a full CRF, without approximation, with which to compare our pseudo-likelihood model.

We use the Wall Street Journal sections of the Penn Treebank (Marcus et al., 1993) as the source of our training and testing data. There are 48 unique PoS tags present in the training sample. Table 6.2 lists additional statistics for the training and testing samples. The two models we compare are:
Chapter 6: CRFs for Supertagging

<table>
<thead>
<tr>
<th>WSJ PoS CORPUS</th>
<th>Training</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sections</td>
<td>2–21</td>
<td>23</td>
</tr>
<tr>
<td>Sentences</td>
<td>39,604</td>
<td>2,407</td>
</tr>
<tr>
<td>Words</td>
<td>929,552</td>
<td>55,371</td>
</tr>
<tr>
<td>Novel types</td>
<td>–</td>
<td>1,337</td>
</tr>
</tbody>
</table>

Table 6.2. Wall Street Journal PoS corpus statistics.

<table>
<thead>
<tr>
<th>TRAINING SENTENCES</th>
<th>CRF\textsuperscript{MAP}</th>
<th>CRF\textsuperscript{PL-MAP}</th>
</tr>
</thead>
<tbody>
<tr>
<td>125</td>
<td>0.856</td>
<td>0.855\textsuperscript{†}</td>
</tr>
<tr>
<td>250</td>
<td>0.897</td>
<td>0.896\textsuperscript{†}</td>
</tr>
<tr>
<td>500</td>
<td>0.915</td>
<td>0.915</td>
</tr>
<tr>
<td>1K</td>
<td>0.935</td>
<td>0.935</td>
</tr>
<tr>
<td>2K</td>
<td>0.946</td>
<td>0.947</td>
</tr>
<tr>
<td>4K</td>
<td>–</td>
<td>0.959</td>
</tr>
<tr>
<td>8K</td>
<td>–</td>
<td>0.964</td>
</tr>
<tr>
<td>16K</td>
<td>–</td>
<td>0.968</td>
</tr>
<tr>
<td>32K</td>
<td>–</td>
<td>0.971</td>
</tr>
<tr>
<td>ALL</td>
<td>–</td>
<td>0.971</td>
</tr>
</tbody>
</table>

Table 6.3. Accuracy results for PoS tagging WSJ Section 23 with a range of training corpus sizes. †Marked results are statistically significantly different than the CRF\textsuperscript{MAP} results at the 95% confidence level.

\textbf{CRF}\textsuperscript{MAP} This model is trained using the standard CRF maximum \textit{a posteriori} (MAP) training algorithm. This algorithm requires the use of the forward-backward algorithm (Section 3.3.2) in order to calculate the feature expectations, incurring complexity quadratic in the number of tags.

\textbf{CRF}\textsuperscript{PL-MAP} This model is trained using the pseudo-likelihood MAP algorithm described in Section 6.2, having complexity linear in the number of tags.

The full set of English features from Table 6.1, including the lexical features, are employed in both models.

The results of the comparison are shown in Table 6.3 and Figure 6.4. We trained each model on increasing numbers of sentences from the full corpus, starting at 125 sentences and doubling. All the models were tested on the full test set. Due to computational resource limitations, principally RAM, the CRF\textsuperscript{MAP} model was unable to be trained on more than 2K sentences. On the two smallest training sets the full model performs marginally better than the pseudo-likelihood approximation. While as the number of training sentences is increased the difference become statistically insignificant. These results indicate that the pseudo-likelihood model is a particularly good approximation when more than a couple of hundred training sentences are available. We also note that the CRF\textsuperscript{PL-MAP} model trained
Figure 6.4. Performance of the MAP and pseudo-likelihood trained models with respect to the number of training sentences.

on the entire training sample achieves accuracy competitive with the current state-of-the-art results of 97.2% reported by Toutanova et al., 2003. This is a particularly encouraging result given the simplicity of our feature set.

6.4 Summary

In this chapter we have introduced supertagging and framed it as a structured classification task. Motivated by the aim to create a grammar and language independent model, we have introduced our supertagging CRF and a generic feature set. We have demonstrated the effectiveness of the pseudo-likelihood approximation by comparing it to the standard training algorithm on the task of PoS tagging.
In subsequent chapters we explore the space of applications for supertagging with two scenarios: firstly that of predicting novel tag-word pairs in a test corpus with high precision (deep lexical acquisition, Chapter 7), and secondly predicting the smallest subset of tags for a word that is highly likely to contain the correct tag (parsing search reduction, Chapter 8). The first of these applications emphasises the identification of word/lexical type taggings in a test corpus that have not been observed in training, while the second requires the partial disambiguation of lexical types for every word in the test corpus. Both these scenarios require our models to be able to trade off precision for recall in their tagging predictions, or alternatively, to be able to multi-tag words with sets of probable tags for a parser to choose from. In all the experiments in the following chapters we use the pseudo-likelihood supertagging model (and its maximum-margin analogy).
Chapter 7

Supertagging for Deep Lexical Acquisition

7.1 Introduction

Over recent years, there has been a resurgence of interest in the use of precision grammars in NLP tasks, due to advances in parsing algorithm development, grammar development tools and computational power (Oepen et al., 2002b). Precision grammars are defined as grammars of natural language which aim to capture fine-grained linguistic distinctions, and are generative in the sense of distinguishing between grammatical and ungrammatical inputs (having some in-built notion of linguistic “markedness”). Additional characteristics of precision grammars are that they are frequently bidirectional, and may output a rich semantic abstraction for each spanning parse of the input string. Examples of such grammars include those based on DELPH-IN, such as the English Resource Grammar (Flickinger, 2002; Uszkoreit, 2002), the various PARGRAM grammars (Butt et al., 1999), the Edinburgh CCG parser (Bos et al., 2004), and OPENCCG (White, 2006).

Due to their high degree of linguistic complexity, precision grammars are generally hand-constructed and are thus restricted in size and coverage. Attempts to (semi-)automate the process of expanding the coverage of precision grammars have focused on two areas. The first is constructional coverage, using either error mining for constructional expansion (van Noord, 2004; Zhang and Kordoni, 2006), or relaxation of lexicogrammatical
constraints to support partial and/or robust parsing (Riezler et al., 2002). The second is lexical coverage, such as using a pre-existing grammar and lexicon to bootstrap a system for learning new lexical items (Baldwin, 2005a). In this chapter our interest is in the latter of these two, the development of methods for automatically expanding the lexical coverage of an existing precision grammar, referred to as deep lexical acquisition (DLA hereafter). Here we follow Baldwin, 2005a in assuming a grammar with a fixed set of lexical types and aim to learn new lexical items. For the purposes of this chapter, we focus specifically on supertagging as the mechanism for proposing new lexical items. We explore the parsing applications of supertagging in Chapter 8.

In the case of DLA, supertagging can be defined as the process of applying a structured classifier to the task of predicting the lexical type associated with each word in an input string, relative to a given grammar. This constitutes an alternate application of supertagging from that presented by Bangalore and Joshi, 1999, in that we are postulating novel lexical items with which to populate the lexicon of a given grammar to boost parser coverage. This can take place off-line for the purposes of increasing the coverage of a static lexicon, in which case we are generally interested in globally maximising precision over a given corpus and hence predicting the single most probable lexical type for each word token (Baldwin, 2005b). Alternatively, DLA can be performed on-line for a given input string to temporarily expand lexical coverage and achieve a spanning parse, in which case we are interested in maximising recall by producing a ranked list of lexical item hypotheses to run past the grammar (Zhang and Kordoni, 2005). In this chapter we focus on the off-line approach, although it would be a trivial extension to integrate our approach with a parser to perform on-line DLA.

Our training and testing data is drawn from the Grammar Matrix-based DELPH-IN family of grammars (Bender et al., 2002), which includes grammars of English, Japanese, Norwegian, Modern Greek, Portuguese and Korean. The Grammar Matrix is a framework for streamlining and standardising HPSG-based multilingual grammar development. One property of Grammar Matrix-based grammars is that they are strongly lexicalist and adhere to a highly constrained lexicon-grammar interface via a unique lexical type for each lexical item. As such, the creation of lexical items in any of the Grammar Matrix-based grammars, irrespective of language, mostly consists of predicting the lexical type for each
lexical item, relative to the lexical hierarchy for the grammar. In order to stay within this framework of standardisation and multilingualism, we also aim to develop language-independent supertagging methods which can be applied to any Grammar Matrix-based grammar with the minimum of effort. This aim informs our choice of simple feature sets which avoid the assumption of complex preprocessing algorithms, for example PoS taggers, which may not be readily available for many languages. Essentially, we hope to provide the grammar engineer with the means to semi-automatically augment the lexicon of a grammar, hence increasing the pace of grammar development and producing a resource with sufficient coverage to be of practical use in NLP tasks.

The remainder of this chapter is structured as follows. Section 7.2 outlines past work relevant to this research, and Section 7.3 reviews the resources used in our DLA supertagging experiments. Section 7.4 then describes our experimental set-up and the results of our evaluations. In Section 7.5 we summarise the results of this research and identify our key contributions.

### 7.2 Past Research

Baldwin, 2005b divides DLA research into two categories: *in vitro* and *in vivo*. For *in vitro* DLA a secondary language resource, such as the web, is used to generate an abstraction of the words we hope to learn lexical items for. In the case of *in vivo* methods, the target resource that we are hoping to perform DLA for is directly used to perform DLA. Supertagging is an instance of *in vivo* DLA, as it treats the task as discriminative learning from data tagged with the lexical types of the precision grammar of interest.

Research on supertagging which is relevant to this chapter includes the work of Baldwin, 2005b in training a transformation-based learner (TBL) over data tagged with ERG lexical types. The FNtbl toolkit used in this research is highly optimised for English PoS tagging, making this research less language independent than our own. We discuss this method in detail in Section 7.4.2 and replicate it over our English data set for direct comparability with this previous research.
One related \textit{in vivo} approach to DLA for precision grammars is that of Fouvry, 2003. Fouvry uses the grammar to inform the process of learning lexical items for unknown words, by generating underspecified lexical items for all unknown words and then parsing with them. Syntactic-semantic interaction between unknown and known lexical items during parsing provides insight into the type of each unknown word. By combining such fragments of information, it is possible to incrementally build a lexical entry for that word. In this method the precision grammar itself drives the learning process within a parsing context. The main limitation of such an approach is that the grammar may license many lexical items for a given unknown word, with the number increasing with the length of the sentence. This reduces the specificity of the induced lexical entries and leads to large-scale spurious ambiguity. In addition, it is often the case that there are a number of unknown words in a given sentence leading to an extremely large number of interpretations and possible lexical entries. Finally, generating lexical entries in this fashion doesn’t calculate any probability or confidence score for a predicted entry, something that could significantly reduce the work of the grammar developer.

An alternate approach is to create a set of contextual word templates for each lexical type (with the qualification that they do not rely on preprocessing of any form), and check for corpus occurrences of an unknown word in such contexts. In this approach the morphological, syntactic and/or semantic predictions implicit in each lexical type are made explicit in the form of templates which represent distinguishing contexts of that lexical type. This approach has been shown to be particularly effective over web data, where the sheer size of the data precludes the possibility of linguistic preprocessing but at the same time lessens the effects of data sparseness inherent in any lexicalised DLA approach (Lapata and Keller, 2004).

Other DLA research, such as Korhonen, 2002, Joanis and Stevenson, 2003, Baldwin, 2005a has taken the \textit{in vitro} DLA approach, employing web or corpus data in order to extrapolate away from a DLR, and analysed word contexts through the use of external tools such as parsers or PoS taggers. Hockenmaier et al., 2004 explored a wide range of statistical techniques for extending the coverage of a CCG lexicon. Both syntactic and semantic information was mined from resources such as the Penn Treebank and pre-existing hand build lexicons using decision trees and a maximum entropy tagger. Of particular note, the
application of a maximum entropy tagger to predicting CCG categories on the WSJ corpus is possibly one of the earliest works on supertagging for lexicalised grammars.

### 7.3 Task and Resources

In this section, we outline the resources targeted in this research, namely the English Resource Grammar (ERG: Flickinger, 2002, Copestake and Flickinger, 2000) and the JACY grammar of Japanese (Siegel and Bender, 2002). Note that our choice of the ERG and JACY as testbeds for experimentation is somewhat arbitrary; we could equally run experiments over any Grammar Matrix-based grammar for which there is treebank data.

Both the ERG and JACY are open-source broad-coverage precision Head-driven Phrase Structure Grammars (HPSGs: Pollard and Sag, 1994). A lexical item in each of the grammars consists of a unique identifier, a lexical type (a leaf type of a type hierarchy), an orthographic form, and a semantic relation. For example, in the English grammar, the lexical item for the noun *dog* is simply:

```plaintext
dog_n1 := n_-_c_le & 
    [ STEM < "dog" >,
      SYNSEM [ LKEYS.KEYREL.PRED "_dog_n_1_rel" ] ].
```

in which the lexical type of *n_-_c_le* encodes the fact that *dog* is a noun which does not subcategorise for any other constituents and which is countable, "dog" specifies the lexical stem, and "_dog_n_1_rel" defines a predicate name for the lexical item to use in constructing a semantic representation. In the context of the ERG and JACY, the task of DLA aims to learn the set of lexical types a given lexeme occurs with, generating a single lexical item for each. In the example above, we would aim to tag occurrences of the lexeme *dog* in our test corpus with the lexical type *n_-_c_le*, thus identifying this pairing as a candidate for a lexical item.

Recent development of the ERG and JACY has been tightly integrated with treebank annotation, and all major versions of both grammars are deployed with a common set of dynamically-updateable treebank data to help empirically trace the evolution of the grammar and retrain parse selection models (Oepen et al., 2002a; Bond et al., 2004). This
serves as a source of training and test data for building our supertaggers, as detailed in Table 7.1.

In translating our treebank data into a form that can be understood by a supertagger, multiword expressions (MWEs) we must be dealt with. Both the ERG and JACY include multiword lexical items, which can either be strictly continuous (e.g. *hot line*) or optionally discontinuous (e.g. transitive English verb particle constructions, such as *pick up* as in *Kim picked the book up*).

Strictly continuous lexical items are described by way of a single whitespace-delimited lexical stem (e.g. *STEM < "hot line" >*). When faced with instances of this lexical item, the supertagger must perform two roles: (1) predict that the words *hot* and *line* combine together to form a single lexeme, and (2) predict the lexical type associated with the lexeme. These roles are combined in a single step through the use of a ditto lexical type, which indicates that the current word combines with the left-adjacent word to form a single lexeme, and shares the same lexical type. This tagging convention is based on that used in the CLAWS7 part-of-speech tagset.

<table>
<thead>
<tr>
<th></th>
<th>ERG</th>
<th>JACY</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GRAMMAR</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Language</td>
<td>English</td>
<td>Japanese</td>
</tr>
<tr>
<td>Lexemes</td>
<td>16,498</td>
<td>41,559</td>
</tr>
<tr>
<td>Lexical items</td>
<td>26,297</td>
<td>47,997</td>
</tr>
<tr>
<td>Lexical types</td>
<td>915</td>
<td>484</td>
</tr>
<tr>
<td>Strictly continuous MWEs</td>
<td>2,581</td>
<td>422</td>
</tr>
<tr>
<td>Optionally discontinuous MWEs</td>
<td>699</td>
<td>0</td>
</tr>
<tr>
<td>Proportion of lexemes with more than one lexical item</td>
<td>0.29</td>
<td>0.14</td>
</tr>
<tr>
<td>Average lexical items per lexeme</td>
<td>1.59</td>
<td>1.16</td>
</tr>
<tr>
<td><strong>TREEBANK</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Training sentences</td>
<td>10,000</td>
<td>40,000</td>
</tr>
<tr>
<td>Training words</td>
<td>113,316</td>
<td>393,668</td>
</tr>
<tr>
<td>Test sentences</td>
<td>1,798</td>
<td>1,095</td>
</tr>
<tr>
<td>Test words</td>
<td>20,287</td>
<td>10,669</td>
</tr>
<tr>
<td>New test lexical types</td>
<td>589</td>
<td>286</td>
</tr>
<tr>
<td>Types</td>
<td>615</td>
<td>360</td>
</tr>
</tbody>
</table>

Table 7.1. Make-up of the English Resource Grammar (ERG) and JACY grammars and treebanks.
Optionally discontinuous lexical items are less of a problem, as selection of each of the discontinuous parts is done using lexical types. In the case of *pick up*, the lexical entry could be as follows:

\[
pick\_up\_v1 := \text{v\_p-np\_le} \& \\
& \text{STEM < "pick" >,} \\
& \text{SYNSEM [ LKEYS [ --COMPKEY \_up\_p\_sel\_rel,} \\
& \text{KEYREL.PRED "\_pick\_v\_up\_rel" ] ] }. \\
\]

in which "pick" selects for the \_up\_p\_sel\_rel predicate, which in turn is associated with the stem "up" and lexical type \_p\_prt\_cl\_le. In terms of tag annotation, we can treat these as separate tags and hope the supertagger can model the mutual dependence between these lexical types, even if they fall outside the first order Markov assumption. For detailed statistics of the composition of the two grammars, see Table 7.1.

For morphological processing (including tokenisation and lemmatisation), we use the pre-existing systems provided with each of the grammars. In the case of the ERG, this consists of a finite state machine which feeds into lexical rules; in the case of JACY, segmentation and lemmatisation is based on a combination of the ChaSen (Matsumoto et al., 2003) segmenter and PoS tagger, and lexical rules. Thus we are able to assume that the Japanese data has been pre-segmented in a form compatible with JACY, as we are able to replicate the pre-processing that it uses.

### 7.4 Evaluation

Evaluation is based on the treebank data associated with each grammar, and a random training–test split of 10,000 training sentences and 1,798 test sentences in the case of the ERG, and 40,000 training sentences and 1,095 test sentences in the case of the JACY. This split is fixed for all models tested.

Given that the goal of this research is to acquire novel lexical items, our primary focus is on the performance of the different models at predicting the lexical type of any lexical items which occur only in the test data. Such novel lexical type taggings include novel lexemes or previously-seen lexemes occurring with a lexical type with which they have not
previously been observed. As such, we identify all unknown lexical items in the test data, relative to the training data, and evaluate according to:

**Token accuracy** \((\text{ACC}_U)\) The proportion of unknown lexical items which are correctly tagged.

**Type precision** \((\text{PRE}_C)\) The proportion of correctly hypothesised unknown lexical entries.

**Type recall** \((\text{REC})\) The proportion of gold-standard unknown lexical entries for which we get a correct prediction.

**Type F-score** \((\text{F-SCORE})\) The harmonic mean of type precision and type recall.

**Overall token accuracy** \((\text{ACC})\) is also measured across all words in the test data, irrespective of whether they represent known or unknown lexical items.

### 7.4.1 Baseline: Unigram Supertagger

As a baseline model, we use a simple unigram supertagger trained based on maximum likelihood estimation over the relevant training data, i.e. the tag \(t_w\) for each token instance of a given word \(w\) is predicted by:

\[
t_w = \arg \max_t p(t \mid w)
\]

In the instance that \(w\) was not observed in the training data, we back off to the majority lexical type in the training data.

### 7.4.2 Benchmark: fnTBL

In order to provide a benchmark for our results with the CRF models, we reimplemented the supertagger model described by Baldwin, 2005b which simply takes fnTBL 1.1 (Ngai and Florian, 2001) off the shelf and trains it over our particular training set. fnTBL is a transformation-based learner that is distributed with pre-optimised POS tagging modules for English and other European languages that can be redeployed over the task of supertagging. Following Baldwin, 2005b, the only modifications we make to the default English
POS tagging methodology are to set the default lexical types for singular common and proper nouns to \( n_{-}c\)le and \( n_{-}pn\)le, respectively, and to reduce the threshold score for lexical and context transformation rules to 1. Unlike our proposed method, the English POS tagger implementation in FN TBL has been fine-tuned to the English POS task, and includes a rich set of feature templates specific to English. Thus the FN TBL tagger, in comparison to our language neutral approach, would require significantly more effort to port to languages other than the small set for which optimised modules exist.

Note that were only able to run FN TBL over the English data, as encoding issues with the Japanese proved insurmountable. We are thus only able to compare results over the English, although this is expected to be representative of the relative performance of the methods.

### 7.4.3 Results

The results for the baseline, benchmark FN TBL method for English and our CRF-based supertagger are presented in Table 7.2 and Table 7.3, for each of the ERG and JACY
respectively. The precision, recall and f-score results are over unseen lexical types in the test set, this includes known lexical items with a novel supertag. In order to gauge the impact of the lexical features on the performance of our CRF-based supertagger, we ran the supertagger first without lexical features (CRF_{-LEX}) and then with the lexical features (CRF_{+LEX}). We report results using both a maximum a posteriori (MAP-CRF) and maximum margin (MM-CRF) training criteria, both employing the Viterbi algorithm for decoding.

The first finding of note is that the proposed model surpasses both the baseline and FNBL in the majority of cases. The exception is the ERG MM-CRF_{-LEX} model, trained without lexical features, which suffers a lower precision than the baseline, however when lexical features are included this model records the highest precision. If we look closer at these results we see that MM-CRF_{-LEX} over-predicts new lexical items, predicting 1025, while MM-CRF_{+LEX} only predicts 794. This marked difference is not present for the MAP-CRF_{±LEX} models (886 vs. 874), suggesting that the MM model is not able to make effective use of the raw word context features, but adding the lexical features solves this problem.

If we look to token accuracy for unknown lexical types, the CRF is far and away the superior method, in comparison to the baseline and benchmark, a result which is somewhat diminished but still marked for type-level precision, recall and F-score. Recall that for the purposes of DLA, our primary interest is in how successfully we are able to learn new lexical items, and in this sense the CRF appears to have a clear advantage over the other models. It is also important to recall that our results over both English and Japanese have been achieved with only the bare minimum of lexical feature engineering, whereas those of FNBL are highly optimised.

Comparing the results for the CRF models with and without lexical features (CRF_{±LEX}), the lexical features appear to have a strong bearing on type precision in particular, for JACY, and to a lesser degree, the ERG.

Looking to the raw numbers, the type-level performance for all methods is far from flattering. However, it is to be expected that the overall token accuracy should be considerably higher than the token accuracy for unknown lexical items. A breakdown of type precision and recall for unknown words across the major word classes for English is presented in
Table 7.4. Individual results when predicted new lexical types for the ERG. These results use the model with best f-score: MAP-CRF\textsubscript{+LEX}.

<table>
<thead>
<tr>
<th></th>
<th>PREC</th>
<th>REC</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>nominal (n_*)</td>
<td>0.381</td>
<td>0.551</td>
<td>0.450</td>
</tr>
<tr>
<td>adverb (av_*)</td>
<td>0.400</td>
<td>0.389</td>
<td>0.394</td>
</tr>
<tr>
<td>adjective (aj_*)</td>
<td>0.326</td>
<td>0.459</td>
<td>0.381</td>
</tr>
<tr>
<td>verb (v_*)</td>
<td>0.112</td>
<td>0.228</td>
<td>0.151</td>
</tr>
</tbody>
</table>

Table 7.5. Individual results when predicted new lexical types for the JACY grammar. These results use the model with best f-score: MM-CRF\textsubscript{+LEX}.

<table>
<thead>
<tr>
<th></th>
<th>PREC</th>
<th>REC</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>noun (n*, ordinary-nohon-n*)</td>
<td>0.932</td>
<td>0.955</td>
<td>0.943</td>
</tr>
<tr>
<td>verb (v*)</td>
<td>0.632</td>
<td>0.706</td>
<td>0.667</td>
</tr>
<tr>
<td>conjugating adjective (i*)</td>
<td>0.444</td>
<td>0.533</td>
<td>0.485</td>
</tr>
<tr>
<td>non-conjugating adjective (na*)</td>
<td>0.268</td>
<td>0.688</td>
<td>0.386</td>
</tr>
</tbody>
</table>

Table 7.4. These numbers suggest that the CRF\textsubscript{+LEX} supertagger is most adept at learning nominal, adverbs and adjectival lexical items (with an F-score of 0.450, 0.394 and 0.381, respectively), and has the greatest difficulties with verbs (with an F-score of 0.151). In the case of Japanese (Table 7.5), nouns and verbs present the least difficulty (with F-scores of 0.943 and 0.667), and conjugating and non-conjugating adjectives considerably harder (with an F-score of 0.485 and 0.386, respectively). Note that approximately two thirds of the unknown type taggings are nouns, thus it is understandable for the model to do well on this majority class.

Figures 7.1 and 7.2 show the impact of the size on the training sample on novel lexical item prediction. In these experiments a novel lexical item is defined as one that doesn’t appear in the whole training sample, thus each performance metric at each sample size is calculated against the same set of lexical items. Interestingly, as the size of the training sample is increased recall remains relatively static, while precision increases steeply. Thus there appears to be a core group of potential lexical items that the model can detect, and others that it cannot, at all sample sizes. The main impact of increasing the sample size being to reduce the number of false lexical item predictions.
When performing DLA we may wish to adjust the precision/recall trade-off of our models in order to be able to predict new lexical types for incorporation into the grammar. Such a trade-off allows the grammar developer to choose the amount of automation they desire by deciding whether they would like the model to provide a small number of highly probable type predictions, or a large number of possibly noisy predictions. In the case of the probabilistically trained models (MAP-CRF+LEX) we are able to use the forward-backward algorithm (Section 3.3.2) to calculate the marginal probability of predicting a particular tag for a given word. This method provides an alternative decoding method to the Viterbi algorithm, and allows us to threshold the marginal probabilities to restrict the prediction of new types. In Figure 7.3 we display the precision/recall trade-off for the JACY data at...
various cutoff values. For example, a cutoff of 0.8 implies that we only include a predicted new type if the model’s marginal probability for that tag is higher than 0.8. From the graph we can see that this cutoff strategy provides an effective trade-off, allowing us to achieve high precision predictions (> 0.9) while still recalling greater than half the new types in the test data. The linear nature of the precision line also confirms that the MAP training criteria has produced a model which accurately captures the tag probability distribution.

The graph for the ERG is shown in Figure 7.4. In this case we again see a nice linear precision curve, however recall is very low at high precision. This very low recall accounts for the statistically anomalous bump in the precision. These trade-off graphs suggest that it should be possible to present the grammar developer with a relatively low-noise set of
Figure 7.3. The effect of applying a cutoff value on the marginal probabilities when predicting new lexical types for the JACY grammar (MAP-CRF + LEX using forward-backward decoding).

automatically learned lexical items, minimising the work required to feed them into the lexicon proper.

7.5 Summary and Future Directions

In this chapter we have explored a method for learning new lexical items for HPSG-based precision grammars through supertagging. Our pseudo-likelihood conditional random field-based approach provides a principled way of learning a supertagger from tens-of-thousands of training sentences and with hundreds of possible tags.
We achieve state-of-the-art results for both English and Japanese data sets with a largely language-independent feature set. Our model also achieves performance at the type- and token-level, over different word classes and at multiword expression identification, superior to a probabilistic baseline and a transformation based learning approach.

Key contributions of this chapter are:

1. Using the pseudo-likelihood approach of approximating the sentence normalised CRF model with locally word normalised training we are able to train models with the very large tag sets of lexical types required for DLA.
2. We have shown that once we have defined the graphical structure and feature functions of the supertagging model, we can easily choose to train the model parameters using either a probabilistic MAP, or a maximum margin objective.

3. With generic, simple, and language independent, feature functions the CRF models achieve higher precision and recall than previous approaches. This demonstrates the superior ability of the CRFs efficient training and decoding algorithms to estimate generalisable feature weights for a large number of heterogeneous features.

4. Using a probabilistic model it is possible to threshold the marginal probabilities when predicting novel lexical types, thus allowing the grammar developer to tune the precision/recall trade-off to their particular requirements.

Thus we can conclude that discriminatively trained undirected graphical models provide a practical and flexible method for speeding the development of deep linguistic resources.
8.1 Introduction

We have shown in Chapter 7 that supertagging can be successfully used to identify lexical types for novel lexical items. In this chapter we return to the original focus of supertagging and investigate the ability of our models to disambiguate lexical types for both known and unknown words with the aim of decreasing the search space of automatic syntactic parsers (Bangalore and Joshi, 1999). By using our sequence based supertagging models to find an ordered subset of possible types for each word in a sentence, a parser can be directed to first attempt to find a spanning parse with the highest scoring types for each word, and then back off to subsequent types until a parse is found.

Previous work has demonstrated the effectiveness of the supertagging approach for improving the scalability of probabilistic parsing for strongly lexicalised grammars (Bangalore and Joshi, 1999; Clark and Curran, 2004a). In fact it is unlikely that parsers for such grammars could be efficiently implemented without a supertagging style parse reduction strategy. An additional benefit of employing a supertagging step in parsing, and treating it as a sequence labelling task, is the ability to introduce features and conditioning context which is either not available, or not practical, for a standard tree based parser.

In this chapter we will first review previous approaches to supertagging for parse reduction across a variety of grammar formalisms. We then present experiments using both CCG and HPSG data sets and a pseudo-likelihood CRF tagging model trained using
Figure 8.1. Elementary tree structures (supertags) for a simple sentence. With the correct supertags for each word, the space of possible derivations is heavily constrained (Bangalore and Joshi, 1999).

a maximum a posteriori (MAP) objective. The trained models are used as multi-taggers, in which the marginal probabilities from the forward backward algorithm are thresholded in order to return a set of possible tags for each word token.

8.2 Past Research

8.2.1 LTAG

Bangalore and Joshi, 1999 presented supertagging as a sequence labelling task for integrating statistical techniques with robust parsing models. They viewed supertagging as almost parsing, arguing that with the correct sequence of lexical types for the words of a sentence there was little for the parser to do beyond combining the types assigned. It was shown that this approach was effective for a Lexicalised Tree-Adjoining Grammar (LTAG) parser.

The LTAG formalism assigns elementary trees as the lexical types for individual words. These elementary tree structures contain all the dependencies in which the word participates and thus represent a much finer grain description of the word’s syntactic properties than regular part-of-speech (PoS) tags. Supertagging treats these trees as atomic tags, in much the same way as PoS tagging. However as these tags include rich syntactic information there are many more possible supertags per-word than PoS tags. It is this
high ambiguity of lexical types that led many to consider automatic parsing with highly lexicalised grammars impractical. However, by pre-assigning supertags to a sentence prior to parsing, in an analogous step to PoS tagging before parsing, the parse search space can be drastically reduced. Figure 8.1 shows a sentence with its LTAG supertags.

The model used by Bangalore and Joshi, 1999 is referred to as a trigram model, although it would seem most natural to consider it a second-order HMM (Section 3.1.1). In this case we see that the supertags are the HMM states and the words are observations. Thus the model performs the standard Markov decomposition of the probability of a tagging \( t \) given sentence \( w \), and we seek to find the tagging \( \hat{t} \):

\[
P(w|t) \approx \prod_{i=1}^{N} P(w_i|t_i)
\]

\[
P(t) \approx \prod_{i=1}^{N} P(t_i|t_{i-1}t_{i-2})
\]

\[
\hat{t} = \arg\max_{t} P(t) \times P(w|t)
\]

Although not explicitly stated by the authors, this argmax can be efficiently solved using the standard Viterbi algorithm extended to the second-order case.

In order to deal with the issues of sparsity in both the tag transition and word emission probabilities, sophisticated backoff and smoothing are required. For the transition probability distribution, Bangalore and Joshi, 1999 use a Katz backoff model which simply reverts to lower order transition probabilities if the second-order tag trigram probability is zero. The frequency counts for the transition model are smoothed using Good-Turing discounting. When a word/tag pair does not appear in the training corpus a leave-one-out estimate is used for the emission probability, multiplied by a distribution over word-features given the tag. These word-features include prefixes and suffixes up to length three, capitalisation and digits. A multinomial distribution, similar to that employed by a Na"ive Bayes model, is used to estimate the probability of a bag of word features given a tag.

By using an undirected discriminative model, instead of the directed generative model of Bangalore and Joshi, 1999, we are able to incorporate more sophisticated features for disambiguating rare lexical items. As the MAP trained CRF includes powerful regularisation,
Chapter 8: Parse Search Space Reduction

Figure 8.2. CCG categories (supertags) for the words in a simple sentence. The CCG categories are similar to LTAG in that they specify a considerable amount of the derivation structure, leaving only a small amount of ambiguity for the parser to resolve.

through the use of a Gaussian prior over model weights, we can include these additional features without explicitly needing to state backoff and smoothing distributions. Although we have only used a first order model here, it would be simple to extend our model to the second order case by adding clique features of tag triples. By still including features over tag pairs we would avoid the need for a backoff algorithm as the discriminative training would effectively estimate the weights of the dependent clique features. However, this additional power would come at the cost of increasing the decoding complexity to be cubic in the number of states, which is a cost the LTAG HMM model incurs as well.

8.2.2 CCG

Another strongly lexicalised formalism that can benefit from supertagging is Combinatory Categorial Grammar (CCG). As noted previously, Hockenmaier et al., 2004 described an early application of a maximum entropy tagger to extending a CCG lexicon for a symbolic system. Clark, 2002; Clark and Curran, 2004b described how supertagging can be used to radically increase the scalability of statistical CCG parsing. They automatically extract a grammar from CCGBank (Hockenmaier and Steedman, 2002; Hockenmaier and Steedman, 2007), a version of the Penn Treebank annotated with CCG dependency structures, using a log-linear parsing model. As the grammar licenses a large number of categories (1000+) the parser requires a supertagging step in order to constrain the space of possible derivations.

Parsing with CCG, like other lexicalised formalisms, is a two stage process. First, categories must be assigned to each word from its lexical entry, then these categories must be combined using the rules of the grammar to form a dependency structure spanning the whole sentence.
Chapter 8: Parse Search Space Reduction

There have been a number of approaches to parsing with CCG (Hockenmaier and Steedman, 2001; Hockenmaier, 2003; Bos et al., 2004; Hockenmaier, 2006), and additionally generation from CCG analyses (White, 2006).

Here we describe the CCG parser of Clark and Curran, 2004a which is a conditional log-linear model that is essentially a CRF over the space of all possible CCG dependency structures. The probability of a dependency structure $\pi$ given a sentence $s$ and the set of all derivations (parse trees) consistent with the dependency structure $\Delta(\pi)$, is defined as:

$$P(\pi|s) = \sum_{d \in \Delta(\pi)} P(d, \pi|s)$$

(8.2)

Two different approaches for training the model are described. The first treats a parse as a dependency structure and derivation pair, maximising the log-likelihood of the gold standard dependencies in CCGBank by summing over all derivations consistent with those dependencies. The second method approximates the log-likelihood calculation by only summing over the gold standard normal-form derivations in CCGBank, ignoring other derivations consistent with the dependency structure. For both these approaches the conditional probability of a parse $\omega$ given a sentence takes the standard log-linear form:

$$P(\omega|s) = \frac{1}{Z(s)} e^{\lambda f(\omega)}$$

(8.3)

where $Z(s)$ is the normalisation function which is the sum over all parses consistent with sentence in the parse forest. Both of these training approaches can be efficiently implemented using the inside-outside dynamic programming algorithm (a tree based version of the sequence forward-backward algorithm) with a chart parser to calculate the feature expectations.

The CCG supertagging model is based on the MEMM sequence model introduced in Section 3.2.1. This model is similar to our pseudo-likelihood model, with the significant difference that the MEMM conditions only on a node’s left neighbour, while our model conditions on both neighbours. We discuss the CCG MEMM more in Section 8.4.2 where we compare its performance with that of our model.
Investors are appealing to the Exchange Commission

**Figure 8.3.** The full CCG parsing model which is approximated by the composition of supertagging and parsing models. The bold links depict graph edges, from which features can be derived, that would not be present if only the parsing model were used.

In the CCG parsing model, both the parser and the supertagger use log-linear models with CCG categories as their state-sets. As we’ve discussed, the main motivation for the supertagging step is to improve the efficiency of parsing. However, given the similarities of the two models, it is of interest to consider how they differ and whether the supertagger could possibly be using conditioning information not available to the parsing model. The conditioning context of the supertagging sequence model is not a subset of that of the hierarchical parsing model. The supertagger provides extra graph edges between non-constituent nodes which are not present in the parse graph. In fact the combination of a sequence supertagging model and a tree parsing model approximates a single parsing model in which the leaf tags are linked in sequence. This approximated model is depicted in Figure 8.3, where the bold links represent additional conditioning context introduced by the sequence model that is not present in the parsing model. As this model is not a
tree and contains cycles at the leaves, training it directly without heavy approximation would be extremely costly. Therefore the two-stage CCG parsing model can be viewed as a decomposition of this model in which the supertagger fixes the values of the leaf nodes which the parser then builds structure on top of. For example, in Figure 8.2 the labelling of the non-constituent bigram *publication that* would have associated clique features in the supertagging model, but not the parsing model. Thus supertagging may be viewed as not just a way of improving parser efficiency, but also a method of expanding the modelling power of the parsing pipeline.

### 8.2.3 HPSG

Most work on parsing with HPSG formalisms has focussed on symbolic approaches. This has meant that little research has been conducted on the possibility of using statistical tagging techniques to constrain the parse search space. However, a number of investigators have explored the use of conditional models over the lexical types for parse ranking and coverage expansion.

Toutanova et al., investigated the use of several discriminative and generative models for parse disambiguation for the Redwoods HPSG Treebank. Included amongst these were trigram tag sequence models used to provide the probability of the lexical label sequence of a sentence given its words. These sequence models are roughly equivalent to our supertagging models, although our models operate at the lexical type level rather than on the finer grained lexical labels. However, instead of using their taggers to restrict the parse search space, Toutanova et al., using the tag sequence probabilities to rank the candidate parses produced by the parser. Of particular relevance for our supertagging models, this parse disambiguation work showed that the lexical label sequence determined approximately 50% of Redwoods parse trees. Given that we are tagging at the higher lexical type level, we can conclude that identifying the correct lexical type sequence for a sentence could dramatically reduce the parse search space for a HPSG parser.

Zhang and Kordoni, 2005 used a log-linear model to predict lexical types for unknown words on-the-fly during parsing. Their log-linear model was a simple word instance classifier that classified only the unknown words in a sentence. Although our supertagging
model is aimed at reducing the parse search space, it also predicts types for unknown words, thus also providing increased robustness. In addition, where the Zhang and Kordoni, 2005 model only makes predictions for unknown words, our model may predict a novel lexical item for a known word.

Ninomiya et al., 2006 proposed supertagging like lexical tagging models to develop simplified models of HPSG parsing. In their parsing models the probability of a parse was approximated by the probability of its lexical type sequence. It was found that extremely lexicalised parse models achieved performance comparable to models defined over the full parse structures. The supertagging model used to calculate the lexical type probabilities was a log-linear model similar to our CRF model. However, their model classified each word in the sentence independently, not treating the task as structured classification and thus not exploiting the inherent sequence nature of the lexical types.

Matsuzaki et al., 2007 built upon the work of Ninomiya et al., 2006, proposing an efficient HPSG parsing system that combines supertagging with a filtering CFG parser that approximates full HPSG parsing. The CFG is used as a post-processor to the Ninomiya et al., 2006 supertagger in order to select maybe-parsable supertag sequences. Thus the CFG can eliminate many unlikely supertag sequences and present a HPSG parser with a list of maybe parsable sequences one at a time, ranked by the probability assigned to them by the supertagger. This approach allows a radically simplified HPSG parser to be used which takes a single supertag sequence and attempts to find a parse consistent with that sequence. The parser is implemented efficiently using a deterministic shift-reduce algorithm and guided by the parse chart produced by the approximating CFG. Their results showed a significant speed up in parse time over the previous Ninomiya et al., 2006 parsing system, with only a small degradation in f-score. They found that approximately 95% of sentences could be parsed using the first-best supertag sequence returned from the CFG filtered supertagger. This work demonstrates the effectiveness of supertagging for increasing the efficiency of highly lexicalised grammar parsing. We can hypothesise that using a more sophisticated supertagger that treats the task as one of whole sentence structured classification, such as the one presented in this thesis, rather than individual word classification, further accuracy and performance improvements could be realised.
8.3 Task and Resources

In order to evaluate our pseudo-likelihood CRF’s ability to provide a useful distribution over lexical types for a parser, we have performed experiments with two heavily lexicalised grammars: HPSG and CCG. We previously introduced the HPSG treebank data (Section 7.3), which we will use again for these experiments. The CCG corpus we use to train and evaluate our model is the CCGBank (Hockenmaier and Steedman, 2002). CCGBank is derived from the Penn Treebank phrase structure trees with specific processing required for some analyses such as coordinations. As mentioned in Section 8.2.2 the spurious ambiguity present in CCG allows many derivations for a given dependency structure, so the CCGBank includes only normal-form derivations (Hockenmaier and Steedman, 2002). For our supertagging experiments we can ignore these issues as we derive our training and test data purely from the CCG categories of the derivation leaves.

For a detailed description of the corpora, and the training and test splits we have used, see Table 8.1.

8.4 Evaluation

We evaluate the HPSG model on the same training and testing split that was used in Chapter 7. Our evaluation of the CCG model follows Clark and Curran, 2004a, using CCGBank WSJ sections 02-21 for training and section 00 for testing. WSJ Section 23 is usually reserved for testing the full parsing model derived from the supertagger. In the case of CCG we follow previous works and restrict the category set to only include those that occur greater than ten times in the training corpus. Words with categories that are
not included in this set are tagged with the state UNK, hopefully allowing the model to recognise rare word/tag pairs. This results in 425 CCG categories, plus the one additional unknown tag.

As supertagging for parse search reduction is an instance of multi-tagging, rather than classification, we evaluate our models on whether the correct lexical type for a word is present in a set of predicted types returned by our model. We define a parameter $\beta$ which controls the number of tags per-word predicted by our models. All tags for a given word with probability greater than $\beta \times P(\text{best tag}|\text{word})$ are returned by the model. Thus a $\beta$ value of 1.0 means that only the most probable tag for each word returned by the model is evaluated. The smaller the value of $\beta$, the greater the average number of tags per word produced by the model, and the more likely it is that the correct tag will be present in the predicted set. However, as $\beta$ decreases less parse ambiguity is resolved by the supertagger and more work is done by the parser in order to find a maximally probable derivation.

The models evaluated use the structure and feature representation, including the lexical features, outlined in Chapter 6. We present results using the probabilistic MAP training objective only, decoding using the forward-backward algorithm to calculate marginal probabilities over tags. As the maximum margin approach doesn’t have an inherent probabilistic interpretation, producing marginals which can be thresholded to perform multi-tagging is not well motivated.

### 8.4.1 Baseline: Unigram Supertagger

As for DLA, our baseline model consists of a simple maximum likelihood unigram supertagger where the $P(t_w|\text{word})$ for each token instance of a given word $w$ is estimated by:

$$t_w = \arg\max_t p(t|w)$$

In the instance that $w$ was not observed in the training data, we back off to the majority lexical type in the training data.
Chapter 8: Parse Search Space Reduction

8.4.2 Benchmark: CCG MEMM Supertagger

In order to compare our model to previous approaches to supertagging we also include an MEMM style model similar to Clark and Curran, 2004b, when presenting results for CCG. This model is similar to our pseudo-likelihood model, where the only difference is that in the MEMM clique features are only included from a node's leftward neighbour. This approach was originally inspired by a HMM directed graphical model view of supertagging in which the conditional probability of a tag was estimated given the previous tag and the current word. However, the approach of encoding a left to right bias in an undirected graphical model seems somewhat arbitrary and the pseudo-likelihood model removes this bias by including clique features from both neighbours of a node. Figure 8.4 compares the conditioning context of the two approaches. Shaded nodes are fixed to their gold standard tags as observed in the training data, and clique pair features are extracted for the nodes in the circled conditioning context.

8.4.3 Results

We report supertagging results at a variety of values of $\beta$, the average number of categories assigned per-word (CATS), the token accuracy (ACC) and the sentence accuracy ($ACC_S$).
Table 8.2. Results of CCG supertagging at various values of $\beta$ for WSJ section 00. Note that the unigram baseline occasionally assigns the same probability to multiple tags, resulting in a value greater than one for the categories per-word when $\beta = 1$.

Table 8.3. Results of JACY supertagging at various values of $\beta$.
fact that the discriminative models specifically optimise the sequence of tags assigned in a
label, and thus highly probable tags can bias ambiguous tags towards values which result
in the most probable sentence label.

As found with previous work on CCG supertagging, very high token accuracies (above
96%) are achieved while assigning less than an average of two categories per-word. This
illustrates supertagging does significantly reduce ambiguity in CCG parsing. Note that as
the CCG tag set was restricted to frequently occurring tags, it is not possible for the model
to achieve a token accuracy of 100%.

The JACY results are displayed in Table 8.3 and Figure 8.6. Again the CRF easily
outperforms the baseline unigram model. In this case we see that extremely high token
accuracies are recorded with only a small number of categories per-word. With slightly
greater than two categories the JACY CRF supertagger achieves an accuracy of 99.9%. This
particularly strong result leads us to conclude that our pseudo-likelihood CRF statistical
supertagger has great potential to improve parser performance for the HPSG based JACY
grammar.

The result for the ERG, presented in Table 8.4 and Figure 8.7, mirror those of JACY,
although in this case the performance is not quite as impressive. This result is probably
due to the much larger type set (615 vs. 360) employed in the ERG tagger, and the smaller
training sample size.
Chapter 8: Parse Search Space Reduction

Table 8.4. Results of ERG supertagging at various values of $\beta$. 

<table>
<thead>
<tr>
<th>$\beta$</th>
<th>CATS</th>
<th>Acc</th>
<th>$\text{Acc}_g$</th>
<th>$\beta$</th>
<th>CATS</th>
<th>Acc</th>
<th>$\text{Acc}_g$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>1.00</td>
<td>0.809</td>
<td>0.257</td>
<td>1.00</td>
<td>0.903</td>
<td>0.499</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>1.32</td>
<td>0.877</td>
<td>0.391</td>
<td>1.10</td>
<td>0.932</td>
<td>0.595</td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td>2.45</td>
<td>0.956</td>
<td>0.670</td>
<td>1.54</td>
<td>0.972</td>
<td>0.780</td>
<td></td>
</tr>
<tr>
<td>0.05</td>
<td>3.30</td>
<td>0.969</td>
<td>0.739</td>
<td>1.94</td>
<td>0.982</td>
<td>0.846</td>
<td></td>
</tr>
<tr>
<td>0.01</td>
<td>7.52</td>
<td>0.985</td>
<td>0.863</td>
<td>4.13</td>
<td>0.993</td>
<td>0.935</td>
<td></td>
</tr>
<tr>
<td>0.005</td>
<td>10.25</td>
<td>0.988</td>
<td>0.887</td>
<td>6.34</td>
<td>0.995</td>
<td>0.956</td>
<td></td>
</tr>
<tr>
<td>0.001</td>
<td>16.14</td>
<td>0.990</td>
<td>0.907</td>
<td>20.38</td>
<td>0.998</td>
<td>0.979</td>
<td></td>
</tr>
</tbody>
</table>

Figures 8.8 and 8.9 show how token accuracy is affected by training sample size. As would be expected, both graphs show a steep initial improvement in performance as the size of the training sample is increased. However, improvements appear to be leveling off at approximately 10,000 sentences, suggesting that we might only see 1–2% improvement in the ERG model if it was scaled to the same training sample size as the JACY model. Also of note is that the ERG model under-performs the JACY model by 2% at 10,000 sentences, suggesting that the ERG task is indeed more difficult.

8.5 Parsing Experiments

To assess the potential of our supertagging model to improve parsing performance we need to determine the size of the reduction in search space achieved by using its supertag sequences to constrain the parser, and the cost of this reduction in terms of gold analyses missed. In order to measure this effect we have devised two experiments using the PET parsing system\(^1\) (Callmeier, 2000) and the ERG test data set used in our supertagging experiments. PET is an efficient unification-based grammar parser for the DELPH-IN family of grammars. The first experiment uses PET to produce a list of all analyses consistent with the grammar for each test sentence. This list is then filtered to retain only those analyses consistent with the multi-supertag sequence produced by the supertagger for that test sentence. In the second experiment we seed the parse chart with the lexical types

\(^1\)http://wiki.delph-in.net/moin/PetTop
output from the supertagger for each token, thus restricting the parser to only generating analyses consistent with these types. Due to time and resource limitations we were unable to perform these experiments for the JACY data set.

### 8.5.1 Parse Filtering

Using PET to parse the ERG test data set we are able to obtain a list of all analyses consistent with the grammar for each test sentence. The parser was unable to find analyses for a number of the test sentences, and these sentences were ignored in the following experiments. In addition, due to the possibility that the parser may hit resource constraints...
before enumerating all consistent analyses, the parser output for a number of the sentences didn’t contain the gold standard supertag sequence. Table 8.5 summarises these statistics.

For each test sentence token, for each value of $\beta$, the supertagger outputs a list of possible lexical types. We filter out any analysis produced by the parser that contains a token/type tagging not present in the supertagger output. By doing so we are able to gain an indication of the proportion of unlikely analyses that the supertagger is able to remove from consideration by the parser. The results over the 1377 successfully parsed ERG test sentences are presented in Figure 8.10. With a $\beta$ value of $1 \times 10^{-5}$ the supertagger produces many lexical types for each token, thus only a small number of analyses are filtered (approx. 13%), and almost all the gold standard sequences are retained. As the $\beta$ value is increased

---

**Figure 8.8.** ERG tag accuracy versus training sample size
to 1, the total number of analyses not consistent with the supertagger output drops off sharply, while the number of gold standard analyses missed increases much less sharply. The effectiveness of the supertagger for filtering non-gold analyses can be seen from the results at $\beta = 0.01$, 60% of the analyses output by the parser are discarded, with only 3% of those being gold-standard. Although this experiment is only indicative, and doesn’t represent a practical use case, we can conclude that the supertagger is effective at reducing the space of ERG analyses with only a small decrease in gold standard performance.

Figure 8.9. JACY tag accuracy versus training sample size
Chapter 8: Parse Search Space Reduction

<table>
<thead>
<tr>
<th>Test sentences</th>
<th>Successfully parsed</th>
<th>Gold analyses found</th>
</tr>
</thead>
<tbody>
<tr>
<td>1798</td>
<td>1377</td>
<td>1252</td>
</tr>
</tbody>
</table>

**Table 8.5.** Parsing the test set with PET.

<table>
<thead>
<tr>
<th>$\beta$</th>
<th>Sentences</th>
<th>Analyses/sentence</th>
<th>Coverage</th>
<th>Seconds/sentence</th>
<th>Passive edges/sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>729</td>
<td>9.48</td>
<td>0.73</td>
<td>0.02</td>
<td>161.7</td>
</tr>
<tr>
<td>0.5</td>
<td>729</td>
<td>10.86</td>
<td>0.81</td>
<td>0.03</td>
<td>181.1</td>
</tr>
<tr>
<td>0.1</td>
<td>729</td>
<td>39.66</td>
<td>0.91</td>
<td>0.04</td>
<td>254.2</td>
</tr>
<tr>
<td>0.05</td>
<td>729</td>
<td>114.18</td>
<td>0.93</td>
<td>0.05</td>
<td>311.3</td>
</tr>
<tr>
<td>0.01</td>
<td>729</td>
<td>3733.76</td>
<td>0.97</td>
<td>0.17</td>
<td>687.9</td>
</tr>
<tr>
<td><strong>ALL</strong></td>
<td><strong>694</strong></td>
<td><strong>70.06</strong></td>
<td><strong>0.79</strong></td>
<td><strong>0.11</strong></td>
<td><strong>555.8</strong></td>
</tr>
</tbody>
</table>

**Table 8.6.** ERG parse chart seeding statistics. Analyses per sentence are the number of parse analyses produced for a given value of $\beta$, we expect that the more lexical items supplied to the parser, the more analyses it should be able to find; Coverage is the proportion of sentences successfully parsed; and passive edges are parse structures representing a valid sub-analysis spanning words in the input, this provides a direct indication of the size of the space searched in order to produce a valid analysis spanning the entire input.

### 8.5.2 Chart Seeding

To directly assess the ability of the supertagger to reduce the search space of the parser we seed the PET chart with the lexical type supertags predicted, such that the parser can only explore the space of parses consistent with these types. An unfortunate side effect of this technique is that the computational overhead of the parser seems to be increased and the parser fails to find a parse for some sentences. Although the cause of this is not clear, it is assumed to be an implementation issue as the parser was not originally intended to be used in this way. The number of successful parses decreases as the value of $\beta$ decreases, i.e. as the parser is given more options to choose from it seems more likely to run out of resources. In order to compare the parsing results across different values of $\beta$ we restrict the comparisons to the intersection of the sentences successfully parsed at all values (a total of 729 sentences).

Table 8.6 displays the results for this experiment. At each value of $\beta$ we measure the average number of analyses per sentence, the coverage over the inputs (sentences parsed
Chapter 8: Parse Search Space Reduction

Figure 8.10. ERG parse filtering: proportion of gold vs. all parses retained after filtering out all those not consistent with the output of the supertagger.

... successfully, irrespective of whether the gold parse was found), the average number of seconds taken to parse each sentence, and the average number of passive edges per parse chart. The **ALL** result is that using the original grammar and lexicon, rather than the supertagging output, using the standard parsing mode. Clear trends from these results are that as the value of \( \beta \) is increased (decreasing the number of lexical types per word seeded in the chart), the coverage, parse time and number of passive edges decrease. We see that at low values of \( \beta \) the parse coverage is considerably higher than using the original grammar. This is due to the fact that the supertagger assigns a lexical type to every word in a sentence, while many words, or word/type pairings, may not be present in the original grammar. This demonstrates that using the supertagger to restrict the parse search space...
is effective in reducing this space significantly, while at the same increasing robustness. Combining this result with that of the parse filtering experiment, we can conclude the our CRF supertagging model has great potential for improving HPSG parsing performance.

8.6 Summary and Future Directions

In this chapter we have explored a method for learning a multi-tagger which reduces the search space required to be explored by parsers for the heavily lexicalised grammars of HPSG and CCG. We achieve promising results for both English and Japanese data sets with a largely language-independent feature set.

Key contributions of this chapter are:

1. We have introduced the idea that the combination of a sequence supertagger and tree parser serve to approximate a single intractable parsing model that links its leaf nodes. This conceptual view of supertagging allows us to consider the possibility of supertagging to improve overall parser performance, rather than to just reduce the search space.

2. Using the pseudo-likelihood approach of approximating the sentence normalised CRF model with locally word normalised training we are able to train models with the very large tag sets required for supertagging.

3. Where previous approaches have used heavily engineered features sets tailored to a specific formalism, we achieve high performance using a generic feature set. Such a grammar and language neutral approach maximises the portability of our models by minimising the work required by parser developers when moving across different formalisms and languages.

4. We have demonstrated that the MEMM model previously proposed for CCG supertagging under-performs the CRF model.

5. By experimenting with seeding the input of the PET parser and the ERG, and filtering its output, we have shown that the supertag sequences produced by our model are
effective at restricting the parse search space while retaining a large proportion of the gold analyses.
Part IV

Conclusions and Further Work
Chapter 9

Conclusion

This thesis has described and evaluated conditional random field (CRF) (Lafferty et al., 2001) structured classification models for two multilingual natural language processing (NLP) tasks: word alignment for statistical machine translation, and language independent supertagging for lexicalised grammars. In structured classification we seek to identify regularities in the classification labels, such as the sequence nature of sentential tagging, in order to allow a tractable maximisation over all possible labels. CRFs have previously been applied to various NLP structured sequence classification tasks (McCallum and Li, 2003; Sha and Pereira, 2003; Peng and McCallum, 2004), demonstrating superior performance over previous standard techniques.

The two systems described in this thesis demonstrate additional flexibility of CRF models. The first of these, word-alignment, aims to find a translation equivalence mapping from words in an input foreign sentence to words in that sentence’s English translation. Such word-alignment mappings are the source of extracted translation tables at the core of modern statistical translation models (Koehn et al., 2003). We applied CRFs to this task by tagging each word in the input sentence with an index into the foreign sentence. In the second task, supertagging for highly lexicalised grammars, we map the large set of lexical types (or categories) in a lexicalised grammar to the tags of a sequential CRF model. Using this approach we are able to suggest new lexical items for inclusion into a precision grammar by identifying novel word and type taggings not previously seen in the training data (deep lexical acquisition). Secondly, we are able to suggest a restricted set of highly
probable types for each word in an input sentence. Such a set can be used by a parsing model to restrict the search space over possible parse trees to those that are consistent with the output of the supertagging model, reducing parse time by removing unlikely syntactic analyses from consideration.

In Part II of this thesis we described the application CRFs to word-alignment for statistical machine translation (SMT). Chapter 4 introduced SMT and motivated the application of discriminative models to this task, one which until recently has been dominated by generative approaches. Next we defined our novel CRF structure which models an alignment as a sequence of word indexes from a source sentence into a target sentence. The performance of a large number of possible feature types was evaluated on a trial alignment data set from the French-English Canadian Hansards parallel corpus. Unlabelled corpus statistics extracted from collocations and generative models were incorporated as real valued features, in addition to a range of orthographic and linguistically inspired features. Finally we evaluated and analysed our model using the most promising features identified by this feature engineering. The key contributions and findings of this chapter were:

- Our word-alignment model is described independently of a fixed label set, using features and their weights to determine the strength of alignment for words in a pair of aligned sentences. This novel model configuration abstracts away from assigning tags from a fixed set to nodes in a graph, and instead focuses on maximising the features weights of the correct word-alignment mapping.

- We developed models that used training objective functions that optimised both probabilistic maximum likelihood and maximum margin criteria. The probabilistic models out-performed the maximum margin models by a small margin, while the flexibility of the latter’s loss functions allowed the tuning of the recall and precision of the resultant alignments.

- The CRFs ability to incorporate a diverse range of feature types defined over both word pairs and the alignment sequence, plus its efficient and optimal training and decoding algorithms, lead to the production of higher quality word-alignments than the previous benchmark generative system.
• Real valued features derived from IBM Model 1 perform better than those derived from Dice collocation statistics. By incorporating features extracted from an unsupervised model our approach can be interpreted as semi-supervised in nature.

• Features indicating Model 4 predictions give a significant boost to performance.

• Many promising orthographic features appear too sparse to estimate from a small word-aligned training set.

• Penn. Treebank style part-of-speech tag features contribute positively to performance, overcoming some of the sparseness issues with word based features.

• Evaluations on a Romanian-English parallel corpus indicated that dictionary based features could be of use, if the dictionary is large enough.

In Chapter 5 we described a series of experiments designed to assess the impact of our high quality CRF produced word-alignments on a state-of-the-art phrase-based SMT system. We first reviewed previous works investigating the impact of word-alignment performance on translations produced by SMT systems derived from those alignments, concluding that no clear link has been established between alignment metrics and translation metrics such as BLEU. In our experiments we aligned the Canadian Hansards Corpus using our CRF models and used them to train a phrase-based SMT decoder. Evaluations of these translation systems on a test set showed that the CRF based alignments achieved higher BLEU scores than a system derived from the benchmark GIZA++ alignment system. We then described the use of a biased loss function in order to produce alignments with a range of recall and precision values. Using these alignments we investigated whether alignment precision, recall or f-score correlated with translation BLEU. However, as for many previous works, no link could be established. Key findings of this chapter were:

• When our high quality CRF alignments were used in the training of full translation systems, we showed that they result in an increase in translation accuracy, when measured using the automatic BLEU translation metric. These increases are statistically significant at the 95% level for the maximum-margin based alignment models.
Chapter 9: Conclusion

- We described and evaluated a maximum-margin variant of the standard minimum error rate training (MERT) for SMT models which aimed to introduce principled regularisation into the estimation of the translation model parameters. Our variant outperformed the MERT baseline on a number of experiments, although no statistically significant increases were recorded.

- By creating a loss-function that could treat aligned and unaligned words differently we were able to develop models that either exhibited high recall, high precision, or a balance of the two. However no clear link between these metrics and translation performance could be found.

In Part III we described and applied a pseudo-likelihood CRF for supertagging which approximates global sentence level CRF training. Chapter 6 introduced the task of supertagging for highly lexicalised grammars and proposed the pseudo-likelihood approximation to full CRF training in order to ameliorate the scaling issues caused by the large tag sets associated with this task. By breaking apart the sequence graphs into cliques, and treating each clique as independent of the others, training complexity was reduced from quadratic to linear in the number of supertags. A compact language and grammar formalism independent feature set was described which aimed to adequately model the lexical, morphological and sequential nature of the supertagging task, while minimising the overhead required by the grammar developer to port the system to new languages.

Chapter 7 described the application of this model to deep lexical acquisition (DLA). In DLA supertagging we seek to detect novel lexical items for incorporation into a grammar. By using our pseudo-likelihood CRF model to label sentences not seen in the training data new lexical type taggings of a lexeme can be extracted for consideration for inclusion into the grammar. Findings of this work are:

- The pseudo-likelihood CRF approximation is able to scale to tens of thousands of sentences and hundreds of tags.

- We achieve type precision and recall well in excess of the previous benchmark transformation based learning approach.
Our language independent features proved effective in modelling the supertagging task. Additional lexical features parameterised by a languages character set improved performance of the base set of features.

The marginal probabilities output by the probabilistic model provide a confidence estimate of proposed new lexical items. By thresholding these marginal probabilities the grammar developer can be presented with either a small high precision set, or a large high recall set, of predicted items for consideration for inclusion into the grammar.

In Chapter 8 we described the application of our pseudo-likelihood CRF model to supertagging for parse search space reduction. In this task the supertagging model was used as a precursor to full syntactic parsing. The model predicts a limited set of possible lexical types for each word in a candidate sentence. This restricted set can then be used by a parser to prune any parse analyses that they aren’t consistent with, thus reducing the space of parse trees to be considered. We described experiments on two lexicalised grammar formalism: Head Driven Phrase Structure Grammar (HPSG) and Combinatory Categorial Grammar (CCG), demonstrating performance improvements over the previous maximum entropy Markov model (MEMM) approach. Contributions of this work are:

- The pseudo-likelihood CRF produces improved results in comparison to the benchmark MEMM. These improvements are also apparent in the marginal probabilities produced by the two models when their performance as multi-taggers are compared.

- Our simple language independent feature set achieves high performance on both the HPSG and CCG experiments. In comparison to previous approaches utilising heavily engineered feature sets, our model reduces the overhead of developing supertagging models for new languages or grammar formalisms.

- Experiments with seeding the input of the PET parser, and filtering its output, have shown that the supertag sequences produced by our model are effective at restricting the parse search space while retaining a large proportion of the gold analyses.
9.1 Extensions and Further Research

There are many avenues for further work related to the research presented in this thesis, both within the application areas of word-alignment and supertagging, and more generally for the approaches and models discussed. Possible opportunities for further research include:

1. In this thesis we have demonstrated the flexibility of sequence based CRF models. Recent works have applied CRFs with other graphical structures, for example tree structured models (Cohn and Blunsom, 2005; Clark and Curran, 2004a) and multi-layer graphs for simultaneous tagging and chunking (Sutton et al., 2004). We see potential to extend our word-alignment model to more complex graphical structures. A lattice of binary indicator variables, one for each possible source-target alignment, could be used to directly model many-to-many alignments. Such an approach would provide additional modelling power at the expense of complexity. An extension of the pseudo-likelihood approach, described in Part III of this thesis, to a lattice model could alleviate some of the complexity issues.

2. In Chapter 5 we showed that the high precision sparse word-alignments produced by our CRF model lead to very large phrase tables when using the standard phrase extraction heuristic. By tailoring a heuristic for the sparse nature of the CRF alignments we may be able to produce smaller, higher quality, phrase tables. Such a heuristic may restrict the number of unaligned words in an extracted phrase, or whether they occur at the start, end, or within an aligned phrase.

3. Our evaluation in Chapter 5 employed a phrase-based translation model. Recent work on syntax based SMT also depends upon an automatically word-aligned corpus (Galley et al., 2004). We hypothesise that the more informative tree fragments extracted from the word-alignments would benefit from higher precision alignments. In addition, it would be possible to directly incorporate features based upon syntactic information from an automatic parser into the word-alignment model.
4. Our word-alignment CRF made heavy use of real valued translation features derived from unsupervised models, trained on a large sentence-aligned parallel corpus. The pair-clique structure of our model would also allow the incorporation of features over word pair to word pair translations, in addition to the currently include word to word features. It should be possible to design and train an unsupervised model, or a simple collocation model, that could supply translation scores over these multi-word translation pairs, thus increasing the ability of the model to align them.

5. The HPSG lexical types, and CCG categories, were treated by our supertagging model as atomic tags. However, both exhibit significant internal structure that may be usefully exploited by the model. A simple grammar formalism dependent approach would be to define features over the sub-components of the tags. For example, we could define a clique feature that was active when a CCG tag expecting an N on its left was paired with a tag on its left that could provide that N. A more advanced approach would be to treat the problem as a multi-layer tagging task (Sutton et al., 2004). In the case of HPSG, we could first label a lexical type as a noun and then tag its countability. Such an approach would allow the definition of specific feature sets for each layer, hopefully improving generalisation.

6. Previous supertagging research (Clark and Curran, 2004b; Bangalore and Joshi, 1999) have used second order Markov sequences for their models. Our model could easily be extended to a second order model at no greater cost to complexity than those previous models. In addition, our word-alignment model could also be extended to a second order model. Such an extension would allow the direct modelling of phrase alignments up to a length of three words. However, a significantly larger word-aligned corpus would be required to estimate feature weights over such phrasal alignments, as they would be very sparse in our corpus of a few hundred sentences.
Bibliography


McCallum, A. and Li, W. (2003). Early results for named entity recognition with condition-
tional random fields, feature induction and web-enhanced lexicons. In Proc. of the 3rd
International Conference on Human Language Technology Research and 4th Annual
Meeting of the NAACL (HLT-NAACL 2003), pages 188–191, Edmonton, Canada.


Proc. of HLT-NAACL 2003 Workshop, Building and Using Parallel Texts: Data Driven
Machine Translation and Beyond, pages 1–6, Edmonton, Canada.

Proc. of the 2005 Conference on Empirical Methods in Natural Language Processing
(EMNLP-2005), pages 81–88, Vancouver, Canada.

Moore, R. C., Yih, W.-t., and Bode, A. (2006). Improved discriminative bilingual word
alignment. In Proc. of the 44th Annual Meeting of the ACL and 21st International Con-
ference on Computational Linguistics (COLING/ACL-2006), pages 513–520, Sydney,
Australia.

Proc. of the 2nd Annual Meeting of the North American Chapter of Association for
Computational Linguistics (NAACL2001), pages 40–7, Pittsburgh, USA.

on Empirical Methods in Natural Language Processing (EMNLP-2006), pages
155–163, Sydney, Australia.

NIST (2001). Automatic evaluation of machine translation quality using n-gram co-
ocurrence statistics. technical report, NIST.

Templates. PhD thesis, Computer Science Department, RWTH.


of the 41st Annual Meeting of the ACL, pages 160–167, Sapporo, Japan.


