Exploiting non-linear probabilistic models in natural language parsing and reranking

Titov, Ivan

UNIVERSITÉ DE GENÈVE FACULTÉ DES SCIENCES

Département d' informatique

Professeur Christian Pellegrini
Docteur James Henderson

Exploiting Non-linear Probabilistic Models in Natural Language Parsing and Reranking

THÈSE

présentée à la Faculté des sciences de l'Université de Genève
pour obtenir le grade de Docteur ès sciences, mention informatique

par

Ivan Titov

de

St. Petersbourg (Russie)

Thèse N^o 3943

GENÈVE
ReproMail
2008

UNIVERSITÉ DE GENÈVE

FACULTÉ DES SCIENCES

Doctorat ès sciences mention informatique

Thèse de *Monsieur Ivan TITOV*

intitulée :

" Exploiting Non-linear Probabilistic Models in Natural Language Parsing and Reranking "

La Faculté des sciences, sur le préavis de Messieurs Ch. PELLEGRINI, professeur ordinaire et directeur de thèse (Département d'informatique), J. HENDERSON, docteur et co-directeur de thèse (University of Edinburgh - School of Informatics – Edinburgh, United Kingdom), M. JOHNSON, professeur (Brown University – Department of Cognitive and Linguistic Sciences – Providence, U.S.A.), Mesdames M. HILARIO, docteur (Département d'informatique), et P. MERLO, docteur (Faculté des Lettres – Département de linguistique), autorise l'impression de la présente thèse, sans exprimer d'opinion sur les propositions qui y sont énoncées.

Genève, le 29 janvier 2008

Thèse - 3943 -

Le Doyen, Jean-Marc TRISCONE

[illegible]

N.B. - La thèse doit porter la déclaration précédente et remplir les conditions énumérées dans les "Informations relatives aux thèses de doctorat à l'Université de Genève".

Nombre d'exemplaires à livrer par colis séparé à la Faculté : - 7 -

Abstract

Since mid-90s research in natural language parsing has been focused on statistical methods. Until very recently, almost all of these models have been linear (or log-linear), i.e. the score assigned to a parse tree could be represented as a sum of scores of structural features of the tree. Even though values of model parameters of linear models are determined by learning algorithms, in most of the approaches model designers have to perform extensive manual selection of predictive structural features before applying the learning algorithms. These models are expensive to create and difficult to port to new languages, grammar formalisms or domains. If the model designer omits an important predictive feature, the resulting model will not be able to discover this statistical dependency and the model accuracy will be diminished. All these problems motivate the need for methods which automatically induce features for parsing problems, which in turn implies using non-linear models. Only recently have such non-linear probabilistic models started to become popular. These include latent variable models which augment non-terminal symbols of Probabilistic Context Free Grammars (PCFGs) with latent variables (Kurihara & Sato, 2004; Matsuzaki et al., 2005; Prescher, 2005; Petrov et al., 2006; Liang et al., 2007), neural network models (Henderson, 2003; Henderson, 2004) and Incremental Sigmoid Belief Networks (ISBNs) (Titov & Henderson, 2007c).

In this thesis we propose techniques which exploit non-linear probabilistic models for parsing:

- *discriminative reranking:* learning discriminative rerankers which use a latent space induced by a model to improve the model accuracy;

- *domain adaptation:* use of this latent space as a compressed representation of the initial feature space to adapt the parser to new domains;

- *improving approximation accuracy*: accurate and tractable variational approximations for ISBNs, which are applicable to a variety of natural language parsing tasks;

- *improving accuracy at decoding time*: Bayes-risk minimization approaches which do not place any restrictions on model structure and therefore are especially suited to non-linear methods.

Though the proposed approaches are general, we focus in this thesis on a recently proposed class of graphical models, ISBNs (Titov & Henderson, 2007c), which are latent variable models inspired by the neural network model (Simple Synchrony Networks, SSNs) (Henderson, 2003). Exact inference in ISBNs is not tractable. We formally show that the computation of the SSNs model is equivalent to the variational approximation for ISBNs under certain constraints. Relaxing these constraints we build a more accurate but still tractable variational approximation to the ISBN model. Importantly, the introduced variational approximation is shown to significantly outperform the neural network model on the standard phrase-structure parsing benchmark - parsing of the Penn Treebank WSJ corpus (Marcus et al., 1993). We also construct an ISBN model for multilingual dependency parsing, where it achieves state-of-the-art performance with only minimal feature engineering.

Unlike most of the state-of-the-art methods for natural language parsing, all the successful latent variable models for parsing, including the one considered in the thesis, are generative. Unfortunately, construction of accurate fully discriminative latent variable models for parsing is an open problem. In this thesis we instead propose to exploit a latent space induced by a generative latent variable model in discriminative training. A discriminative model linear in this latent space is trained to correct the score predicted by the generative latent variable model. We propose two different approaches for induction of this latent space, which can be regarded as generalizations of the Fisher kernel (Jaakkola & Haussler, 1998) and the TOP kernel (Tsuda et al., 2002a) to parse reranking. We call both of these kernels *data-defined* because mapping to their feature space is defined by the training data, as the mapping function to the feature space is dependent on latent variable model parameters estimated from the training data. We demonstrate that data-defined kernels not only allow us to improve the accuracy of a latent variable model in the standard WSJ parsing task but also can be used to port a parser to a new domain. We obtain significant improvement from our adaptation methods when using them to port the parser from the WSJ corpus domain to individual chapters of Brown corpus (Francis & Kucera, 1982). The proposed approach is general and can be applied to virtually any non-linear probabilistic model.

Another advantage of probabilistic models compared to arbitrary discriminative classifiers is that they are directly trained to approximate probabilities. This can be used not only in language processing pipelines, but also allows us to modify the decoding criterion. Instead of attempting to select the most probable structure, the model can predict a structure which minimizes expected loss. Loss minimization, or minimum Bayes risk (MBR) decoding, was previously studied in the context of PCFGs (Goodman, 1996) with a linear loss function (labeled constituent recall measure), but there was no attempt to apply it to more complex non-linear models and different loss functions. In this thesis we explain how MBR decoding can be implemented with an arbitrary statistical model and different loss functions both for phrase-structure and dependency parsing. The obtained results suggest that even when the loss function used in MBR decoding is only weakly related to the loss function used in evaluation, results are still significantly improved with respect to standard Maximum A-posteriori Probability (MAP) decoding. This conclusion makes the use of MBR decoding attractive for natural language parsing tasks, where the true loss function is not usually known or cannot be automatically computed.

Résumé de la Thèse

Dès la moitié des années 90, les recherches dans le domaine de l'analyse syntaxique des langues naturelles se sont focalisées sur l'utilisation de méthodes statistiques. Jusqu'à récemment, la plupart de ces modèles ont été linéaires (ou log-linéaires), où le score assigné à une analyse syntaxique est obtenu en sommant les scores assignés aux traits structuraux de l'arbre de cette analyse. Bien que les paramètres des modèles linéaires soient estimés par des algorithmes d'apprentissage, les concepteurs de modèles linéaires doivent effectuer, dans la plupart des cas, une importante sélection manuelle des traits prédictifs structuraux avant de pouvoir appliquer un algorithme d'apprentissage. De tels modèles exigent des développements coûteux et sont difficiles à adapter à de nouvelles langues, à de nouveaux formalismes grammaticaux ou à de nouveaux domaines linguistiques. Si le concepteur d'un modèle linéaire omet de spécifier un trait prédictif pertinent, alors le modèle qui en résulte ne sera pas en mesure de capturer telle ou telle dépendance statistique et l'exactitude du modèle s'en ressentira. Ces problèmes motivent la nécessité de développer des méthodes qui induisent les traits prédictifs de manière automatique. Un tel développement exige, à son tour, l'utilisation de modèles non-linéaires. Ce n'est que récemment que de tels modèles non-linéaires probabilistes ont commencé à être utilisés. Parmi ces modèles, on trouve les grammaires probabilistes indépendantes du contexte avec des symboles non-terminaux cachés (Kurihara & Sato, 2004; Matsuzaki et al., 2005; Petrov et al., 2006; Liang et al., 2007), les réseaux de neurones artificiels (Henderson, 2003; Henderson, 2004) ainsi que les réseaux de croyance incrémentaux et sigmoïdaux (Incremental Sigmoid Belief Networks, ISBNs) (Titov & Henderson, 2007c).

Dans cette thèse de doctorat, nous proposons plusieurs techniques d'analyse syntaxique qui exploitent des modèles probabilistes non-linéaires:

- ré-analyse discriminative: apprentissage de ré-analyseurs discriminatifs (rerankers) utilisant l'espace latent induit par un modèle dans le but d'améliorer les performances du modèle;
- adaptation à un domaine: utilisation de cet espace caché pour adapter l'analyseur statistique à un domaine particulier;
- amélioration de l'exactitude des approximations : approximations variationnelles précises et efficacement calculables pour les ISBNs, appliquées à de nombreuses tâches d'analyse des langues naturelles;

- augmentation de la précision du décodage : approches basées sur la minimisation du risque de Bayes qui n'impliquent aucune contrainte sur la forme du modèle et qui sont ainsi particuliérement intéressantes pour être utilisées avec les méthodes non-linéaires.

Bien que les approches que nous proposons dans cette thèse soient générales, nous accordons une attention particulière aux ISBNs qui forment une classe de modèles graphiques (Titov & Henderson, 2007c). Ces modèles, développés récemment, sont des modèles à variables cachées inspirés des réseaux de neurones appelés SSNs (réseaux simples synchronisés, Simple Synchrony Networks) (Henderson, 2003). Le problème de l'inférence dans les modèles ISBN n'est pas efficacement calculable. Formellement, nous démontrons que les calculs effectués par le modèle SSN sont équivalents à l'approximation variationnelle, sous certaines contraintes, des modèles ISBN. En relaxant ces contraintes, nous construisons une approximation variationnelle des modèles ISBN non seulement plus exacte mais également plus efficacement calculable. Ce qui est important, c'est que nous pouvons montrè que l'approximation variationnelle que nous avons introduite atteint un niveau de performance significativement supérieur à celui obtenu par le modèle SSN pour la tâche standard d'analyse syntaxique du corpus Penn Treebank WSJ (Marcus et al., 1993). Nous avons également développé un modèle ISBN pour l'analyse syntaxique multilingue des dépendances syntaxiques qui atteint un niveau de performance égal à celui de l'état de l'art et qui ne requiert qu'un minimum de dévelopments techniques.

Contrairement à la plupart des méthodes de l'analyse syntaxique des langues naturelles, tous les modèles à variables cachées, y compris ceux examinés dans cette thèse, sont des modèles génératifs. Malheureusement, la construction de modèles à variables cachées performants et entièrement discriminatifs pour l'analyse syntaxique reste un problème ouvert. Dans cette thèse, plutôt que de chercher à développer un modèle discriminatif, nous proposons d'exploiter l'espace latent induit par un modèle génératif à variables cachées pour réaliser un apprentissage discriminatif. Le modèle discriminatif qui, dans cet espace latent, est linéaire, est entraîné dans le but de ré-évaluer et de corriger les scores calculés par le modèle génératif à variables cachées. Deux approches distinctes sont proposées pour la construction de cet espace caché qui peuvent être considérées comme des généralisations du noyau de Fisher (Jaakkola & Haussler, 1998) et du noyau TOP (Tsuda et al., 2002a) au problème de ré-analyse des arbres syntaxiques. Ces deux noyaux, appelés noyaux dérivés des données, ont un espace de traits qui est défini sur la base des données d'entraînement: la fonction qui renvoie à cet espace de traits dépend des paramètres du modèle à variables cachées estimés sur la base des données d'entraînement. Nous démontrons que ces noyaux dérivés des données permettent non seulement d'améliorer la précision du modèle à variables cachées pour la tâche standard d'analyse syntaxique mais également d'adapter un analyseur syntaxique à un nouveau domaine. Notre méthode d'adaptation nous a permis d'obtenir des améliorations significatives pour la tâche d'adaptation d'un analyseur du corpus Penn Treebank WSJ au corpus Brown (Francis & Kucera, 1982). L'approche proposée ici est générale et peut être appliquée à quasiment n'importe quel modèle probabiliste non-linéaire.

Un avantage supplémentaire des modèles probabilistes sur les classifieurs discriminatifs est qu'ils sont directement entraînés pour approximer des probabilités. Cette propriété peut être exploitée non seulement dans l'enchaînement de processus de traitement linguistique, mais aussi pour modifier les critères de décodage. Au lieu de retourner la structure la plus probable, un modèle probabiliste est capable de prédire la structure qui minimise l'espérance d'une fonction de perte. La minimisation d'une fonction de perte ou la minimisation du risque bayésien (Minimum Bayes Risk decoding, MBR) ont été précédemment étudiées pour les grammaires probabilistes indépendantes du contexte (Goodman, 1996) où la fonction de perte est une fonction linéaire des pertes qui sont définies par la mesure de rappel d'un constituant. Toutefois, aucune tentative n'a été précédemment faite pour appliquer une telle minimisation à des modèles non linéaires plus complexes et à des fonctions de pertes plus variées. Dans cette thèse, nous expliquons comment le décodage MBR peut être appliqué à un modèle statistique quelconque et comment intégrer les fonctions de perte pour l'analyse des constituants et des dépendances syntaxiques. Les résultats obtenus suggèrent que, même si la fonction de perte utilisée lors du décodage MBR n'est que faiblement reliée à la fonction de perte utilisée lors de l'estimation des paramètres, les analyses obtenues sont significativement plus précises que celles obtenues par un décodage standard basé sur la probabilité a posteriori maximale (MAP). Cette conclusion rend l'utilisation du décodage MBR attrayante pour les tâches d'analyse syntaxique des langues naturelles, où la véritable fonction de perte est généralement inconnue.

Acknowledgments

I am infinitely grateful to James Henderson, my thesis supervisor, for introducing me to the field, encouragement, sound advice and a lot of great ideas. I am also grateful to Christian Pellegrini, my another thesis supervisor, for giving me freedom to satisfy my curiosity and ensuring that all the administrative problems are solved fast and flawlessly. I am indebted to Paola Merlo for many interesting insights, critical and constructive views on my work and encouraging me to see a bigger picture behind technical details. Also I would like to thank other members of my thesis committee, Mark Johnson and Melanie Hilario, for excellent suggestions and thought-provoking questions.

I wish to thank all my student and post-doc colleagues: Frederick Ehrler, Jee-Hyub Kim, Alexandros Kalousis, Serhiy Kosinov, Levi Lucio, Gabriele Musillo, Julien Prados, Adam Woznica, Antonio Jimeno-Yepes and many others for fun and stimulating environment.

Lastly, but most importantly, I would like to thank the most important people for me - my parents and my wife Tatiana for their infinite support, encouragement and an incredible amount of patience.

Contents

Chapter 1

Introduction

Natural language parsing is a fundamental problem of natural language processing considered for decades in the fields of linguistics, artificial intelligence and psychology. The parsing problem is a task to learn a mapping from sequences of words to trees which represent syntactic structure of a sentence. Though it was long believed that the use of syntactic information is beneficial for a wide range of applications (Collins, 1999), only recently syntactic structures provided by full parsers were proven to be useful in such applications as machine translation (see, e.g., (Quirk et al., 2005; Huang et al., 2006; Galley et al., 2006; DeNeefe et al., 2007)), information extraction (Katz & Lin, 2003; Jijkoun & de Rijken, 2004; Cui et al., 2005) and sentence compression (Knight & Marcu, 2000; Riezler et al., 2003; Turner & Charniak, 2005; McDonald, 2006b).

In this introduction we will discuss the motivation for the thesis: the development of non-linear models and approaches to improve their prediction accuracy.

This thesis, as well as most state-of-the-art approaches to natural language parsing, considers data-driven approach - statistical natural language parsing. We start by defining the statistical natural language parsing problem setting. Given a treebank, i.e. a set of sentences manually annotated with their syntactic structures, a mapping function ψ from the space of sentences X to the space of structures Y should be learned. Then this function ψ can be used to predict structures for new sentences. The accuracy of the mapping is evaluated on a testing set using a loss function which penalizes differences between the trees produced by the mapping function and the trees provided by human annotators (*gold standard trees*).

A mapping from the space X to the output space Y is normally defined by assigning scores to each possible sentence-tree pair and predicting the structure with the maximal score:

$$\psi(x) = \arg\max_{y \in Y} f(x, y), \tag{1.1}$$

where f is a scoring function, belonging to a (usually parameterized) class of functions F. Thus, we can distinguish three components in the statistical parsing methods:

- a *class of scoring functions* F;
- a *learning algorithm* which given a training set selects a function $f \in F$ by optimizing certain optimization criterion;
- a *decoding algorithm* which performs parsing, i.e. given a function f and a sentence x outputs the syntactic tree y with the highest score, or often only approximation to this optimal tree.

Though here we define the class of scoring functions, the learning and inference algorithms as three distinct components of the statistical parsing method, they are coupled, as the choice of the function class restricts the choice of learning and inference algorithms. For example consider a class of Probabilistic Context Free Grammars (PCFGs) (Booth, 1969). For learning PCFG models we can use either the maximum likelihood estimation of the parameters or, e.g., one of maximum margin training methods (Tsochantaridis et al., 2004) depending on the choice of the optimization criterion. For decoding with PCFGs usually a variant of the Cocke-Kasami-Younger (CKY) algorithm is used (Cocke & Schwartz, 1970). In this thesis we will primarily focus on the first component: defining an appropriate probabilistic model, its approximations and feature representations, though in some parts of the thesis we will also discuss decoding algorithms, decoding criteria and learning methods.

Research in parsing, as well as in structured prediction in general, has been focused on linear models. By linear models, in this thesis we mean models with the linear scoring function $f(x, y)$:

$$f(x, y) = w^T \varphi(x, y), \tag{1.2}$$

$\varphi(x, y)$ is some feature representation of the sentence-tree pair. The simplest example of such a model for phrase-structure parsing is PCFGs, where the log-probability of a parse tree equals the sum of the log-probabilities of all the productions used to generate the tree. Therefore, $\varphi(x, y)$ for PCFGs is a vector, where each component contains the number of times the corresponding production rule appears in the tree, and w is the vector of the log-probabilities of these production rules.

Even though the parameter vector w is determined by learning algorithms, in most of the approaches model designers have to perform extensive manual selection of predictive structural features before applying the learning algorithms, in other words, they need to specify the feature representation φ. For example consider different methods to enrich treebank grammars with non-local features to produce more expressive PCFG-like models (Collins, 1999; Charniak, 2000; Klein & Manning, 2003). These models are expensive to create and difficult to port to new languages, grammar formalisms or domains. If the model designer omits an important predictive feature, the resulting model will not be able to discover the corresponding statistical dependency and the model accuracy will be worse. It is especially difficult to build these models for languages where the model designer does not have sufficient linguistic knowledge. All these problems motivate the need for methods which automatically induce predictive features from a set of elementary features. A standard approach to model the interaction between elementary features, i.e. extraction of complex predictive features, is the use of latent variables.

It is important to point out, that any meaningful latent variable model is non-linear, otherwise the function class it defines is contained in the class of linear models, and the use of such a latent variable models does not have any advantage over the use of linear models.

Until recently, latent variable models for parsing were not particularly successful, demonstrating results significantly below the state-of-the-art models (Kurihara & Sato, 2004; Matsuzaki et al., 2005; Peshkin & Savova, 2005; Riezler et al., 2002) or they were used in combination with already state-of-the-art models (Koo & Collins, 2005) and showed only a moderate improvement. Recently several methods (Petrov et al., 2006; Petrov & Klein, 2007; Liang et al., 2007), framed as grammar refinement techniques, achieved results similar to the best results shown by any generative method. All these previous approaches considered extensions of a classic PCFG model with latent features.

The use of hidden layers of a neural network to encode complex dependencies between decisions was considered in (Henderson, 2003). This was achieved by using complex patterns of interconnection between hidden layers of the neural network. In this thesis we focus on a recently proposed class of graphical models, Incremental Sigmoid Belief Networks (ISBNs) (Titov & Henderson, 2007c), which are inspired by the neural network model of (Henderson, 2003). ISBNs allow us to move beyond PCFG-like models to models which do not impose strict constraints on the structure of statistical dependencies between latent variables. Exact inference in ISBNs, densely interconnected and large graphical models, is not tractable. We formally show that the computation of the neural network model is equivalent to the variational approximation to ISBNs under certain constraints. Relaxing these constraints we build a more accurate but still tractable variational approximation to this model. Importantly, the introduced variational approximation is shown to significantly outperform the neural network model on the standard phrase-structure parsing benchmark - parsing of the Penn Treebank WSJ corpus (Marcus et al., 1993). The fact that a more accurate approximation leads to a more accurate parser suggests that ISBNs are an appropriate abstract model for natural language parsing. It also motivates further research into more accurate approximations of the ISBNs which should be able to advance the state of the art in natural language parsing.

In order to demonstrate that ISBNs are applicable to a wider variety of problems, we also construct an ISBN model for multilingual dependency parsing. It achieves state-of-the-art performance on a large number of different languages with only minimal feature engineering. These results suggest that ISBNs, and methods developed in other parts of the thesis, have potential applicability to problems outside of natural language processing.

All the latent variable models for parsing we have considered so far are generative: they model the joint probability $p(x, y)$ of sentence x and syntactic structure y. Discriminative models instead are trained to predict structures y for sentences x without making any assumption about the nature of the input distribution $p(x)$. This can be achieved either by modeling conditional probabilities $p(y|x)$ directly (Ratnaparkhi, 1999; Henderson, 2004) or using some alternative discriminative criterion (e.g., margin (Vapnik, 1995) in (Taskar et al., 2004)). In general, it is believed that the asymptotic error of discriminative classifiers is lower than the asymptotic error of generative classifiers though generative models achieve their asymptotic error with smaller training set sizes (Ng & Jordan, 2002). In standard classification problems it often results in improved accuracy of discriminative models for all but very small datasets. In constituent parsing of natural language even the best fully discriminative approaches (Turian & Melamed, 2006; Taskar et al., 2004) struggle to achieve state-of-the-art accuracy and are generally inferior to their generative counterparts. This may be explained by one of the following observations. First, globally trained conditional models require an expensive decoding stage during training, which is usually implemented by using aggressive approximation techniques (Taskar et al., 2004) and locally trained models (Ratnaparkhi, 1999) either suffer from the label-bias problem (Bottou, 1991) or do not have clear motivation for combining scores of entire trees from local scores of decisions (Turian et al., 2006). Secondly, it may also be hypothesized that the accuracy of phrase-structure parsers is still far from their asymptotic accuracy for the considered dataset sizes, and in the region where generative models outperform discriminative ones. All the state-of-the art models in constituent parsing exploit both generative and discriminative models (Charniak & Johnson, 2005; Henderson, 2004; Collins, 2000). It has been done by either using the scores computing by a generative model (Taskar et al., 2004; Collins & Roark, 2004) as an additional feature or reranking candidates provided by a generative model (Henderson, 2004), or both (Charniak & Johnson, 2005; Collins, 2000). Unfortunately, the construction of accurate fully discriminative latent variables problems for parsing is still an open problem. In this thesis we instead propose to exploit a latent space induced by a generative latent variable model in discriminative training. A discriminative model, linear in this latent space, is trained to correct the score predicted by a generative latent variable model. We propose two different approaches for induction of this feature space, both of these approaches can be regarded as generalizations of Fisher kernels (Jaakkola & Haussler, 1998) and TOP kernels (Tsuda et al., 2002a) to parse reranking. We call both these kernels *data-defined* because mapping to their feature space is defined by the training data, as the mapping function to the feature space is dependent on latent variable model parameters estimated from the training data. We demonstrate that the proposed method achieves significant improvement in parsing accuracy. Data-defined kernels construction is a general method and it can be applied to virtually any non-linear generative probabilistic model, such as the proposed latent variable model.

Despite significant progress made in recent years in the area of natural language parsing, until very recently most of research has been focused on the development of statistical parsers trained on large annotated corpora for a particular domain. The best statistical parsers have shown good results on this benchmark, but these statistical parsers demonstrate far worse results when they are applied to data from a different domain (Roark & Bacchiani, 2003; Gildea, 2001; Ratnaparkhi, 1999). This is an important problem because we cannot expect to have large annotated corpora available for most domains. In this thesis, we propose to use data-defined kernels and discriminative methods to address this problem by adapting a statistical parser to a new domain. Data-defined kernels are used to construct a new model which exploits information from a parser trained on a large out-of-domain corpus. Large margin methods are used to train this parser to optimize performance on a small corpus for the target domain.

Another advantage of probabilistic models compared to arbitrary discriminative classifiers is that they are directly trained to approximate probabilities. Not only it can be used in language processing pipelines, but also allows us to modify the decoding criterion. Instead of attempting to select the most probable structure, the model can predict a structure which minimizes expected loss. Loss minimization or minimum Bayes risk (MBR) decoding, has previously been studied in the context of Probabilistic Context Free Grammars (PCFG)(Goodman, 1996) with a linear loss function (labeled constituent recall measure), but there has been no attempt to apply it to more complex non-linear models and different loss function. In this thesis we explain how MBR decoding can be implemented with arbitrary statistical models and different loss functions for the two most popular parsing problems - constituent and dependency parsing. The obtained results suggest that even when the loss function used in MBR decoding is only weakly related to the loss function used in evaluation, results are still significantly improved with respect to standard Maximum A-posteriori Probability (MAP) decoding. This conclusion makes the use of MBR decoding attractive for the natural language parsing tasks, where the true loss function is not usually known or cannot be automatically computed.

The main general goal of this thesis is to demonstrate that latent variable models which do not impose strict constraints on the structure of statistical dependencies are appropriate models for natural language parsing. All the subproblems studied in this thesis are directly related to proving this statement. We show that there exist effective and tractable approximations for this type of models. We also show flexibility and wider applicability of this class of models by constructing a dependency model which achieves state-of-the-art results on a large set of languages. These models provide significant advantages over standard 'linear' models for parsing. First, the powerful feature induction ability simplifies construction of parsers for new problems and domains. Secondly, the latent space induced by the model can be exploited in discriminative rerankers, and also can be used to adapt the parser to new domains. We also show that the non-factorizability of the model does not prevent us from using different decoding criteria by introducing new methods for Bayes risk minimization applicable to arbitrary models of natural language parsing.

The thesis is organized as follows:

- Chapter 2 reviews background on statistical natural language parsing, including state-of-the-art models for constituent and dependency parsing. It also focuses on previous work on latent variable models of parsing.

- Chapter 3 describes the recently proposed latent variable model for natural language parsing - Incremental Sigmoid Belief Networks (ISBNs)(Titov & Henderson, 2007c). The chapter explores its connection with the previous neural network model, Simple Synchrony Networks (SSNs) (Henderson, 2003) showing that the computation of this neural network model is equivalent to the variational approximation for ISBNs under certain constraints, and a more accurate variational approximation is then proposed. This chapter also presents experiments with artificial data, demonstrating that the proposed approximation is indeed more accurate.

- Chapter 4 considers constituent parsing with ISBNs. It is demonstrated that the constructed variational approximation achieves state-of-the-art performance on the constituent parsing problem and outperforms the SSN model which uses the same linguistic features and model structure.

- Chapter 5 proposes a new dependency parsing model based on ISBNs. It presents multilingual parsing experiments and analysis of the model performance.

- Chapter 6 introduces data-defined kernels for parsing and presents their evaluation on a constituent parsing task with ISBNs.

- Chapter 7 considers domain adaptation and proposes a novel approach for domain adaptation with data-defined kernels. The domain adaptation method is evaluated on adaptation of the ISBN constituent parser to various domains.

- Chapter 8 describes algorithms for Bayes risk minimization in natural language parsing with different loss functions both for constituent and dependency parsing. The proposed method is evaluated with ISBNs and several other parsing models.

Some of the material presented in this thesis was published in conference proceedings. Parts of chapters 3 and 4 are drawn from material in (Titov & Henderson, 2007c; Titov & Henderson, 2007a). Parts of chapter 5 were published in (Titov & Henderson, 2007d; Titov & Henderson, 2007b). Chapter 6 includes material from (Titov & Henderson, 2005; Henderson & Titov, 2005). Parts of chapter 7 are taken from (Titov & Henderson, 2006c). Material presented in (Titov & Henderson, 2006b) was used in chapter 8.

Chapter 2

Statistical Natural Language Parsing

In this chapter we will define the problems considered in the thesis and present a review of relevant previous work in statistical methods for natural language parsing, with emphasis on non-linear methods: latent variable models and other feature induction techniques.

We will mostly focus on supervised statistical parsing which we defined in chapter 1. There had not been much work on supervised statistical natural language parsing until the 90s when the first treebanks started to appear. First model estimation experiments considered such treebanks as Lancaster corpus of computer manuals (Black et al., 1992) or Air Travel Information System (ATIS) corpus (Dowding et al., 1993). However, these treebanks were small, represented only very restricted language and were clearly not appropriate for evaluation and learning of broad-coverage parsers. Virtually all the statistical parser development efforts starting from mid-90s were evaluated and trained on a much more complex treebank - the Penn Wall Street Journal (WSJ) Treebank (Marcus et al., 1993). Soon treebanks for different languages started to emerge (Hajic, 1998; Chen et al., 1999), but unfortunately annotation schemes and grammar formalisms for these treebanks are generally incompatible, which made evaluation and construction of parsing models for multiple languages virtually impossible. An important step forward was made in 2006, when in CoNLL-2006 shared task (Buchholz & Marsi, 2006) dependency treebanks for 13 different languages were standardized, and this effort was continued during CoNLL-2007 shared task (Nivre et al., 2007a) bringing another 6 treebanks into the standardized form.

We will start this chapter by introducing the two most frequently considered types of natural language parsing, constituent parsing and dependency parsing, and discussing the corresponding parsing methods.

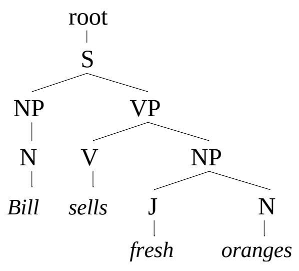


Figure 2.1: An example constituent tree.

2.1 Parsing Problems

2.1.1 Phrase-Structure Parsing Problem

Constituent parsing, or phrase-structure parsing, is a syntactic parsing task, where syntactic structure of a sentence is represented as a tree, which defines a hierarchical decomposition of the sentence into phrases. An example of a constituent tree is presented in figure 2.1. For example, this tree declares that the adjective (J) *fresh* and the noun (N) *oranges* form a noun phrase (NP) *"fresh oranges"*, which, when combined with the verb (V) *sells*, forms the verb phrase (VP) *"sells fresh oranges"*. We will further refer to the labels associated with each type of a phrase as constituent labels (e.g., VP, NP, S) and to the part of speech type labels (N, V, J in the example) - as PoS tags. We will also refer to leaves as terminals and to internal nodes of the tree as non-terminals, as they correspond to terminals and non-terminals of the underlying grammar.

We can think of a constituent tree as a set of non-crossing labeled brackets, i.e. triples (Z, s, e), where Z is a label of an internal node, s and e are indexes of the first and the last words spanned by the node. Accuracy of the model is evaluated by comparison of trees $\{y_i\}_{i=1}^n$ predicted by the model for sentences $\{x_i\}_{i=1}^n$ with a gold standard trees $\{y_i^\star\}_{i=1}^n$ provided by a human annotator. The most standard measures of accuracy (PARSEVAL measures (Black et al., 1991)) for constituent parsing involve the number of equally labeled brackets $\sum_i |y_i \bigcap y_i^\star|$, where for bracket equality both the spans and the labels should be the same. The Labeled Recall (LR) measure is defined as $\frac{\sum_i |y_i \bigcap y_i^\star|}{\sum_i |y_i^\star|}$, Labeled Precision (LP) as $\frac{\sum_i |y_i \bigcap y_i^\star|}{\sum_i |y_i|}$ and their harmonic mean $F_1 = \frac{2\sum_i |y_i \bigcap y^\star|}{\sum_i |y_i^\star| + \sum |y_i|}$.[1]

2.1.2 Dependency Parsing Problem

Another way to define a syntactic structure of a sentence is to use dependency trees (Tesniére, 1959; Hudson, 1984). There has been an increasing interest

[1]Parents of tree leaves, part of speech (PoS) tags, are excluded from parse tree evaluation and their accuracy is evaluated separately. Assignment of PoS tags, PoS-tagging, often precedes the parsing stage and PoS tags are provided to the parser along with words. A *root* node presents in all the trees, but it is also excluded from accuracy evaluation.

in dependency parsing recently both because learning to predict dependency structures is arguably a simpler task than learning to predict constituent trees, and also because for a number of applications dependency representation is more appropriate than hierarchical phrase structure. For example, it is easier to infer a predicate-argument structure from dependency trees.

Dependency trees represent dependencies of words in sentences as a directed acyclic graphs, often with labeled edges where each label represents a type of the relation. An example of a labeled dependency tree is presented in figure 2.2. As it is often done, we augment the dependency trees with a *root* node which serves as a root for the tree. For each edge a parent node is called a head and a child node is called a dependent. This tree, for example, indicates that the word *Bill* is a subject of the verb *called* and the word *oranges* is an object.

It is usual to enforce *projectivity* constraint on the grammar (Lecerf, 1960). A tree is projective in respect to word order iff for any edge $v \rightarrow u$ and any word w between u and v, w has v as its ancestor. The tree in figure 2.2 is an example of a projective structure. Even though the projectivity requirement is too strict for many languages, especially for languages with free or flexible order, it is widely used in practice and often viewed as a reasonable assumption.

Standard measures of accuracy for dependency parsing are labeled and unlabeled attachment scores. The labeled attachment score is the fraction of relations where both edges and labels match, and the unlabeled accuracy is defined similarly but without requiring labels to match (or when there are no labels available).

Also we should note that in this thesis we consider only genuine dependency parsing approaches. We do not consider statistical models for constituent parsing trained on constituent treebanks, the output of which is then post-processed to produce dependency structures (see, e.g., section 7.7.1 of (Collins, 1999)). There is no methodologically appropriate way to compare these models with genuine dependency parsers trained on the post-processed gold standard trees. First, the constituent parser is given more information during training than the dependency parser. Secondly, and probably more importantly, when doing this comparison the constituent parser is trained on a “natural” task, where as a dependency parser is trained on data produced by likely imperfect conversion. Therefore, it is not entirely surprising that the constituent parser outperforms pure dependency parsers in this set-up. Also we should not that this difference is not large, for example, the difference between the state-of-the-art constituent parser (Charniak, 2000) and the state-of-the-art dependency parser (McDonald et al., 2006) trained on dependency relations extracted from the WSJ Penn Treebank does not exceed 0.7% in the unlabeled attachment score (McDonald, 2006a).

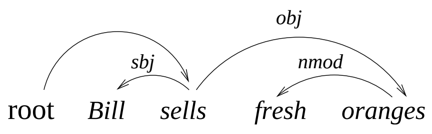


Figure 2.2: An example dependency tree.

2.2 Decomposition of Structures and Linear Models

As described in chapter 1, in order to define a mapping from sequences of words to syntactic trees we need to define a class of scoring functions F, where each function f from this class assigns a score to a possible tree y reflecting how plausible this tree is. We consider that the class of functions is parameterized by a vector w, i.e. the scoring function has a form $f(x, y|w)$, where x is the input sentence and y is the considered syntactic tree.

The space of possible syntactic trees Y is unbounded and, thus, parsing cannot be treated as simply an instance of standard multiclass classification problems. In order to build the parsing model one should explore correlations between components of syntactic trees.

The usual way to exploit correlations in the output structures is to define decomposition of sentence-tree pairs into set of *properties* of sentence-tree pairs (also called *events* in (Collins, 1999)), and model a score $f(x, y|w)$ as a superposition of scores of these properties c:

$$f(x, y|w) = \sum_c f'(c|w).$$

Usually there is a distinct parameter associated with each property c and this decomposition can be viewed as a linear model defined in equation (1.2).

As an example consider PCFGs (Booth, 1969). PCFGs are a stochastic modification of context free grammars, where there is a probability associated with each production rule in the grammar. Words are terminal symbols in this grammar. Therefore, the log-probability of a sentence-tree pair (x, y) $f(x, y|w)$ decomposes into the sum of the log-probabilities of each production rule appearing in the tree. In other words, each property in this model is presence of a particular production rule in the parse tree. In dependency parsing the MST model (McDonald et al., 2005) defines decomposition of a dependency structure into edges and each edge is scored independently. In general, it is not required that these properties correspond to non-overlapping components of a tree, consider the example of convolution tree kernels for constituent parsing (Collins & Duffy, 2002) or any other DOP (Data-Oriented Parsing) parsing model (Bod, 2003). In DOP models each property of the tree is presence of a particular subtree in the tree and, thus, the score of the tree is the sum over scores corresponding to all its overlapping subtrees.

The decomposition should be performed so as to maximally explore the correlations in the space of parse trees. As discussed in (Collins, 1999), when defining the decomposition two main requirements should be respected: expressiveness (i.e. sufficient discriminative power of the function family) and compactness. Expressiveness requires that there exist an assignments for the parameter w so that for most sentences the wrong parse trees will have lower scores than the correct parse trees. Compactness roughly requires that the model have the minimal number of parameters, in other words, low dimensionality of the parameter vector w. The compactness requirement ensures that the mapping is learnable from a limited amount of training data. This requirement is important, even though good generalization properties can be achieved even with a very large parameter space by using appropriate regularization (Vapnik, 1979; Vapnik, 1995; Bartlett, 1997; Evgeniou et al., 2000; Ng, 2004). Also, the use of a large number of parameters, in general, will slow down parsing and learning.

The standard PCFG decomposition fails on expressiveness criteria, and even given an optimal set of weight they will not be able to differentiate between correct and incorrect parses for many sentences and, thus, their asymptotic performance is significantly below the performance of state-of-the-art methods learned on treebanks of limited size. Conversely, a model which has a distinct parameter for each possible tree, effectively memorizing the training set, will fail on the compactness criteria.

It is clear that defining such decomposition is a challenging problem and generally requires sufficient linguistic knowledge and will be language dependent. In the following sections we will discuss how latent variables models and other feature induction methods can be used to avoid the need to explicitly define this decomposition. But we first start with the introduction of arguably the most standard class of statistical parsing models: history-based probabilistic models.

2.3 History-based Models

Many models in natural language parsing, including the most standard PCFG models, can be regarded as *history-based probability models* (Brown et al., 1992). History-based probability models use the chain rule to factorize the probability of a parse tree according to the parser decisions. They can be used to model conditional probability $P(Y|X)$ (e.g., (Ratnaparkhi, 1999)):

$$P(Y|X) = P(D^1, ..., D^m|X) = \prod_t P(D^t|D^1, \ldots, D^{t-1}, X), \tag{2.1}$$

where $D^1, \ldots, D^m$ is a sequence of decisions used to create the tree Y, or joint (or generative) probability $P(X, Y)$:

$$P(X, Y) = P(D^1, ..., D^m) = \prod_t P(D^t|D^1, \ldots, D^{t-1}), \tag{2.2}$$

where $D^1, \ldots, D^m$ is a sequence of decisions used to generate both the tree Y and the sentence X. In both cases we assumed that there is a one-to-one correspondence between sequences of decisions and trees.[2] Deterministic statistical models for dependency parsing (e.g., (Nivre et al., 2004)) can also be viewed as conditional history-based models. Even though they do not have direct probabilistic interpretation, they predict the most likely next decision on the basis of the sequence of previous decisions. For now we will focus on generative probability models as they are still more standard in parsing and do not suffer from the label-bias problem (Bottou, 1991), though most of the discussion can equally be applied to conditional models.

When modeling the sequence of decisions, each decision probability $P(D^t|D^1, \ldots, D^{t-1})$ is conditioned on the unbounded sequence of all the previous decisions $D^1, \ldots, D^{t-1}$. Straightforward modeling of these probabilities would require an unbounded number of model parameters, which would compromise the requirement for model compactness. The standard approach is to define independence assumptions, i.e., to assume that the distribution of each decision D^t is dependent only on a subset of relevant previous decisions $R(D^1, \ldots, D^{t-1}) = (R_1, \ldots, R_n)$:

$$P(D^t|D^1, \ldots, D^{t-1}) = P(D^t|R(D^1, \ldots, D^{t-1})),$$

thus, only distribution $P(D^t|R(D^1, \ldots, D^{t-1}))$ should be estimated from the training set. Independence assumption effectively defines decomposition of the tree because the log-probability assigned to the tree by a model of this form can be represented as the sum of the scores assigned to subsequences of decisions $(D^t, R(D^1, \ldots, D^{t-1}))_{t=1}^m$.

At the same time, the use of history-based models does not automatically imply that the score of the model decomposes over the structure (i.e. it can be represented as a linear combination of features). In (Henderson, 2003; Henderson, 2004) a neural network is used to estimate these probabilities, in this computation compressed representation of the entire parsing history in the form of hidden layer activations is used in the computation of the decision probability. The neural network compresses the unbounded parsing history in a vector of real numbers without making any explicit independence assumptions.[3] In a similar way Incremental Sigmoid Belief Networks (Titov & Henderson, 2007c), latent variable models considered in details in chapter 3, represent history as a distribution of latent variable vectors. Both of these models, as well as other models discussed in section 2.5, do not decompose over sub-structures and are non-linear.

[2]Though it is not always true in practice (Collins, 1999).

[3]Similarly a language model of (Xu et al., 2003) uses a neural network to compress the set of structured and lexical features of the Structured Language Model (Chelba & Jelinek, 2000) in a hidden layer. The structured set of features in the Structured Language Model is induced from a partial syntactic structure constructed by an incremental parser. The crucial difference between the approaches of (Xu et al., 2003) and (Henderson, 2003) is that model of (Xu et al., 2003) does not have interconnection between layers corresponding to different steps. Therefore (Xu et al., 2003) make independence assumptions, even though their model can handle large and sparse (but finite) sets of features.

2.4 Discriminative Models vs. Generative Models

In this chapter we provide a short comparison of discriminative and generative approaches to constituent and dependency parsing. We will also discuss such techniques as candidate reranking (Ratnaparkhi, 1999; Collins & Koo, 2005). As we discussed in the introduction to the thesis, generative probability models estimate the joint probability $p(x, y)$ of sentence x and syntactic structure y, whereas discriminative models are instead trained to predict structures y for sentences x.

2.4.1 Constituent Parsing

Though it is generally believed that performing conditional modeling, which is directly related to the considered parsing task, is preferable to modeling joint probability (Vapnik, 1995), all the state-of-the-art models for constituent parsing rely in some way on a generative component.

The most accurate generative models (Charniak, 2000; Henderson, 2003; Collins, 1999) achieve F_1 score on the standard WSJ Penn Treebank benchmark in between 88.0% and 89.5%.[4] The history-based conditional log-linear model (Ratnaparkhi, 1999) achieved lower F_1 of around 86.9.[5] The main drawback of conditional history-based models is that they suffer from the *label-bias problem* (Bottou, 1991; Lafferty et al., 2001). The label-bias problem can be observed if we compare how the probability mass is distributed in conditional and generative probability models. In generative history-based models, at each step local decisions compete against each other and not against all other possible derivation sequences for the considered sentence. But unlike generative models, which can downgrade unlikely derivation sequences when estimating probabilities of the following words, the total probability mass in conditional models is distributed among the possible continuations of the derivation. This means that conditional history-based models are not able to accurately model parsing tasks where dependencies between decisions and words are not local.

This problem is solved by using global discriminative training, e.g. in max-margin parsing (Taskar et al., 2004; Tsochantaridis et al., 2004) a model is trained to assign a larger score (larger by a margin) to the gold standard tree than to any other possible tree for the sentence. Unfortunately, this approach requires performing decoding (i.e. parsing) on each training step, which leads to excessively long training times.[6] Also, the use of dynamic programming for

[4]The latent PCFG models (Petrov et al., 2006; Petrov & Klein, 2007) achieve slightly better results, but the presented numbers are for Bayes risk decoding instead of MAP decoding used in other parsers. This makes their results not directly comparable. See chapter 8 for details.

[5]This value is computed from labeled recall of 86.3% and precision of 87.5% reported in (Ratnaparkhi, 1999) and might have a rounding error.

[6]According to (Turian & Melamed, 2006) it took several months to train the max-margin model described in (Taskar et al., 2004) on the WSJ treebank (Marcus et al., 1993) reduced to a set of sentences of length under 15 words in length.

the inference permits only specific types of features (or kernels) decomposable over substructures; other features or kernels are not allowed in this framework. The discriminative model of (Taskar et al., 2004) managed to beat the Collins' generative parser (Collins, 1999) only when it used probability estimates from the Collins' model as one of its features. Global discriminative training was also attempted in (Turian & Melamed, 2006). They achieved statistically significant improvement over their baseline, Collins' parser (Collins, 1999), without the use of probability estimates from a generative model. They used a nonlinear model, ensembles of decision trees, to perform induction of complex features from elementary ones. We will discuss their approach in more detail in the following section 2.5.

However, a far more typical approach for using discriminative models in constituent parsing is reranking. In reranking an initial model produces a list of candidates and the second model, a reranker, select a candidate from this list. Reranking was originally proposed in (Ratnaparkhi, 1999), where the author observed that the perfect reranker (oracle) can achieve 50% error reduction in F_1 measure over the considered parser if it selects the best tree from only 20 top parses returned by the parser. Normally a generative model is used as the initial parser and a discriminative model as the reranker. To facilitate this setup, a number of efficient dynamic programming algorithms were proposed to efficiently obtain large candidates list from PCFG-like models (Huang & Chiang, 2005; Jimenez & Marzal, 2000). Aside from different learning criterion, the major advantage of the reranking approach is that arbitrary features of parse trees can be easily incorporated in a reranking models resulting in richer models, whereas integration of arbitrary features in any full parser would normally result in non-tractable decoding. It is especially difficult to integrate interdependent features directly in generative models, as a straightforward approach would not result in a proper probability model. All these observations explain why discriminative reranking is so popular. The main drawback of the reranking approach is that the models are biased towards prediction of an initial model - they are limited to prediction of a candidate from the provided list of candidates.[7] There has been considerable research on different algorithms and feature representation for discriminative reranking for parsing (Collins & Koo, 2005; Collins, 2000; Collins & Duffy, 2002; Henderson, 2004; Charniak & Johnson, 2005) and the best of these models achieved more than 10% error reduction over the corresponding generative baselines. However, the majority of these methods not only use a candidate list provided by a generative model but also probability estimates from initial generative models (Collins & Koo, 2005; Collins, 2000; Collins & Duffy, 2002; Charniak & Johnson, 2005).

On the basis of this analysis, we can see that though joining discriminative and generative models for constituent parsing is beneficial, there is no direct evidence that discriminative models are preferable to their generative counterparts. This motivates the types of models considered in this thesis: accurate genera-

[7]An approach where reranking was used only during training but full parsing during actual use of the model was attempted in (Henderson, 2004), and the resulting model did not achieve any improvement over the generative baseline.

tive latent variable models and a reranker which uses the space induced by the latent variable model to improve the accuracy by optimizing a discriminative criterion. Though, as we discussed above, it is generally difficult to integrate different inter-dependent features into generative models, latent variable models leverage this limitation by automatically inducing latent annotation and not requiring complex interdependent features.

2.4.2 Dependency Parsing

Discriminative methods have been much more successful in dependency parsing than in constituent parsing. Even though the first statistical methods for dependency parsing were generative (Eisner, 1996), all the state-of-the-art approaches for dependency parsing are discriminative (e.g. see (Yamada & Matsumoto, 2003; Nivre et al., 2006; McDonald et al., 2005; McDonald et al., 2006)).

Two types of discriminative approaches are common in dependency parsing: transition-based methods and global models.

Transition-based models are similar to history-based models in that they also model parse trees as sequences of decisions, the main difference being that they do not necessarily use a probabilistic model. Different parsing strategies have been considered for transition-based parsing, including LR parsing, as in (Sagae & Tsujii, 2007; Watson & Briscoe, 2007), and modifications of shift-reduce, as has been done in multi-pass parsing strategies (Yamada & Matsumoto, 2003) and in an arc-eager single-pass parser (Nivre et al., 2004). All these approaches handle only projective parsing, but in (Attardi, 2006) the set of actions was extended to handle non-projective dependencies. Most of transition-based methods use greedy strategies: at each step a single analysis is preserved. Different machine-learning methods are used in different models to predict a correct decision, e.g., support vector machines (Vapnik, 1998) in (Yamada & Matsumoto, 2003; Nivre et al., 2006), memory-based learning (Daelemans & van den Bosch, 2005) in (Nivre et al., 2004) or maximum entropy classifiers (Berger et al., 1996) in (Attardi, 2006). Recently, instead of using deterministic approaches a beam search in the induced conditional history-based model was considered (Duan et al., 2007). As discussed in section 2.4.1, it is expected that such models may be affected by the label-bias problem.

Global models are instead trained to score complete dependency parse trees. The first discriminative global models were proposed in (McDonald et al., 2005), where it was shown that searching for an optimal dependency tree under a statistical model which factorizes over arcs in the dependency tree is equivalent to searching for a minimum spanning tree (MST) in the directed graph (Tarjan, 1977). Later algorithms were proposed where this factorizability assumption was relaxed (McDonald et al., 2006; Carreras, 2007). These methods do not explicitly define probability model, but very recently log-linear models were proposed to define a conditional distribution over dependency trees (McDonald & Satta, 2007; Koo et al., 2007; Smith & Smith, 2007), all of which use the Kirchhoff's Matrix-Tree theorem (Williams, 1996) to efficiently compute the partition function.

The fact that conditional models can be directly applied to the dependency parsing problem lessens the motivation for doing discriminative reranking. However, reranking allows for the introduction of features in the model which would make decoding NP-hard (McDonald et al., 2006) and difficult to approximate. Discriminative reranking for dependency parsing was explored in (Hall et al., 2007b), where the reranker achieved average error reduction of the labeled attachment score of around 7% over edge-factored MST model (McDonald et al., 2005) used as the source of candidate parses.

For dependency parsing, the previous work seems to suggest that discriminative approaches are preferable and help to achieve good results, but, however, it is worth noting that there has not been much research on generative models for dependency parsing. Lack of strong requirements on model structure in ISBNs allows us to easily apply them to dependency parsing and test whether generative models for dependency parsing can also achieve state-of-the-art accuracy. These experiments are described in chapter 5.

2.5 Non-linear Models for Parsing and Feature Induction

As discussed in the introduction there has not been much previous work on non-linear models for parsing and feature induction. In this section we will give a brief review of the previous non-linear models both for constituent and dependency parsing.

Standard PCFGs defined on the basis of original treebank annotation, as discussed in section 2.2, make excessively strong independence assumptions and consequently do not have sufficient discriminative power. They achieve accuracy of around 73% on the standard Penn Treebank WSJ (Klein & Manning, 2003). To address this problem and improve the discriminative power of standard PCFGs, different approaches to extend original treebank annotation has been proposed. Some of these approaches were focused on lexicalization, i.e. extension of constituent labels by annotating them with lexical information, usually with the lexical head of the phrase spanned by the corresponding constituent (Collins, 1997; Charniak, 1997; Collins, 1999). However modeling dependencies between all the lexically annotated constituents would lead to exceedingly sparse feature space and these models perform smoothing over these sparse statistics using directly only bilexical dependencies. The most accurate Collins' model 2 (Collins, 1999) achieved accuracy which is still close to the state-of-the-art results - labeled F_1 measure of 88.2% in the standard settings. However, lexicalization leads to increased asymptotic complexity of parsing: $O(n^5)$ when using straightforward parsing algorithms or $O(n^4)$ with more efficient algorithms for bilexical PCFGs (Eisner & Satta, 1999) comparing to $O(n^3)$ asymptotic complexity with unlexicalized PCFGs. In (Klein & Manning, 2003) instead of lexicalization it was proposed to annotate constituent labels with a set of carefully chosen labels motivated by linguistic considerations and then to use a PCFG model in the extended grammar. This model, consequently, was much more efficient, allowing for $O(n^3)$ asymptotic parsing time and, importantly, achieved the competitive labeled F_1 measure of 85.7%. This work inspired latent variable models which augment non-terminal symbols of Probabilistic Context Free Grammars (PCFGs) with latent variables (Kurihara & Sato, 2004; Matsuzaki et al., 2005; Prescher, 2005; Petrov et al., 2006; Liang et al., 2007; Finkel et al., 2007).

Unlike one-to-one mapping defined in (Klein & Manning, 2003), in these latent variable models there exist many trees extended with latent annotations corresponding to a single tree in the original treebank grammar.[8] Consequently, the probability of a tree in the original grammar can be obtained by summing over all the corresponding extended trees, i.e. *marginalizing* out all the latent variables. This marginalization over latent variables makes the model non-linear. Even though marginalization can be performed efficiently by using dynamic programming, parsing under this model is NP-hard (Matsuzaki et al., 2005; Sima'an, 1992). Instead approximate parsing algorithms are considered. The family of parsing algorithms described in (Petrov & Klein, 2007) is related to Bayes risk minimization methods proposed in chapter 8 of this thesis. Another problem of this approach is that it is difficult to discover the appropriate split of non-terminals. Thus, early models which used straight-forward implementations of expectation maximization algorithms (Dempster et al., 1977; Pereira & Schabes, 1992) were not particularly successful (Matsuzaki et al., 2005; Prescher, 2005) against the manually annotated model of (Klein & Manning, 2003). To solve this problem the split-and-merged approach was considered in (Petrov et al., 2006; Petrov & Klein, 2007) and Dirichlet Process priors in (Liang et al., 2007). The model of (Petrov & Klein, 2007) achieved F_1-measure of 90.1% on the WSJ parsing task which is currently the best reported result for a single model parser. Note, though, that this model used a form of Bayes risk minimization in the decoding and, therefore, it is not entirely fair to compare the results to thous of maximum a-posteriori decoding with other parsers.

Though recent research in feature induction methods for parsing is dominated by latent PCFG models, other approaches have also been considered. This includes a neural network model SSN (Henderson, 2003; Henderson, 2004), which uses a history-based model with a neural network hidden layer to represent a parsing history:

$$P(D^t|D^1, \ldots, D^{t-1}) = P(D^t|h(D^1, \ldots, D^{t-1})),$$

where $h(D^1, \ldots, D^{t-1})$ is a vector of hidden unit activations associated with a parsing state at time t, or hidden state. Hidden vectors are interconnected with a dynamic pattern of interconnection, so that a hidden vector for the current state is computed from the hidden vectors for the previous most relevant states and also by using information about previous relevant decisions. It is

[8] If no restriction is placed on the assignment of latent variables and each original constituent label is split into m latent labels, there exist n^m latent trees corresponding to a single observed tree, where n is the number of constituents in the tree.

important to note that this model, like latent PCFG models, does not make any explicit independence assumptions. The neural network model achieves results of F-measure 89.1% on the standard task and can be further improved by discriminative retraining to demonstrate state-of-the-art accuracy of 90.1%, though such an accuracy score is achieved only when reranking candidates from the generative model.

Recently, a latent variable model, Incremental Sigmoid Belief Networks, inspired by this neural network model, was proposed (Titov & Henderson, 2007c). Instead of representing a parsing state as a vector of neural network activations, this model represents a parsing state as the distribution of a vector of binary latent variables. One important difference between ISBNs and latent PCFGs is that ISBNs use a vector representation instead of atomic labels in latent PCFGs; therefore, ISBNs can explore larger and more structured similarity spaces. Another difference is that ISBN models, unlike PCFGs, do not place strong restrictions on the structure of statistical dependencies between latent variables, which makes approximation of this model a much more challenging task. Both the neural network model and the ISBN model will be introduced in greater detail in the following chapters, where we will discuss approximations for ISBNs and their application to constituent and dependency parsing tasks.

Though latent PCFGs, SSNs and ISBNs are arguably the most accurate non-linear probabilistic models, there has also been other attempts to apply latent variable models to parsing. In (Koo & Collins, 2005), an undirected graphical model was used for parse reranking. However, in this case the model will not be able to predict an output structure which does not belong to the list of a few candidates provided by some baseline model, and thus the full potential of latent variable models will not be realized. Another approach would be to consider each decision as a separate inference problem with its own set of latent variables. This was done for dependency parsing but with very limited success (Peshkin & Savova, 2005). Roughly, their model considered the whole sentence at a time, with the latent variable model being used to decide which words correspond to leaves of the dependency tree. The chosen words are then removed from the sentence and a new instance of the model is applied to the reduced sentence.

All the feature induction methods discussed so far have used probabilistic models. Unlike these approaches the fully discriminative constituent parser proposed in (Turian & Melamed, 2006) uses decision trees to build compound features from atomic features during training. It achieves results significantly better than these of Collins' parser on the WSJ dataset reduced to sentences of less than 15 words in length (89.4 % F_1 vs. 88.4% F_1). Importantly, the model does that without relying on any probability estimates from generative models, which was usually the case for discriminative methods in constituent parsing as was discussed in section 2.4.1. Unfortunately, training times for this parser are still large. Learning on this restricted dataset took around 130 CPU-days, though parallelization made such training feasible. This model was trained locally to predict correct derivation decisions at each derivation step in the treebank, i.e. very similarly to discriminative transition-based parsers for dependency parsing (Yamada & Matsumoto, 2003; Nivre et al., 2004; Nivre et al., 2006; Duan et al., 2007). Large training times for their model might seem somewhat surprising considering that there is no need for inference during training, which was the main computational challenge for the discriminatively trained parsers (Taskar et al., 2004). However, one should observe that the number of possible decisions is still large and probably significantly larger than that in dependency parsing, resulting in a training set of 40 million decisions (Turian & Melamed, 2006). The model is not locally normalized which probably makes it less amenable to label-bias problem. This property, however, does not allow it to be treated as a probabilistic model because computation of the partition function is not tractable.

Even though research into non-linear models in natural language parsing is still in its infancy, the state-of-the-art results that have been demonstrated (Henderson, 2003; Henderson, 2004; Petrov et al., 2006; Petrov & Klein, 2007; Turian & Melamed, 2006) suggest that non-linear models represent a very promising direction. Importantly, they significantly minimize the time needed for feature engineering and permit the creation of accurate trainable models for virtually arbitrary natural languages.

Chapter 3

Incremental Sigmoid Belief Networks

In this chapter we describe a class of graphical models appropriate for natural language parsing where the model structure is a function of the output structure and propose inference algorithms for this model. ISBNs were recently introduced in (Titov & Henderson, 2007c) and can be applied to various structured prediction problems, i.e. classification problems with a large (or infinite) structured set of output categories. Natural language parsing is one example of such a problem, but structured prediction problems frequently arise not only in natural language processing but also in biology (e.g. protein structure prediction), chemistry, or image processing.

ISBNs use history-based probability models. As we discussed in section 2.3, the most common approach to handling the unbounded nature of the parse histories is to choose a pre-defined set of features which can be unambiguously derived from the history (e.g. (Charniak, 2000; Collins, 1999; Nivre et al., 2004)). Decision probabilities are then assumed to be independent of all information not represented by this finite set of features. ISBNs instead use a vector of binary latent variables to encode the information about the parser history. This history vector is similar to the hidden state of a Hidden Markov Model. But unlike the graphical model for an HMM, which specifies conditional dependency arcs only between adjacent states in the sequence, the ISBN graphical model can specify conditional dependency arcs between states which are arbitrarily far apart in the parse history. The source state of such an arc is determined by the partial output structure built at the time of the destination state, thereby allowing the conditional dependency arcs to be appropriate for the structural nature of the problem being modeled. This structure sensitivity is possible because ISBNs are a constrained form of the switching model (Murphy, 2002), where each output decision switches the model structure used for the remaining decisions.

ISBNs are closely related to the neural network (SSN) of (Henderson, 2003), but have a clear probabilistic semantics for all their variables. ISBNs are a kind of Sigmoid Belief Network (Neal, 1992), but are dynamic models and have an incrementally specified model structure. Because these models use directional arcs and only allow decisions to switch the future model structure, the portion of the model structure which affects the inference of any given $P(D^t|D^1, \ldots, D^{t-1})$ is always known, thereby avoiding the need to sum over model structures, as discussed in section 3.2. As we will show in this thesis, these properties of ISBNs allow us to have large numbers of latent state variables without making the models impractical to use.

Large numbers of latent variables in heavily interconnected directed models make exact inference intractable. We demonstrate the practical applicability of these models by providing efficient approximations. We consider two forms of approximation for ISBNs, a feed-forward neural network approximation (NN) and a form of mean field approximation (IMF). We formally show that the computation of the SSNs (NN) model is equivalent to the variational approximation to ISBNs under certain constraints. Relaxing these constraints we build a more accurate but still tractable variational approximation to this model, the IMF approximation. Both these approximations give us valid probability models.

In this chapter we consider evaluation on synthetic data and in the following chapters we will evaluate the model on real tasks: constituent and dependency parsing. In the artificial experiment, we trained both of the approximation models on artificial data generated from random ISBNs. The NN model achieves a 60% average relative error reduction over a baseline model and the IMF model achieves a further 27% average relative error reduction over the NN model. These results demonstrate that the distribution of output structures specified by an ISBN can be approximated, that these approximations can be learned from data, and that the IMF approximation is indeed better than the NN approximation.

3.1 Sigmoid Belief Networks

A Sigmoid Belief Network (SBN) (Neal, 1992) is a type of Bayesian Network with binary variables and conditional probability distributions (CPDs) in the form:

$$P(S_i = 1|Par(S_i)) = \sigma(\sum_{S_j \in Par(S_i)} J_{ij} S_j), \tag{3.1}$$

where σ denotes the logistic sigmoid function $\sigma(x) = 1/(1 + e^{-x})$, and J_{ij} is the weight for the arc from variable S_j to variable S_i.[1] SBNs are similar to feed-forward neural networks, but unlike neural networks SBNs have a precise probabilistic semantics of their hidden variables. In ISBNs we consider a generalized version of SBNs where we allow variables with any range of discrete values. The normalized exponential function is used to define the CPDs at these nodes:

[1]For convenience, where possible, we will not explicitly include bias terms in expressions assuming that every latent variable in the model has an auxiliary parent variable set to 1.

$$P(S_i = k|Par(S_i)) = \frac{\exp(\sum_{S_j \in Par(S_i)} W^i_{kj} S_j)}{\sum_{k'} \exp(\sum_{S_j \in Par(S_i)} W^i_{k'j} S_j)}, \tag{3.2}$$

where W^i is the weight matrix for the variable S_i.

Exact inference with all but very small SBNs is not tractable. Initially sampling methods were used (Neal, 1992), but this is also not feasible for large networks, especially for the dynamic models of the type described in section 3.3. Variational methods have also been proposed for approximating SBNs (Saul et al., 1996; Saul & Jordan, 1999). The main idea of variational methods (Jordan et al., 1999) is, roughly, to construct a tractable approximate model with a number of free parameters. The free parameters are set so that the resulting approximate model is as close as possible to the original model for a given inference problem.

The simplest example of a variation method is the mean field method, originally introduced in statistical mechanics and later applied to neural networks in (Hinton et al., 1995). Let us denote the set of visible variables in the model by V and hidden variables by $H = h_1, \ldots, h_l$. The mean field method uses a fully factorized distribution $Q(H|V) = \prod_i Q_i(h_i|V)$ as the approximate model, where each Q_i is the distribution of an individual latent variable. The independence between the variables h_i in this approximate distribution Q does not imply independence of the free parameters which define the Q_i. These parameters are set to minimize the Kullback-Leibler divergence between the approximate distribution $Q(H|V)$ and the true distribution $P(H|V)$ or, equivalently, to maximize:

$$L_V = \sum_H Q(H|V) \ln \frac{P(H,V)}{Q(H|V)}. \tag{3.3}$$

The expression L_V is a lower bound on the log-likelihood $\ln P(V)$. It is used in the mean field theory (Saul & Jordan, 1999) as an approximation of the log-likelihood. However, in our case of dynamic graphical models as explained later, we have to use a different approach which allows us to construct an incremental structured prediction method without needing to introduce the additional parameters proposed in (Saul & Jordan, 1999), as we will discuss in section 3.4.3.

3.2 Exploiting Structural Locality

As discussed in the introduction, we want to extend SBNs to allow the model structure to depend on the structure being output. In particular, we want the arcs of the model to reflect the same statistical dependencies which are reflected by locality in the output structure. When these arcs connect latent variables, information can be propagated between latent variables, thereby providing an even larger structural domain of locality than that provided by single arcs. However, there is a bias against propagating information through long chains of latent variables. This provides a potentially powerful form of feature induction, which is nonetheless biased toward a notion of locality which is appropriate for the problem.

To extend SBNs for processing arbitrarily long sequences such as the decision sequence $D^1, ..., D^m$, we use dynamic models. This gives us a kind of Dynamic Bayesian Network (DBN). In DBNs, a new set of variables is instantiated for each position in the sequence, but the arcs and weights for these variables are the same as in other positions. The arcs which connect variables instantiated for different positions must be directed forward in the sequence, thereby allowing a temporal interpretation of the sequence. DBNs based on Sigmoid Belief Networks were considered in (Sallans, 2002) in the context of reinforcement learning.

In order to have arcs which reflect locality in the output structure, we need to specify arcs based on the actual outputs of the decision sequence, not based on adjacency in the sequence. We constrain this specification so that a decision can only effect the placement of any arc whose destination is after the decision. This gives us a form of switching model (Murphy, 2002), where each decision switches the model structure used for the remaining decisions. The incoming arcs for a given position are a discrete function of the sequence of decisions which precede that position. For this reason we call our model an "incremental" model, not just a dynamic model. The structure of the model is determined incrementally as the decision sequence proceeds.

Incremental Sigmoid Belief Networks allow the model structure to depend on the output structure without overly complicating the inference of the desired conditional probabilities $P(D^t|D^1, \ldots, D^{t-1})$. At position t in the sequence, the only arcs whose placement are not specified by $D^1, \ldots, D^{t-1}$ have their destinations after t. Also, there are no visible variables after t. Therefore none of the arcs whose placement is not yet known can have any impact on the inference of $P(D^t|D^1, \ldots, D^{t-1})$. This is why in figure 3.1, discussed below, it is not necessary to try to draw the portion of the graph after t. This property of ISBNs allows us to do inference without the need to sum over all possible model structures, which in general would make inference intractable. Note that this property would not hold if we used an undirected graphical model, such as Conditional Random Fields (Lafferty et al., 2001).

3.3 The Probabilistic Model of Structured Prediction

In this section we complete the definition of Incremental Sigmoid Belief Networks for structured prediction. We only consider joint probability models, since they are generally simpler and, unlike history-based conditional models, do not suffer from the label bias problem (Bottou, 1991). Also, in many complex predication tasks, such as phrase structure parsing, all the most accurate models make use of a joint model (Charniak & Johnson, 2005; Henderson, 2004).

We use a history-based probability model, as in equation (2.2), but instead

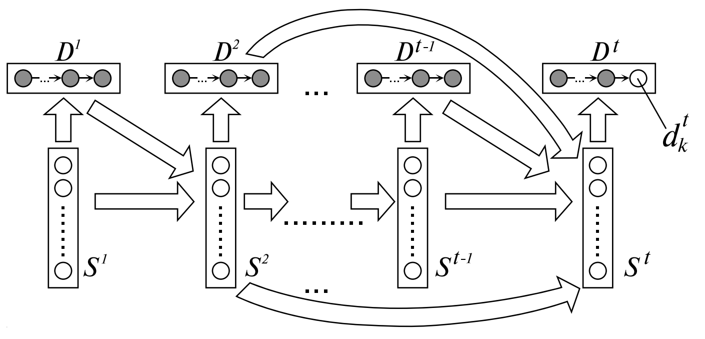


Figure 3.1: ISBN for estimating $P(d_k^t|h(t,k))$.

of treating each D^t as an atomic decision, it is convenient to further split it into a sequence of elementary decisions $D^t = d_1^t, \ldots, d_n^t$:

$$P(D^t|D^1, \ldots, D^{t-1}) = \prod_k P(d_k^t|h(t,k)), \tag{3.4}$$

where $h(t,k)$ denotes the decision history $D^1, \ldots, D^{t-1}, d_1^t, \ldots, d_{k-1}^t$. For example, a decision to create a new node in a labeled output structure can be divided into two elementary decisions: deciding to create a node and deciding which label to assign to it.

An example of the kind of graphical model we propose is illustrated in figure 3.1. It is organized into vectors of variables: latent state variable vectors $S^{t'} = s_1^{t'}, \ldots, s_n^{t'}$, representing an intermediate state at position t', and decision variable vectors $D^{t'}$, representing a decision at position t', where $t' \leq t$. Variables whose value are given at the current decision (t,k) are shaded in figure 3.1, latent and current decision variables are left unshaded.

As illustrated by the arcs in figure 3.1, the probability of each state variable $s_i^{t'}$ depends on all the variables in a finite set of relevant previous state and decision vectors, but there are no direct dependencies between the different variables in a single state vector.

Which previous state and decision vectors are connected to the current state vector is determined by a set of structural relations specified by the model designer. For example, we could select the most recent state where the same output structure node was on the top of the processor's stack, and a decision variable representing that node's label. Each such selected relation has its own distinct weight matrix for the resulting arcs in the graph, but the same weight matrix is used at each position where the relation is relevant.

More formally, we can write the dependency a latent variable component $s_i^{t'}$ on previous latent variable vectors and a decision history as:

$$P(s_i^{t'} = 1|S^1, \ldots, S^{t'-1}, h(t',1)) = \sigma\left(\sum_{r:\exists R_r(t')} \sum_j J_{ij}^r s_j^{R_r(t)} + \sum_k B_{id_k^{R_r(t)}}^{rk}\right), \tag{3.5}$$

where r is the index of relation type, $\{R_1(t') \dots R_m(t')\}$ is the list of positions in the past relevant to the decision considered at time t'. If there is no previous step which is in relation r to the time step t', then the corresponding index is skipped in the summation as declared by the predicate $\exists R_r(t')$. For each relation r there is a weight matrix J^r_{ij}, which determines the influence of jth node in the related previous latent vector $S^{R_r(t)}$ on the distribution of ith node of the considered latent vector S^t. Similarly, $B^{rk}_{id_k^{R_r(t)}}$ defines the influence of the past decision $d_k^{R_r(t)}$ on the distribution of the considered latent vector component $s_i^{t'}$.

In this simplified case we assumed that for each relation r both the previous latent variable vector $s^{R_r(t)}$ and the previous decision vector $D^{R_r(t)}$ are relevant. However, in reality the related decision and the related latent variables often correspond to different time steps. For example, in our preceding discussion, the previous parsing step with the same node on top of the stack is different from the time step where the label of this node was selected. For this reason we could distinguish relations used to define relevant previous latent vectors R from relations used to define relevant previous decision vectors R':

$$P(s_i^{t'}=1|S^1,\dots,S^{t'-1},h(t',1))=\sigma\left(\sum_{r:\exists R_r(t')}\sum_j J^r_{ij}s_j^{R_r(t)}+\sum_{r:\exists R'_r(t')}\sum_k B^{rk}_{id_k^{R'_r(t)}}\right). \quad (3.6)$$

For simplicity, in the remaining part of this chapter we assume that $R = R'$.

In the previous paragraph we defined the conditional distribution of the latent vector components. Now we describe the distribution of the decision vector $D^{t'} = d_1^{t'}, \dots d_n^{t'}$. As indicated in figure 3.1, the probability of each elementary decision $d_k^{t'}$ depends both on the current latent vector $S^{t'}$ and on the previously chosen elementary action $d_{k-1}^{t'}$ from $D^{t'}$. This probability distribution has the normalized exponential form:

$$P(d_k^{t'} = d|S^{t'}, d_{k-1}^{t'}) = \frac{\Phi_{h(t',k)}(d)\exp(\sum_j W_{dj}s_j^{t'})}{\sum_{d'}\Phi_{h(t',k)}(d')\exp(\sum_j W_{d'j}s_j^{t'})}, \quad (3.7)$$

where $\Phi_{h(t',k)}$ is the indicator function of the set of elementary decisions that can possibly follow the last decision in the history $h(t', k)$, and the W_{dj} are the weights of the arcs from the state variables. Φ is essentially switching the output space of the elementary inference problems $P(d_k^{t'} = d|S^{t'}, d_{k-1}^{t'})$ on the basis of the previous decision. For example, in a generative history-based model of natural language parsing, if decision $d_1^{t'}$ was to create a new node in the tree, then the next possible set of decisions defined by $\Phi_{h(t',2)}$ will correspond to choosing a node label, whereas if decision $d_1^{t'}$ was to generate a new word then $\Phi_{h(t',2)}$ will select decisions corresponding to choosing this word.

Note that the idea of incremental specification of model structure can be applied to any Bayesian Network architecture, not just Sigmoid Belief Networks. In particular, such Incremental Bayesian Networks could be built from Factorial

HMMs (Ghahramani & Jordan, 1996). In Factorial HMMs, unlike the Dynamic SBNs considered here, each latent variable s_i^t is conditioned only on a single latent variable s_i^{t-1} in the previous vector S^{t-1}. This difference makes the inference in Factorial HMMs models significantly less computationally expensive than in Dynamic SBNs. Also note that all the dependencies between latent vector components in Factorial HMMs are conveyed via the observables variables (i.e. bottom-up reasoning is crucial for them[2]), which is another significant difference between Dynamic Sigmoid Belief Networks and Factorial HMMs. We could define Incremental Factorial HMMs as an incremental modification of the Factorial HMMs in the same way as ISBNs are defined as an incremental modification of the Dynamic SBNs. Although it should be significantly easier to approximate Incremental Factorial HMMs than ISBNs, we hypothesize that they do not have sufficient expressive power for such complex structured prediction problems as natural language parsing. The primary reason for this is that, in parsing, the decisions at different positions are of different types. Therefore, different representations of the parsing context are needed. This can be accomplished by ISBNs, where the transition matrices J can perform such transformations of the parsing context representation. However, this cannot be done in the Incremental Factorial HMMs, where each latent vector component is conditioned only on a single variable from each previous related vector and, therefore, has a very limited ability to change roles across different positions. To put the argument another way, natural language parsing is sufficiently complex to require the powerful function approximation abilities of SBN hidden vectors, because there is no linguistic motivation for the adequacy of the factorial assumption of Factorial HMMs.

3.4 Approximating Inference in ISBNs

Exact inference with ISBNs is straightforward, but not tractable. It involves a summation over all possible variable values for all the latent variable vectors. The presence of arbitrary arcs between latent variable vectors does not allow us to use efficient belief propagation methods. Even in the case of Dynamic SBNs (i.e. Markovian models), the large size of each individual latent vector would not allow us to perform the marginalization exactly. This makes it clear that we need to develop methods for approximating the inference problems required for structured prediction. Gibbs sampling (Geman & Geman, 1984) is also expensive because of the huge space of variables and the need to resample after making each new decision in the sequence. Thus, we know of no reasonable alternatives to the use of variational methods.

This section is structured as follows. We start by describing the application of the standard mean field approximation to ISBNs and discuss its limitations. Then we propose an approach to overcome these limitations, and two approximation methods. First we show that the neural network computation used in (Henderson, 2003) can be viewed as a mean field approximation with the

[2]See section 3.4.2 for a definition and a discussion of bottom-up reasoning.

added constraint that computations be strictly incremental. Then we relax this constraint to build more accurate but still tractable mean field approximation.

3.4.1 Applicability of Mean Field Approximations

In this section we derive the most straightforward way to apply mean field methods to ISBN. Then we explain why this approach is not feasible for structured prediction problems of the scale of natural language parsing.

The standard use of the mean field theory for SBNs (Saul et al., 1996; Saul & Jordan, 1999) is to approximate probabilities using the value of the lower bound L_V from expression (3.3). To obtain a more accurate bound, as we explained above, the L_V is maximized by choosing the optimal distribution Q. To approximate $P({d^t}_k|h(t,k))$ using the value of L_V, we have to include the current decision d^t_k in the set of visible variables, along with the visible variables specified in $h(t,k)$. Then to estimate the conditional probability $P(d^t_k|h(t,k))$, we need to normalize over the set of all possible value of d^t_k. Thus we need to compute a separate estimate $\max_Q L_V^{t,k}(d)$ for each possible value of $d^t_k = d$:

$$\max_Q L_V^{t,k}(d) = \max_Q \sum_H Q(H^t|h(t,k), d^t_k = d) \ln \frac{P(H^t, h(t,k), d^t_k = d)}{Q(H^t|h(t,k), d^t_k = d)},$$

where $H^t = \{S^1, \dots, S^t\}$. Then $P({d^t}_k = d|h(t,k))$ can be approximated as the normalized exponential of $L_V^{t,k}(d)$ values:

$$\hat{P}(d^t_k = d|h(t,k)) = \frac{\exp(\max_Q L_V^{t,k}(d))}{\sum_{d'} \exp(\max_Q L_V^{t,k}(d'))}. \tag{3.8}$$

It is not feasible to find the optimal distribution Q and mean field methods (Saul et al., 1996; Saul & Jordan, 1999) use an additional approximation to estimate $\max_Q L_V^{t,k}(d)$. Even with this approximation the maximum can be found only by using an iterative search procedure. This means that when decoding estimator (3.8) requires performing this numerical procedure for every possible value of the next decision. Unfortunately, in general this is not feasible, in particular with labeled output structures where the number of possible alternative decisions d^t_k can be large. For natural language parsing, decisions include word predictions, and there can be a very large number of possible next words. Even if we choose not to recompute mean field parameters for all the preceding states $S^{t'}$, but only for the current state S^t (as proposed below), tractability still remains a problem.[3]

In our modifications of the mean field method, we propose to consider the next decision d^t_k as a hidden variable. Then the assumption of full factorizability of $Q(H^t, d^t_k|h(t,k))$ is stronger than in the standard mean field theory because

[3]We conducted preliminary experiments with natural language parsing on very small datasets and even in this setup the method appeared to be very slow and, surprisingly, not as accurate as the modification considered further in this section.

the approximate distribution Q is no longer conditioned on the next decision d_k^t. The approximate fully factorisable distribution $Q(H|V)$ can be written as:

$$Q(H|V) = q_k^t(d_k^t) \prod_{t'i} \left(\mu_i^{t'}\right)^{s_i^{t'}} \left(1 - \mu_i^{t'}\right)^{1-s_i^{t'}}.$$

where $\mu_i^{t'}$ is the free parameter which determines the distribution of state variable i at position t', namely its mean, and $q_k^t(d_k^t)$ is the free parameter which determines the distribution over decisions d_k^t. Importantly, we use $q_k^t(d)$ to estimate the conditional probability of the next decision:

$$P(d_k^t = d|h(t,k)) = q_k^t(d),$$

and the total structure probability is therefore computed as the product of decision probabilities corresponding to its derivation:

$$P(T) = (D^1, \ldots, D^m) = \prod_{t,k} q_k^t(d_k^t). \tag{3.9}$$

3.4.2 A Feed-Forward Approximation

In this section we will describe the sense in which neural network computation can be regarded as a mean field approximation under an additional constraint of strictly feed-forward computation. We will call this approximation the feed-forward approximation. As in the mean field approximation, each of the latent variables in the feed-forward approximation is independently distributed. But unlike the general case of mean field approximation, in the feed-forward approximation we only allow the parameters of every distribution $Q(s_i^{t'}|h(t,k))$ and $Q(d_k^t|h(t,k))$ to depend on the approximate distributions of their parents and require that any information about the distribution of its descendants is not taken into account. This additional constraint increases the potential for a large KL divergence with the true model, but it significantly simplifies the computations.

We start with a simple proposition for general graphical models. Under the feed-forward assumption, computation of the mean field distribution of a node in an ISBN is equivalent to computation of a distribution of a variable corresponding to a sink in the graph of the model, i.e. a node which does not have any outgoing arcs. E.g. node A is a sink in figure 3.2. The following proposition characterizes the mean field distribution of a sink.

Lemma 3.4.1. *The optimal mean field distribution of a sink A depends on the mean field distribution $Q(B)$ of its hidden parents $B = (B_1, \ldots, B_m)$ as*

$$Q(A = a) \propto \exp(E_Q \log P(A = a|B, C)),$$

where Q is the mean field distribution of hidden variables, P is the model distribution, C are visible parents of the node A and E_Q denotes the expectation under the mean field distribution $Q(B)$.

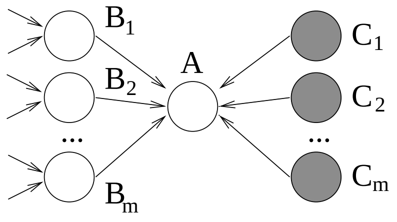


Figure 3.2: A graphical model fragment where variable A is a sink.

Proof. This proposition is straightforward to prove by maximizing the variational bound L_V (3.3) with respect to the distribution $Q(A)$. As we consider maximization of the variational bound L_V by choosing only mean field distribution of the variable A, we can write only the part of L_V dependent on this distribution $Q(A)$:

$$
\begin{aligned}
L_V[A] = & -\sum_a Q(A=a)\log Q(A=a) \\
& + \sum_a Q(A=a) \sum_{b_1\ldots,b_m} \prod_i Q(B_i=b_i) \log P(A=a|B=b_1,\ldots,b_m,C).
\end{aligned}
$$

This maximization problem is constrained, as we need to keep probability distribution normalized $\sum_a Q(A=a)=1$. To perform this maximization we construct the Lagrangian $F[A]$ and its saddle point can be found by solving the system of equations $\nabla_{Q(A),\lambda} F[A]=0$:

$$
-\log Q(A=a) - 1 - \sum_{b_1\ldots,b_m} \prod_i Q(B_i=b_i) \log P(A=a|B=b_1,\ldots,b_m,C) - \lambda = 0
$$

for every a, which gives us exactly the statement of the lemma. □

Now we can use the fact that SBNs have log-linear CPD. By substituting their CPD given in expression (3.1) for P in the lemma statement, we obtain:

$$
Q(S_i=1) = \sigma\Big(\sum_{S_j \in Par(S_i)} J_{ij}\mu_j\Big),
$$

which exactly replicates computation of a feed-forward neural network with the logistic sigmoid activation function. Similarly, we can show that for variables with soft-max CPD, as defined in (3.2), their mean field distribution will be the log-linear function of their parents' means. Therefore minimizing KL divergence under the constraint of feed-forward computation is equivalent to using log-linear functions to compute distributions of random variables given means of their parents.

Now let us return to the derivation of the feed-forward approximation of ISBNs. As we just derived, under the feed-forward assumption, means of the latent vector $S^{t'}$ are given by

$$\mu_i^{t'} = \sigma(\eta_i^{t'}),$$

where $\eta_i^{t'}$ is the weighted sum of the parent variables' means:

$$\eta_i^{t'} = \sum_{r:\exists R_r(t')} \sum_j J_{ij}^r \mu_j^{R_r(t')} + \sum_k B^{rk}_{id_k^{R_r(t')}}, \tag{3.10}$$

as follows from the definition of the corresponding CPD (3.5).

The same argument applies to decision variables, the approximate distribution of the next decision $q_k^t(d)$ is given by

$$q_k^t(d) = \frac{\Phi_{h(t,k)}(d) \exp(\sum_j W_{dj}\mu_j^t)}{\sum_{d'} \Phi_{h(t,k)}(d') \exp(\sum_j W_{d'j}\mu_j^t)}. \tag{3.11}$$

The resulting estimate of the probability of the entire structure is given by (3.9).

This approximation method replicates exactly the computation of the feed-forward neural network model (Henderson, 2003), where the above means $\mu_i^{t'}$ are equivalent to the neural network hidden unit activations. Thus, that neural network probability model can be regarded as a simple approximation to the ISBN graphical model.

In addition to the drawbacks shared by any mean field approximation method, this feed-forward approximation cannot capture bottom-up reasoning. By bottom-up reasoning, we mean the effect of descendants in a graphical model on distributions of their ancestors. For mean field approximations to ISBN, it implies the need to update the latent vector means $\mu_i^{t'}$ after observing a decision d_k^t, for $t' \leq t$. The next section discusses how bottom-up reasoning can be incorporated in the approximate model.

3.4.3 Incremental Mean Field Approximation

In this section we relax feed-forward assumption to incorporate bottom-up reasoning into the approximate model. Again as in the feed-forward approximation, we are interested in finding the distribution Q which maximizes the quantity L_V in expression (3.3). The decision distribution $q_k^t(d_k^t)$ maximizes L_V when it has the same dependence on the latent vector means μ_k^t as in the feed-forward approximation, namely expression (3.7). However, as we mentioned above, the feed-forward computation does not allow us to compute the optimal values of state means $\mu_i^{t'}$.

Optimally, after each new decision d_k^t, we should recompute all the means $\mu_i^{t'}$ for all the latent vectors $S^{t'}$, $t' \leq t$. However, this would make the method intractable for tasks with long decision sequences. Instead, after making each decision d_k^t and adding it to the set of visible variables V, we recompute only the means of the current latent vector S^t. This approach also speeds up computation because, unlike in the standard mean field theory, there is no need to introduce an additional variational parameter for each hidden layer variable s_i^t.

The denominator of the normalized exponential function in (3.7) does not allow us to compute L_V exactly. Instead, we approximate the expectation of its logarithm by substituting S_j^t with their means:[4]

$$E_Q \ln \sum_d \Phi_{h(t,k)}(d) \exp(\sum_j W_{dj} S_j^t) \approx \ln \sum_d \Phi_{h(t,k)}(d) \exp(\sum_j W_{dj} \mu_j^t),$$

where the expectation is taken over the latent vector S^t distributed according to the approximate distribution Q. Unfortunately, even with this assumption there is no analytic way to maximize the approximation of L_V with respect to the means μ_k^t, so we need to use numerical methods. We can rewrite the expression (3.3) as follows, substituting the true $P(H, V)$ defined by the graphical model and the approximate distribution $Q(H|V)$, omitting parts independent of μ_k^t:

$$\begin{aligned} L_V^{t,k} &= \sum_i -\mu_i^t \ln \mu_i^t - (1-\mu_i^t) \ln \left(1-\mu_i^t\right) + \mu_i^t \eta_i^t \\ &+ \sum_{k'<k} \sum_j W_{d_{k'}^t j} \mu_j^t - \ln \left(\sum_d \Phi_{h(t,k')}(d) \exp(\sum_j W_{dj} \mu_j^t) \right), \end{aligned} \tag{3.12}$$

here, η_i^t is computed from the previous relevant state means and decisions as in (3.10). This expression is concave with respect to the parameters μ_i^t, so the global maximum can be found. In section 3.5, where we derive the learning algorithm, we show that the Hessian of this expression can be viewed as the negated sum of a positive diagonal matrix and some covariance matrices, thus implying the concavity of expression (3.12). We use coordinatewise ascent, where each μ_i^t is selected by a line search (Press et al., 1996) while keeping other $\mu_{i'}^t$ fixed.

Though we avoided recomputation of means of the previous states, estimation of the complex decision probability $P(D^t|h(t,k))$ will be expensive if the decision D^t is decomposed in a large number of elementary decisions. As an example, consider a situation in dependency parsing, where after deciding to create a link, the parser might need to decide on the type of the link and, then, predict the part of speech type of the word and, finally, predict the word itself. The main reason for this complexity is the presence of the summation over k' in expression (3.12), which results in expensive computations during the search for an optimal value of μ_i^t. This computation can be simplified by using the means of S^t computed during the estimation of $P(d_{k-1}^t|h(t, k-1))$ as priors for the computation of the same means during the estimation of $P(d_k^t|h(t,k))$. If we

[4]In initial research, we considered the introduction of additional variational parameters associated with every possible value of the decision variable in a way similar to (Saul & Jordan, 1999), but this did not improve the prediction accuracy of the model, and considerably increased the computational time.

denote the means computed at an elementary step (t, k) as $\mu_i^{t,k}$, then for $k = 1$, minimization of $L_V^{t,k}$ can be performed analytically, by setting

$$\mu_i^{t,1} = \sigma(\eta_i^t). \tag{3.13}$$

For $k > 1$, expression (3.12) can be rewritten as:

$$\begin{aligned} L_V^{t,k} =& \sum_i -\mu_i^{t,k} \ln \mu_i^{t,k} - (1 - \mu_i^{t,k}) \ln \left(1 - \mu_i^{t,k}\right) \\ &+ \mu_i^{t,k} \left(\ln \mu_i^{t,k-1} - \ln(1 - \mu_i^{t,k-1})\right) + W_{d_{k-1}^t i} \mu_i^{t,k} \\ &- \ln \left(\sum_d \Phi_{h(t,k-1)}(d) \exp(\sum_j W_{dj} \mu_j^{t,k}) \right). \end{aligned} \tag{3.14}$$

Note that maximization of this expression is done also after computing the last decision K_t for the state t. The resulting means μ^{t,K_t+1} are then used in the computation of $\eta_i^{t'}$ for the relevant future states t', i.e. $t = R_r(t')$ for some r:

$$\eta_i^{t'} = \sum_{r:\exists R_r(t')} \sum_j J_{ij}^r \mu_j^{R_r(t'), K_{R_r(t')}+1} + \sum_k B^{rk}_{i d_k^{R_r(t')}}, \tag{3.15}$$

Concavity of expression (3.14) follows from concavity of (3.12), as their functional forms are different by only a linear term and the presence of summation over the elementary decisions. See the next section where we will show that the Hessian of $L_v^{t,k}$ is negative semidefinite, confirming this statement.

3.5 Learning

We train the models described in sections 3.4.2 and 3.4.3 to maximize the fit of the *approximate* models to the data. We use gradient descent, and a maximum likelihood objective function. In order to compute the derivatives with respect to the model parameters, the error should be propagated back through the structure of the graphical model. For the feed-forward approximation, computation of the derivatives is straightforward, as in neural networks (Rumelhart et al., 1986). But for the mean field approximation, this requires computation of the derivatives of the means μ_i^t with respect to the other parameters in expression (3.14). The use of a numerical search in the mean field approximation makes the analytical computation of these derivatives impossible, so a different method needs to be used to compute their values.

This section is structured as follows. We start by introducing the challenges arising when using maximum likelihood estimation with the incremental mean field algorithm. We explain how these challenges can be overcome by using implicit differentiation, and present the resulting backpropogation algorithm. We conclude the section by discussion appropriateness of the standard regularization techniques and possible need for the alternative regularization when using our variational methods. We should note that this section is important for understanding the challenges posed by learning with an incremental mean field approximation, but familiarity with the material presented in it is not necessary for understanding of subsequent sections and, therefore, it can be safely skipped by the reader.

3.5.1 Computing Gradients for the Incremental Mean Field Approximation

We perform maximum likelihood estimation of the ISBN parameters, using the estimator of the structure probability defined in expression (3.9). We focus on the incremental mean field approximation introduced in section 3.4.3. As we have shown there, estimates of the conditional distribution $q_k^t(d) \approx P(d_k^t = d | h(t,k))$ are dependent on the means $\mu^{t,k}$ computed at the elementary step (t,k) in the same way as the estimates $q_k^t(d)$ in the feed-forward approximation depend on the means μ^t in expression (3.11), i.e.

$$q_k^t(d) = \frac{\Phi_{h(t,k)}(d) \exp(\sum_j W_{dj} \mu_j^{t,k})}{\sum_{d'} \Phi_{h(t,k)}(d') \exp(\sum_j W_{d'j} \mu_j^{t,k})}. \tag{3.16}$$

We use the gradient descent algorithm, so the goal of this section is to describe how to compute derivatives of the log-likelihood

$$\hat{L}(T) = \sum_{t,k} \sum_j W_{d_k^t j} \mu_j^{t,k} - \log \left(\sum_{d'} \Phi_{h(t,k)}(d') \exp(\sum_j W_{d'j} \mu_j^{t,k}) \right)$$

with respect to all the model parameters. The derivatives of $\hat{L}(T)$ with respect to model parameters can be expressed as

$$\frac{d\hat{L}(T)}{dx} = \sum_{d,j} \frac{\partial \hat{L}(T)}{\partial W_{dj}} \frac{dW_{dj}}{dx} + \sum_{t,k,i} \frac{\partial \hat{L}(T)}{\partial \mu_i^{t,k}} \frac{d\mu_i^{t,k}}{dx}, \tag{3.17}$$

where x is any model parameter, i.e. entries of the weight matrices J, B and W. All the terms except for $\frac{d\mu_i^{t,k}}{dx}$ are trivial to compute:

$$\frac{\partial \hat{L}(T)}{\partial W_{dj}} = \sum_{t,k} \mu_j^{t,k} \left(\delta_{d_k^t d} - q_k^t(d) \right), \tag{3.18}$$

$$\frac{\partial \hat{L}(T)}{\partial \mu_i^{t,k}} = (1 - \delta_{k,K_t+1}) \left(W_{d_k^t i} - \sum_d q_k^t(d) W_{di} \right), \tag{3.19}$$

where δ_{ij} is the Kronecker delta. Computation of the total derivatives $\frac{d\mu_i^{t,k}}{dx}$ is less straightforward. The main challenge is that dependence of $\mu_j^{t,k}$ for $k > 1$

on other model parameters cannot be expressed analytically, as we found values of $\mu_j^{t,k}$ by performing numerical maximization of the expression $L_V^{t,k}$ (3.14). In the next several paragraphs we will consider only the case of $k > 1$, but later we will return to the simpler case of $k = 1$, where the computation of derivatives is equivalent to the backpropogation algorithm in standard feed-forward neural networks.

Note that the gradient of the log-likelihood can be easily computed in the standard mean field methods for SBNs (Saul & Jordan, 1999; Saul et al., 1996), even though they also use numeric strategies to find optimal means. There means are selected so as to maximize the variational upper bound L_V (3.3), which is used as the log-likelihood $\hat{L} = L_V$ in their approach. In static SBNs it is feasible to perform complete maximization of the entire $\hat{L}$, which involves multiple backward-forward passes through the structure of the graphical model. This leads to all the derivatives $\frac{d\hat{L}}{d\mu_i}$ being equal to zero. Therefore, no error backpropogation is needed in their case. All the derivatives $\frac{d\hat{L}}{dx}$ can be computed using variational parameters associated with the nodes corresponding to the parameter x. E.g. if x is a weight of an arc then only variational parameters associated with the variables at its ends are needed to compute the derivative. Unfortunately, learning with the incremental mean field approximation proposed in this section is somewhat more complex.

In order to compute derivatives $\frac{d\mu_i^{t,k}}{dx}$ we assume that maximization of $L_V^{t,k}$ is done until convergence, then the partial derivatives of $L_V^{t,k}$ with respect to $\mu_i^{t,k}$ are equal to zero. This gives us a system of linear equations, which describes interdependencies between the current means $\mu^{t,k}$, the previous means $\mu^{t,k-1}$ and the weights W:

$$F_i^{t,k} = \frac{\partial L_V^{t,k}}{\partial \mu_i^{t,k}} = \ln(1-\mu_i^{t,k}) - \ln\mu_i^{t,k} - \ln(1-\mu_i^{t,k-1}) + \ln\mu_i^{t,k-1}$$
$$+ W_{d_{k-1}^t i} - \sum_d \hat{q}_{k-1}^t(d) W_{di} = 0,$$

for $1 < i \leq n$, where $\hat{q}_{k-1}^t$ is the distribution over decisions computed in the same way as q_{k-1}^t (3.16), but using means $\mu_i^{t,k}$ instead of $\mu_i^{t,k-1}$:

$$\hat{q}_{k-1}^t(d) = \frac{\Phi_{h(t,k-1)}(d)\exp(\sum_j W_{dj}\mu_j^{t,k})}{\sum_{d'}\Phi_{h(t,k-1)}(d')\exp(\sum_j W_{d'j}\mu_j^{t,k})}.$$

This system of equations permits the use of implicit differentiation to compute the derivatives $\frac{\partial \mu_i^{t,k}}{\partial z}$, where z can be a weight matrix component W_{dj} or a previous mean $\mu_j^{t,k-1}$ involved in expression (3.14). It is important to distinguish z from x, used above, because x can be an arbitrary model parameter not necessary involved in the expression $L_V^{t,k}$ but affecting the current means $\mu_i^{t,k}$ through $\mu_j^{t,k-1}$. Equally important to distinguish partial derivatives $\frac{\partial \mu_i^{t,k}}{\partial z}$ from the total derivatives $\frac{d\mu_i^t}{dz}$, because the dependency of $\mu_i^{t,k}$ on parameter z can be both direct, through maximization of $L_V^{t,k}$, but also indirect through previous maximization steps (t', k'), where $L_V^{t',k'}$ was dependent on z. The relation between the total and partial derivatives can be expressed as

$$\frac{d\mu_i^{t,k}}{dz} = \frac{\partial \mu_i^{t,k}}{\partial z} + \sum_j \frac{\partial \mu_i^{t,k}}{\partial \mu_j^{t,k-1}} \frac{d\mu_j^{t,k-1}}{dz},$$

meaning that indirect dependencies of $\mu_i^{t,k}$ $(k > 1)$ on parameters z are coming through previous means $\mu_j^{t,k-1}$. We apply the implicit differentiation theorem and obtain the vector of partial derivatives with respect to a parameter z $D_z\mu^t = \{\frac{\partial \mu_1^t}{\partial z}, \ldots, \frac{\partial \mu_n^t}{\partial z}\}$ as

$$D_z\mu^t = -\left(D_{\mu^t} F^{t,k}\right)^{-1} D_z F^{t,k}, \tag{3.20}$$

where $D_{\mu^t} F^{t,k}$ and $D_z F^{t,k}$ are Jacobians:

$$D_{\mu^t} F^{t,k} = \begin{pmatrix} \frac{\partial F_1^{t,k}}{\partial \mu_1^t} & \ldots & \frac{\partial F_1^{t,k}}{\partial \mu_n^t} \\ \ldots & \ldots & \ldots \\ \frac{\partial F_n^{t,k}}{\partial \mu_1^t} & \ldots & \frac{\partial F_n^{t,k}}{\partial \mu_n^t} \end{pmatrix}, \quad D_z F^{t,k} = \begin{pmatrix} \frac{\partial F_1^{t,k}}{\partial z} \\ \ldots \\ \frac{\partial F_n^{t,k}}{\partial z} \end{pmatrix}.$$

Now we derive the Jacobians $D_{\mu^t} F^{t,k}$ and $D_z F^{t,k}$ for different types of parameters z. The matrix $D_{\mu^t} F^{t,k}$ consists of the components

$$\begin{aligned} \frac{\partial F_i^{t,k}}{\partial \mu_j^{t,k}} = & - \frac{\delta_{ij}}{\mu_j^{t,k}(1-\mu_j^{t,k})} - \sum_d \hat{q}_{k-1}^t(d) W_{di} W_{dj} \\ & + \left(\sum_d \hat{q}_{k-1}^t(d) W_{di}\right)\left(\sum_d \hat{q}_{k-1}^t(d) W_{dj}\right), \end{aligned} \tag{3.21}$$

where δ_{ij} is the Kronecker delta. If we consider $W_{\cdot i}$ as a random variable accepting values W_{di} under distribution $\hat{q}_{k-1}^t$, we can rewrite the Jacobian $D_{\mu^t} F^{t,k}$ as the negated sum of a positive diagonal matrix and the covariance matrix $\Sigma_{\hat{q}_{k-1}^t}(W)$. Therefore the matrix $D_{\mu^t} F^{t,k}$ is negative semidefinite.

Note that this matrix is the Hessian for the expression $L_V^{t,k}$ (3.14), which implies concavity of $L_V^{t,k}$ stated previously without proof. Similarly, the Hessian for (3.12) is only different by including output weight covariances for all the previous elementary decision, not only for the last one, and therefore expression (3.12) is also concave.

To conclude with the computation of $\frac{\partial \mu_i^{t,k}}{\partial z}$, we compute $D_{\mu^{t,k-1}} F^{t,k}$ and $D_W F^{t,k}$:

$$\frac{\partial F_i^{t,k}}{\partial \mu_j^{t,k-1}} = \frac{\delta_{ij}}{\mu_j^{t,k-1}(1-\mu_j^{t,k-1})}, \tag{3.22}$$

$$\frac{\partial F_i^{t,k}}{\partial W_{dj}} = \delta_{ij}\delta_{dd_k^t} - \hat{q}_{k-1}^t(d)\left(\delta_{ij} + (W_{di} - \sum_{d'} \hat{q}_{k-1}^t(d')W_{d'i})\mu_j^{t,k}\right). \quad (3.23)$$

Now the partial derivatives $\frac{\partial \mu_i^{t,k}}{\partial W_{dj}}$ and $\frac{\partial \mu_i^{t,k}}{\partial \mu_i^{t,k-1}}$ can be computed by substituting expressions (3.21)-(3.23) into (3.20).

For $k = 1$, $\mu_i^{t,1}$ was shown to be equal to the sigmoid function of the weighted sum of the parents means as defined in (3.13) and (3.15). Therefore, we can compute the partial derivatives of $\mu_i^{t,1}$ with respect to other means and parameters involved in (3.13) and (3.15):

$$\frac{\partial \mu_i^{t,1}}{\partial \mu_j^{t',K_{R_r(t)}+1}} = \sigma'(\eta_i^t) \sum_{r:t'=R_r(t)} J_{ij}^r,$$

$$\frac{\partial \mu_i^{t,1}}{\partial J_{jl}^r} = \delta_{ij}\sigma'(\eta_i^t)[\exists R_r(t)]\mu_l^{R_r(t),K_{R_r(t)}+1},$$

$$\frac{\partial \mu_i^{t,1}}{\partial B_{jd}^{rk}} = \delta_{ij}\sigma'(\eta_i^t)[\exists R_r(t)]\delta_{dd_k^{R_r(t)}},$$

where $[P]$ is the indicator function for the predicate P, and $\sigma'(\eta_i^t) = \sigma(\eta_i^t)\,(1 - \sigma(\eta_i^t))$.

In order to compute $\frac{d\mu_i^{t,k}}{dx}$ in (3.17), derivatives with respect to previous means $\frac{\partial \mu_i^{t,k}}{\partial \mu_i^{t,k-1}}$ are used to propagate the error in a similar way to the neural network backpropogation algorithm (Rumelhart et al., 1986). We denote the total derivative of the approximate log-likelihood with respect to the means of the latent variables as $\epsilon_i^{t,k} = \frac{d\hat{L}(T)}{d\mu_i^{t,k}}$. The incrementality of the mean field algorithm guarantees that latent vectors of means $\mu^{t,k}$ are computed from the means of the previous elementary steps. Therefore, values $\epsilon_i^{t,k}$ can be computed in the opposite order, propagating the information back through the structure. Namely, the recursive formulas would be:

$$\epsilon_i^{t,k} = \frac{\partial \log q_k^t}{\partial \mu_i^{t,k}} + \sum_j \epsilon_j^{t,k+1} \frac{\partial \mu_j^{t,k+1}}{\partial \mu_i^{t,k}}, \quad k \leq K_t\ ,$$

$$\epsilon_i^{t,K^t+1} = \sum_r \sum_{t':t=R_r(t')} \sum_j \epsilon_j^{t',1} \sigma'(\eta_j^{t'}) J_{ji}^r.$$

After computing values ϵ for all the elementary steps (t,k), we can evaluate the derivatives of the model parameters. We start with the output distribution parameters W_{di}:

$$\frac{d\hat{L}(T)}{dW_{di}} = \frac{\partial \hat{L}(T)}{\partial W_{di}} + \sum_{t,k}\sum_j \epsilon^{t,k} \frac{\partial \mu_j^{t,k}}{\partial W_{di}}.$$

The first term here is evaluated as defined in (3.18), the term $\frac{\partial \mu_j^{t,k}}{\partial W_{di}}$ is computed as explained above.

Finally, the total derivatives of the log-likelihood with respect to the parameters J_{ij}^t and B_{id}^{rk} are found as follows

$$\frac{d\hat{L}(T)}{dJ_{ij}^r} = \sum_t \mu_j^t \sum_{t':t=R_r(t')} \epsilon_i^{t',1} \sigma'(\eta_i^{t'}),$$

$$\frac{d\hat{L}(T)}{dB_{id}^{rk}} = \sum_t \delta_{d_k^t d} \sum_{t':t=R_r(t')} \epsilon_i^{t',1} \sigma'(\eta_i^{t'}).$$

3.5.2 Computational Complexity

If we assume that the number of iterations in the step-wise descent, when optimizing expression (3.14), does not depend on the latent variable vector size n then the asymptotic complexity of the learning algorithm is dominated by the matrix inversion. If we additionally assume that the number of iterations over the training set also does not depend on n and we choose to use a trivial algorithm for matrix inversion[5] then the asymptotic complexity can be written $O(n^3 D)$, where D is the number of parser decisions in the training set. The use of more advanced algorithms for matrix inversion can reduce the asymptotic complexity, e.g. for the CoppersmithWinograd algorithm (Cohn et al., 2005) to $O(n^{2.4} D)$, but they are unlikely to speed up the model in practice because of the large multiplicative constant hidden in the O notation. Both of the assumptions made appear to be reasonable, as in the range of n we considered in our experiments, there was no strong dependence of iteration number on n.

The learning algorithm can be parallelized, as with any stochastic gradient learning method, by performing the computation of the gradients at different nodes and then exchanging with the information about the needed updates of the weight vector. In this case the gradient algorithm will not be purely stochastic, as the model will not be updated after processing each instance, but sufficiently frequent exchange of the update information will result in a method with convergence properties similar to the standard stochastic gradient descent.

3.5.3 Regularization

The standard mean field approach considered in (Saul & Jordan, 1999) maximized L_V (3.3) during learning, because L_V was used as an approximation of the log-likelihood of the training data. L_V is actually the sum of the log-likelihood and the negated KL divergence between the approximate distribution $Q(H|V)$ and the SBN distribution $P(H|V)$. Thus, maximizing L_V will at the same time direct the SBN distribution toward configurations which have a lower

[5]In our experiments we used the Gaussian elimination algorithm.

approximation error. It is important to distinguish this regularization of the approximate distribution from the Gaussian priors on the SBN parameters, which can be achieved by simple weight decay. We believe that these two regularizations should be complementary. However, in our version of the mean field method the approximate distributions of hidden decision variables q_k^t are used to compute the data likelihood (3.9) and, thus, maximizing this target function will not automatically imply KL divergence minimization. Application of an additional regularization term corresponding to minimization of the KL divergence might be beneficial for our approach, and it could be a subject of further research. In our current experiments, we used standard weight decay, which regularizes the SBN distribution with a Gaussian prior over weights.

3.6 Decoding

ISBNs define a probability model which does not make any a-priori assumptions of independence between any decision variables. As we discussed in section 3.2, the use of relations based on the partial output structure makes it possible to take into account statistical interdependencies between decisions closely related in the output structure, but separated by arbitrarily many decisions in the input structure. In general, this property leads to the complexity of complete search being exponential in the number of processor steps. Fortunately, for many problems, such as natural language parsing, efficient heuristic search methods are possible. In the following chapters 4 and 5 we will describe the search strategies used in out methods and show that efficient approximate search leads to good parsing accuracy.

3.7 Artificial Experiment

In this section we perform evaluation of the ISBN model on the artificial data. We will show that learning the proposed incremental mean field approximation (IMF method) results in a sufficiently accurate model, and that this model is more accurate than the feed-forward neural network approximation (NN method) originally proposed in (Henderson, 2003) and rederived in section 3.4.2 as a variational approximation. We perform an artificial experiment where the true distribution is generated by a dynamic SBN, and compare both of the approximate models *learned* on this artificial data. In the next chapter we consider a real problem, constituent parsing of natural language, where we compare our approximations with state-of-the-art models.

In order to have an upper bound for our artificial experiments, we do not consider incremental models, but instead use a dynamic Sigmoid Belief Network, a first order Markov model, and consider a sequence labeling task. This simplification allowed us to use Gibbs sampling from a *true* model as an upper bound of accuracy. We generated the training data from random dynamic SBNs of the following type: first a label Y^t is sampled from the dis-

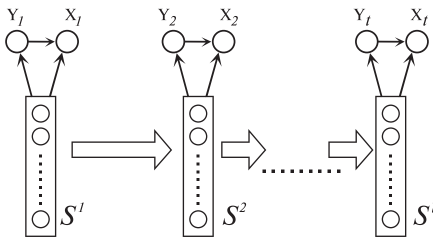


Figure 3.3: Dynamic SBN used in artifical experiments.

tribution $P(Y^t|S^t)$ as in (3.7), then an input element X^t is sampled from the distribution $P(X^t|Y^t, S^t)$, and finally the next latent state vector is sampled from $P(S^{t+1}|S^t, X^t, Y^t)$. A graphical representation of this dynamic model is shown in figure 3.3. Different weight matrices were used in the computation of $P(X^t|Y^t, S^t)$ for each value of the label Y^t. It is easy to see that this model is a special case of the ISBN graphical model, namely figure 3.1 with removed non-Markovian dependencies. The state size was set to 5, the number of possible labels to 6, and the number of distinct input elements to 8. We performed 10 experiments.[6] For each of the experiments, we trained both IMF and NN approximations on training sequence of 20,000 elements, and tested them on another 10,000 elements. Weight-decay and learning rate were reduced through the course of the experiments whenever accuracy on the development set went down. Beam search with a beam of 10 was used during testing. The IMF methods achieved average error reduction of 27% with respect to the NN method, where accuracy of the Gibbs sampler was used as an upper bound (average accuracies of 80.5%, 81.0%, and 82.3% for the NN, IMF, and sampler, respectively).

The IMF approximation performed better than the NN approximation on 9 experiments out of 10 (statistically significant in 8 cases). These results suggest that the IMF method leads to a much more accurate model when the true distribution is defined by a dynamic SBN. In addition, the average relative error reduction of even the NN approximation over the unigram model exceeded 60% (the unigram model accuracy was 77.4% on average), which suggests that both approximations are sufficiently accurate and learnable.

3.8 Summary of the Chapter

In this chapter we described the class of models for structured prediction problems, Incremental Sigmoid Belief Networks. These graphical models allow the structure of the model to be dependent on the output structure, which allows the induction of latent variables with a structural locality bias appropriate for the

[6]We preselected these 10 models to avoid random dynamic SBNs with trivial distributions. We excluded SBNs for which unigram model accuracy was within 3% of the Gibbs sampler accuracy, and where accuracy of the Gibbs sampler did not exceed 70%. All these constants were selected before conducting the experiments.

domain. Exact inference with this class of graphical models is not tractable, but we derive two tractable approximations. First, it is shown that the feed-forward neural network of (Henderson, 2003) can be considered as a simple variational approximation to ISBNs. Second, a more accurate but still tractable approximation based on mean field theory is proposed.

Both approximation models are empirically evaluated. Artificial experiments were performed, where both approximations significantly outperformed a baseline. The incremental mean field method achieved average relative error reduction of about 27% over the neural network approximation, demonstrating that it is a significantly more accurate approximation.

Chapter 4

ISBN Constituent Parsing Model

In this chapter we describe the application of ISBNs to constituent parsing. We compare two approximations, the feed-forward approximation equivalent to the neural network model SSN (Henderson, 2003), and the incremental mean field approximation proposed in the previous chapter. In the previous chapter we have shown that the incremental mean field approximation is indeed more accurate approximation than the neural network approximation, and also that it is beneficial to use this approximation if ISBNs are an appropriate model for the considered problem. In this chapter we show that the mean field approximation results in a more accurate model of parsing than the neural network. This suggests that ISBNs are a good abstract model for natural language parsing.

The chapter is structured as follows. The first half of this chapter is dedicated to the description of the ISBN model for constituent parsing and the parsing algorithm. This is mostly background material, because we reused the parsing order and the pattern of interconnections from the SSN parser, we present it here for completeness. In the second half we describe our experiments, where we compare the incremental mean field approximation with the neural network.

4.1 Parsing Algorithm

The ISBN model considered in this chapter uses a modification of the predictive LR order (Soisalon-Soininen & Ukkonen, 1979) proposed in (Henderson, 2003). In this ordering a parser decides to introduce a node into the parser tree after the entire subtree rooted at the node's first child has been fully constructed. Then the subtrees rooted at the remaining children of the node are constructed in their left-to-right order. Though predictive LR is a form of left-corner order (Rosenkrantz & Lewis, 1970), unlike the standard left corner order, where when introducing a node the parser chooses all the children of this node, in the predictive LR order no new children are chosen when introducing the node.

The state of the parser is defined by the current stack S of nodes (terminal or non-terminals), the queue I of remaining input words (terminals) and the partial structure. The parser starts with an artificial *root* element in the stack S and terminates when it reaches a configuration with an empty queue I and with the artificial root element in the stack. The algorithm uses 3 main types of decisions:

1. The decision $\mathbf{Shift}_w$ shifts the word w from the queue to the stack.
2. The decision $\mathbf{Project}_Y$ replaces the current top of the stack X with a new node Y, and specifies that Y is the parent of X in the partial structure.
3. The decision **Attach** removes the current top of the stack X and specifies that element Y under the top of the stack is the parent of X.

These three types of decisions are sufficient to parse any constituent tree. A sequence of decisions and associated stack states of the parser for an example parse tree are presented in figure 4.1. Note that the shown stack states correspond to the parser state before the corresponding decision. An edge of the tree introduced by a decision is denoted by the corresponding number. A detailed example of parsing a sentence with this parsing algorithm is presented in appendix A along with the corresponding configurations of the graphical model.

Henderson (Henderson, 2003) extends this parsing strategy to perform specific treatment of the *Chomsky adjunction* structures. Chomsky adjunction structures are structures of the form $(X(Y\ldots)(X\ldots))$, where Y is said to be *Chomsky adjoined* to X. They are replaced in this parsing order with a structure of the form $(X(_{mod}Y\ldots)\ldots)$, where *mod* corresponds to a special link in the constituent tree and X is called a "modifier" parent of Y. This link is introduced by an additional decision *Modify*:

4. The decision **Modify** removes the current top of the stack Y and specifies that element X under the top of the stack is the "modifier" parent of Y.

Another modification introduced in (Henderson, 2003) is in processing of frequent sub-structures of the form $(X(Y\ldots))$, where X is a non-branching node in the tree. For frequent pairs of X and Y appearing in sub-structures of these form (e.g., pairs (S, VP), (S, NP) in the Penn Treebank corpus) a new composed constituent labeled X-Y are introduced.[1]

4.2 Model Structure

As was discussed in section 3.3, it is convenient to split parser decisions into elementary decisions, to overcome sparsity and avoid expensive summation over the set of all the possible next decisions in expression 3.11 for the neural-network approximation, and in expressions 3.16 and 3.14 for the incremental

[1]Note that all these modifications do not affect parser evaluation: reverse transformations were applied before performing the evaluation.

Decisions	Stack S
1. Shift$_{\text{Bill/N}}$	[root]
2. Project$_{\text{NP}}$	[root,Bill/N]
3. Project$_{\text{S}}$	[root,NP]
4. Shift$_{\text{sells/V}}$	[root,S]
5. Project$_{\text{VP}}$	[root, S, sells/V]
6. Shift$_{\text{fresh/J}}$	[root,S,VP]
7. Project$_{\text{NP}}$	[root,S,VP,fresh/J]
8. Shift$_{\text{oranges/N}}$	[root,S,VP,NP]
9.–12. Attach	[root,S,VP,NP,oranges/N],...,[root,S]

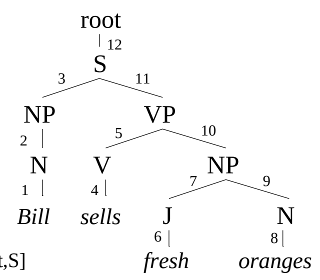


Figure 4.1: Derivation for a constituent parse tree.

mean field approximation. The decision $\textbf{Project}_Y$ is split into two elementary decisions: the decision to project (i.e. introduce a node), where the probability $P(Project|h(t,1))$ is estimated, and the decision to select a label Y, where the probability $P(Project_Y|h(t,1), Project)$ is computed. $h(t,1)$ denotes, as previously, the set of parser actions before the current step t. We focus in this chapter on generative models, therefore shifting a word from the queue involves estimation of the probability of this word given the current context. Therefore, the decision $\textbf{Shift}_w$ is also split into three elementary decisions: the decision to shift a word, prediction of the PoS tag of the word and prediction of the word w itself.[2] We will refer to this sequence of elementary decisions as a composite decision.

As was defined in expression (3.6), the probability of each state variable s^t_j in ISBN depends on all the latent variables in a finite subset of previous relevant states $R(t) = R_1(t) \ldots R_n(t)$ and it also depends on a set of previous relevant decisions made at steps $R'(t) = R'_1(t) \ldots R'_{n'}$.

In the ISBN model we use the same sets of relations $R(t)$ and $R'(t)$ as the sets of interconnections in the SSN neural network of (Henderson, 2003). Namely, the set $R(t)$ consists of the following 4 types of relations:

1. *Stack Context*: the last previous state with the same stack S.

2. *Sub-Top of S*: the last previous state where the node under the current top of the stack was on top of the stack.

3. *Left Child of Top of S*: the last previous state where the leftmost child of the current stack top was on top of the stack.

4. *Right Child of Top of S*: the last previous state where the rightmost child of the current stack top was on top of the stack.

[2]Note, that the parser considered in the experiments takes tagged input, i.e. all the words are annotated with PoS tags on a preprocessing step, therefore, the model does not need to search through all the possible PoS tags.

These relations were motivated by linguistic considerations and many of them have already been found useful in parsing models (Johnson, 1998; Roark & Johnson, 1999). Also this set of relations ensures that the immediately preceding state is always included somewhere in the set of connected states. This requirement ensures that information, at least theoretically, can pass between any two states in the decision sequence, thereby avoiding any hard independence assumptions.

CPD of the latent variables S^t is conditioned on the composite decisions made at the parsing steps $R'(t)$:

1. *Previous*: the previous step $t-1$.
2. *Top*: the step at which current top of the stack S was shifted (if it is a terminal) or introduced (if non-terminal).
3. *Last Shift*: the step at which the last terminal was shifted.
4. *Left Terminal of Top of S*: the step when the leftmost terminal dominated by the current stack top was shifted.

Equivalently, observing the types of the decisions made at these steps, we can define the set of these explicit features as

1. the type of the previous decision (**Shift**, **Project**, **Attach** or **Modify**), the projected constituent label if the decision was **Project**, or the PoS tag and the lexical form of the word if the decision was **Shift**;
2. the constituent label of the current top of the stack S if it is a non-terminal, the PoS tag and the lexical form if it is a terminal;
3. the PoS tag and the lexical form of the last shifted word (if any);
4. the PoS tag and the lexical form of the leftmost terminal dominated by the current stack top (if any).

An example of incremental construction of the ISBN model when parsing a sentence is represented in appendix A.

4.3 Decoding

As we discussed in section 3.6 the complexity of the exact search with unrestricted ISBN model is exponential in length of the derivation. However, the chosen left corner parsing order permits to perform very aggressive pruning without much loss of the accuracy. As proposed in (Henderson, 2003), we use the best-first search with the search space pruned in two different ways. First, only a fixed number of the most probably partial derivations are pursued after each word shift operation. Secondly, the branching factor at each decision is limited. In the experiments presented in this chapter, we used the post-shift beam width of 10 and the branching factor of 5.

	R	P	F_1
(Bikel, 2004)	87.9	88.8	88.3
(Taskar et al., 2004)	89.1	89.1	89.1
NN method	**89.1**	**89.2**	**89.1**
(Turian & Melamed, 2006)	89.3	89.6	89.4
IMF method	**89.3**	**90.7**	**90.0**
(Charniak, 2000)	90.0	90.2	90.1

Table 4.1: Constituent parsing: percentage labeled constituent recall (R), precision (P), combination of both (F_1) on the testing set.

4.4 Experiments

In this section we evaluate the two approximations to ISBNs discussed in the previous chapter, the feed-forward method equivalent to the neural network of (Henderson, 2003) (NN method) and the proposed incremental mean field method (IMF method). The hypothesis we wish to test here is that the more accurate approximation of ISBNs will result in a more accurate model of constituent structure parsing. If this is true, then it suggests that ISBNs are a good abstract model of the nature of natural language parsing.

We used the Penn Treebank WSJ corpus (Marcus et al., 1993) to perform the empirical evaluation of the considered approaches. It is expensive to train the IMF approximation on the entire WSJ corpus, so instead we use only sentences of length at most 15, as in (Taskar et al., 2004) and (Turian & Melamed, 2006). The standard split of the corpus into training (sections 2–22, 9,753 sentences), validation (section 24, 321 sentences), and testing (section 23, 603 sentences) was performed.[3]

As in (Henderson, 2003; Turian & Melamed, 2006) we used a publicly available tagger (Ratnaparkhi, 1996) to provide the part-of-speech tag for each word in the sentence. For each tag, there is an unknown-word vocabulary item which is used for all those words which are not sufficiently frequent with that tag to be included individually in the vocabulary. We only included a specific tag-word pair in the vocabulary if it occurred at least 20 time in the training set, which (with tag-unknown-word pairs) led to the very small vocabulary of 567 tag-word pairs. Many of these 567 tag-word pairs corresponded to tag-word pairs, but they also included other words frequent in the WSJ corpus, such as “remain”, “prices” and “company”.

During parsing with both the NN method and the IMF method, we used beam search with a post-word beam of 10. Increasing the beam size beyond this value did not significantly affect parsing accuracy. For both of the models, the

[3]Training of our IMF method on this subset of WSJ took less than 6 days on a standard desktop PC. The long training times on the entire WSJ would not allow us to tune the model parameters properly, which would have increased the randomness of the empirical comparison, although it would be feasible for building a system. Note also that preliminary experiments suggested that the IMF method outperforms the NN method more significantly on longer sentences. Therefore, this set-up is biased in favour of the NN methods.

state vector size of 40 was used. All the parameters for both the NN and IMF models were tuned on the validation set. A single best model of each type was then applied to the final testing set.

Table 4.1 lists the results of the NN approximation and the IMF approximation,[4] along with results of different generative and discriminative parsing methods (Bikel, 2004; Taskar et al., 2004; Turian & Melamed, 2006; Charniak, 2000) evaluated in the same experimental setup.[5]

The IMF model improves over the baseline NN approximation, with an error reduction in F-measure exceeding 8%. This improvement is statically significant.[6] The IMF model achieves results which do not appear to be significantly different from the results of the best model in the list (Charniak, 2000). Note that, as we discussed in section 2, the model of (Turian & Melamed, 2006) is similar to ISBNs in that it also performs feature induction, constructing complex features from elementary ones, and does not make any explicit independence assumptions.

These experimental results suggest that ISBNs are an appropriate model for structure prediction. Even approximations such as those tested here, with a very strong factorizability assumption, allow us to build quite accurate parsing models.We believe this provides strong justification for work on more accurate approximations of ISBNs.

4.5 Discussion

The results obtained in this chapter place ISBNs amongst the most accurate latent variable models for natural language parsing. The difference in the experimental set-up does not allow for direct comparison of our best approximation (IMF) with the most accurate LA-PCFG model (Petrov et al., 2006; Petrov & Klein, 2007). Nevertheless, we can observe that the results of LA-PCFGs are comparable with the results of the NN model (Henderson, 2003),[7] whereas the IMF approximation achieves significantly better results that the NN approximation in our experiments. In order to construct an accurate statistical model Petrov et al. had to use the split-and-merge technique: iterative selec subsets of constituents to split and subsets of constutuents to merge them back by observing expected changes in the likelihood. The authors argued that both splitting and merging were crucial to achieve competitive performance. This indeed is expected to be crucial for the models with atomic latent labels, such

[4]All our constituent parsing results in this and the following chapters are computed with the evalb program following the standard criteria in (Collins, 1999).

[5]The model of (Bikel, 2004) is a reimplementation of (Collins, 1999) generative parser. The results of (Charniak, 2000) parser trained and tested on sentences ≤ 20 were originally reported in (Turian & Melamed, 2006).

[6]We measured significance of all the parsing experiments in this thesis with the randomized significance test (Yeh, 2000).

[7]Again, note that inference algorithms for LA-PCFGs(Petrov et al., 2006; Petrov & Klein, 2007) essentially directly optimize the F_1 score, whereas MAP decoding used for the neural network parser in (Henderson, 2003) does not explicitly target the F_1 score. Decoding criteria and corresponding algorithms will be discussed in detail in chapter 8.

size	5	10	20	30	40
F_1	80.0	85.8	87.7	88.8	89.3

Table 4.2: Constituent parsing: F_1 on the development set with different latent vector sizes.

as LA-PCFGs. There, splitting of a constituent corresponds to dividing occurrences of the constituent in the training data into the corresponding number of sub-classes, and the estimates of decision probabilities for each of the induced latent constituents will be based only on statistics of the corresponding subset of occurrences. Therefore, over-splitting leads to the use of unreliable statistics. In contrast, in the models which use composite latent labels, such as ISBNs, use of a high dimensional latent space, does not necessarily lead to performance degradation. Unlike LA-PCFGs, these models do not attempt to find disjoint decomposition of constituent labels (or decisions) into sub-classes, but rather annotate each constituent (or decision) with a list of latent features characterizing the parsing context. Though sparsity in the feature space may lead to performance degradation, but this can be tackled by using priors on the paramters. We believe that this property constitutes a major advantage of models with composite latent representations, as they do not require careful tuning of their latent parameter dimensionality (or cardinality of the set of admissable values).

In order to support the above statement we performed a set of experiments with the IMF approximation on the training set, where we varied the size of the latent vector. Results of these experiments are presented in table 4.2. These results suggest that the accuracy is monotonically improving when increasing the size of the latent vector, at least in the region where learning and testing of the model is practical. Therefore we can conclude that the model which uses composite annotations does not suffer from this sparsity problem and does not require ad-hoc techniques for selecting of an optimal latent space size.

Another potential problem for learning ISBNs is non-convexity of the optimized function. Following (Henderson, 2003), we tackle this problem by using an 'annealing' procedure: we start with large initial regularization which is reduced during training. This procedure reduces the variance across different initialization seeds. Difference in results on the development set did not normally exceed 0.2%, which suggests that search is not a major problem for our model.

4.6 Summary of the Chapter

This chapter considered evaluation of the Incremental Sigmoid Belief Networks and their approximations on the constituent parsing task. In the previous chapter we considered two approximation: first, we have shown there that the feed-forward neural network of (Henderson, 2003) is a simple approximation to ISBNs, second, a more accurate but still tractable approximation based on mean field theory was proposed. In the previous chapter, we only evaluated the approximations in artificial experiments, where the mean field method achieved average relative error reduction of about 27% over the neural network approximation, demonstrating that it is a more accurate approximation. In this chapter, both approximations are applied to the natural language parsing task, and the mean field method again demonstrated significantly better results. These results are non-significantly different from the results of one of the most accurate history-based probabilistic model of constituent parsing (Charniak, 2000). The fact that a more accurate approximation leads to a more accurate parser suggests that the ISBNs proposed here are a good abstract model for structure prediction. This empirical result motivates further research into more accurate approximations of ISBNs.

Chapter 5

ISBN Dependency Parsing Model

As we discussed in chapter 2, dependency parsing has been a topic of active research in natural language processing during the last several years. The CoNLL-2006 shared task (Buchholz & Marsi, 2006) made a wide selection of standardized treebanks for different languages available for the research community and allowed for easy comparison between various statistical methods on a standardized benchmark. One of the surprising things discovered by this evaluation was that the best results were achieved by methods which were quite different from state-of-the-art models for constituent parsing, e.g. the deterministic parsing method of (Nivre et al., 2006) and the minimum spanning tree parser of (McDonald et al., 2006). All the most accurate dependency parsing models were fully discriminative, unlike constituent parsing where all the state-of-the-art methods have a generative component (Charniak & Johnson, 2005; Henderson, 2004; Collins, 2000). Another surprising thing is the lack of latent variable models among the methods used in the shared task. Latent variable models would allow complex features to be induced automatically, which would be highly desirable in multilingual parsing, where manual feature selection might be very difficult and time consuming, especially for languages unknown to the parser developer.

In this chapter we propose a generative latent variable model for dependency parsing. It is based on Incremental Sigmoid Belief Networks (ISBNs), a class of directed graphical model described in chapter 3. In the previous chapter 4 we demonstrated that ISBNs achieve competitive results on the constituent parsing task. As we discussed there, computing the conditional probabilities which we need for parsing is in general intractable with ISBNs, but they can be approximated efficiently in several ways. In particular, the neural network constituent parsers in (Henderson, 2003) and (Henderson, 2004) can be regarded as coarse approximations to their corresponding ISBN model.

We build an ISBN model of dependency parsing using the parsing order proposed in (Nivre et al., 2004). However, instead of performing deterministic parsing as in (Nivre et al., 2004), we use this ordering to define a generative history-based model, by integrating word prediction operations into the set of parser actions. Then we propose a simple, language independent set of relations which determine how latent variable vectors are interconnected by conditional dependency arcs in the ISBN model. ISBNs also condition the latent variable vectors on a set of explicit features, which we vary in the experiments.

In experiments we evaluate both the performance of the ISBN dependency parser compared to previous work, and the ability of the ISBN model to induce complex history features. Our model achieves state-of-the-art performance on the languages we test, significantly outperforming the model of (Nivre et al., 2006) on two languages out of three and demonstrating about the same results on the third. In order to test the model's feature induction abilities, we train models with two different sets of explicit conditioning features: the feature set individually tuned by Nivre et al. for each considered language, and a minimal set of local features. These models achieve comparable accuracy, unlike with the discriminative SVM-based approach of (Nivre et al., 2006), where careful feature selection appears to be crucial. We also conduct a controlled experiment where we used the tuned features of (Nivre et al., 2006), but disable the feature induction abilities of our model by elimination of the arcs connecting latent state vectors. This restricted model achieves far worse results, showing that it is exactly the capacity of ISBNs to induce history features which is the key to its success. It also motivates further research into how feature induction techniques can be exploited in discriminative parsing methods.

We analyze how the relation accuracy changes with the length of the head-dependent relation, demonstrating that our model very significantly outperforms the state-of-the-art baseline of (Nivre et al., 2006) on long dependencies. Additional experiments suggest that both feature induction abilities and use of the beam search contribute to this improvement.

We participated with this parser in the CoNLL-2007 shared task on dependency parsing. The model achieved the third overall results, which were only 0.4% worse than results of the best participating system. It is also important to note, that it achieved the best result among single model parsers: the parsers which achieved better results (Hall et al., 2007a; Nakagawa, 2007) used combinations of models. We will also study parser efficiency on the datasets from the shared task.

As we discussed in detail in chapter 3, the fact that our model defines a probability model over parse trees, unlike the previous state-of-the-art methods (Nivre et al., 2006; McDonald et al., 2006), makes it easier to use this model in applications which require probability estimates, e.g. in language processing pipelines. Also, as with any generative model, it may be easy to improve the parser's accuracy by using discriminative retraining techniques (Henderson, 2004) or data-defined kernels which we will describe in the next chapter, with or even without introduction of any additional linguistic features. In addition, there are some applications, such as language modeling, which require generative models. Another advantage of generative models is that they do not suffer from the label bias problems (Bottou, 1991), which is a potential problem for conditional or deterministic history-based models, such as (Nivre et al., 2004).

In the remainder of this chapter, we define the generative parsing model, based on the algorithm of (Nivre et al., 2004), and propose an ISBN for this model. The empirical part of the chapter then evaluates both the overall accuracy of this method and the importance of the model's capacity to induce features.

5.1 The Dependency Parsing Algorithm

We used the pseudo-projective transformation introduced in (Nivre & Nilsson, 2005) to cast non-projective parsing tasks as projective. Following (Nivre et al., 2006), the encoding scheme called *HEAD* in (Nivre & Nilsson, 2005) was used to encode the original non-projective dependencies in the labels of the projectivized dependency tree. Reverse transformation was applied after the parsing.

We use the parsing strategy for projective dependency parsing introduced in (Nivre et al., 2004), which is similar to a standard shift-reduce algorithm for context-free grammars (Aho et al., 1986). It can be viewed as a mixture of bottom-up and top-down parsing strategies, where left dependencies are constructed in a bottom-up fashion and right dependencies are constructed top-down. In this section we describe the algorithm and explain how we use it to define our history-based probability model. For further details on the parsing algorithm we refer the reader to (Nivre et al., 2004).

In this thesis, as in the CoNLL-2006 and CoNLL-2007 shared tasks, we consider labeled dependency parsing. The state of the parser is defined by the current stack S, the queue I of remaining input words and the partial labeled dependency structure constructed by previous parser decisions. The parser starts with an empty stack S and terminates when it reaches a configuration with an empty queue I. The algorithm uses 4 types of decisions:

1. The decision **Left-Arc**$_r$ adds a dependency arc from the next input word w_j to the word w_i on top of the stack and selects the label r for the relation between w_i and w_j. Word w_i is then popped from the stack.

2. The decision **Right-Arc**$_r$ adds an arc from the word w_i on top of the stack to the next input word w_j and selects the label r for the relation between w_i and w_j.

3. The decision **Reduce** pops the word w_i from the stack.

4. The decision **Shift**$_{w_j}$ shifts the word w_j from the queue to the stack.

Unlike the original definition in (Nivre et al., 2004) the **Right-Arc**$_r$ decision does not shift w_j to the stack. However, the only thing the parser can do after a **Right-Arc**$_r$ decision is to choose the **Shift**$_{w_j}$ decision. This subtle modification does not change the actual parsing order, but it does simplify the definition of our graphical model, as explained in section 5.2. A derivation and associated

Decisions	Stack S / Queue I
1. Shift$_{\text{Bill}}$	[root] / [Bill, ...]
2. Left−Arc$_{\text{sbj}}$	[root, Bill] / [sells,...]
3. Right−Arc$_{\text{\_}}$	[root] / [sells, ...]
4. Shift$_{\text{sells}}$	[root] / [sells, ...]
5. Shift$_{\text{fresh}}$	[root, sells] / [fresh, ...]
6. Left−Arc$_{\text{nmod}}$	[root, sells, fresh] / [oranges]
7. Right−Arc$_{\text{obj}}$	[root, sells] / [oranges]
8. Shift$_{\text{oranges}}$	[root, sells] / [oranges]
9. Reduce	[root, sells, oranges] / []
10. Reduce	[root,sells] / []

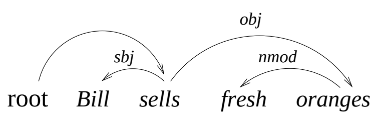


Figure 5.1: Derivation for a dependency parse tree.

stack and queue states of the parser for an example parse tree are presented in figure 5.1. A detailed example of parsing a sentence with this parsing algorithm is presented in appendix B along with the corresponding configurations of the graphical model.

Since we use a generative model, the action $\mathbf{Shift}_{w_j}$ also predicts the next word in the queue I, w_{j+1}, thus the $P(Shift_{w_j}|D^1, \ldots, D^{t-1})$ is a probability both of the shift operation and the word w_{j+1} conditioned on current parsing history. In preliminary experiments, we also considered look-ahead, where the word is predicted earlier than it appears at the head of the queue I, and "anti-look-ahead", where the word is predicted only when it is shifted to the stack S. Early prediction allows conditioning decision probabilities on the words in the look-ahead and, thus, speeds up the search for an optimal decision sequence. However, the loss of accuracy with look-ahead was quite significant. The described method, where a new word is predicted when it appears at the head of the queue, led to the most accurate model and quite efficient search. The anti-look-ahead model was both less accurate and slower.

As discussed in section 3, instead of treating each decision D^t in the history-based model as an atomic decision, we split it into a sequence of elementary decisions $D^t = d_1^t, \ldots, d_n^t$ as in expression 3.4. We split **Left-Arc**$_r$ and **Right-Arc**$_r$ each into two elementary decisions: first, the parser decides to create the corresponding arc, then, it decides to assign a relation r to the arc. Similarly, we decompose the decision $\mathbf{Shift}_{w_j}$ into an elementary decision to shift a word and a prediction of the word w_{j+1}. In our experiments we use datasets from the CoNLL-2006 and CoNLL-2007 shared tasks, which provide additional properties for each word token, such as its part-of-speech tag and some fine-grain features. This information implicitly induces word clustering, which we use in our model: first we predict a part-of-speech tag for the word, then a set of word features, treating feature combination as an atomic value, and only then a particular word form. This approach allows us to both decrease the effect of sparsity and to avoid normalization across all the words in the vocabulary, significantly reducing the computational expense of word prediction.

5.2 An ISBN for Dependency Parsing

In this section we define the ISBN model we use for dependency parsing. To do that we need to described the sets of relevant previous decisions and relevant previous latent vectors connected to the current latent vector.

We start with the decisions. The precise set of decision variables which are connected in this way can be adapted to a particular language. As long as these connected decisions include all the new information about the parse, the performance of the model is not very sensitive to this choice. This is because ISBNs have the ability to induce their own complex features of the parse history, as demonstrated in the experiments in section 5.4.

While for CoNLL-2006 experiments we tried different features set to test our hypothesis of low importance of the feature model, as described in the experimental section of this chapter, for most of the languages in CoNLL-2007 experiments we used the set proposed by (Nivre et al., 2006). However, we had to remove from it all the lookahead features to obtain a valid generative history-based model.

We could describe the structure of ISBN for dependency parsing by defining the set of previous related time steps (i.e. $R'(t)$), decisions which are related to the considered decision as in expression (3.6). However, we chose to use the terminology employed by (Nivre et al., 2006), and describe explicitly the set of relevant previous words, their features and relations.[1] The words in the datasets had the following features: *FORM* - the word form, *LEMMA* - lemmatized word, *CPOS* - a coarse PoS tag, *POS* - more fine-grain PoS tag, *FEAT* - set of various word features. All these features were available both at training and at testing.[2] Though not all the datasets had the same complete set of the features. The detailed description of datasets is available in (Buchholz & Marsi, 2006) for the CoNLL-2006 shared task and in (Nivre et al., 2007b) for the CoNLL-2007 shared task. Additionally, the dependency relation type of a word to its parent in the partial dependency structure built by the parser before the current step (*DEPREL*) for convenience is also regarded as a feature associated with the word. This resulted in the following feature model:

1. *Top of S*: *DEPREL*, *FORM*, *LEMMA*, *CPOS POS* and *FEAT* of the word at the top of the stack.

2. *Sub-Top of S*: *POS* of the word immediately under the top of the stack.

[1]Defining this model in terms of related composite decisions $R'(t)$ is more difficult, because for simplicity in expression (3.6) we assumed that all the sub-decisions $d_1^{R'_r(t)}, \ldots, d_{K_{R'_r(t)}}^{R'_r(t)}$ are relevant. In the feature model described in (Nivre et al., 2006) only specific atomic decisions from the set $d_1^{R'_r(t)}, \ldots, d_{K_{R'_r(t)}}^{R'_r(t)}$ are used. E.g. only the PoS tag of the word under the top of the stack is used, while it is only a single elementary decisions for a composite decision $D^{R'_r(t)}$, other elementary decisions within $D^{R'_r(t)}$ (prediction of $FEAT$ and $FORM$) are not used in the feature model of (Nivre et al., 2006). Also some word features, e.g. *LEMMA* are not predicted in $\mathbf{Shift}_w$ decision but they are still used as conditioning features in the parser.

[2]According to the rules of the shared tasks gold standard PoS tags are used both at training and at testing. We follow this set-up.

3. *Next*: *FORM*, *LEMMA*, *CPOS POS* and *FEAT* of the front element of the queue.[3]

4. *Head of Top of S*: *DEPREL* and *FORM* of the head word of the current stack top.

5. *Left Child of Top of S*: *DEPREL* of the leftmost left child of the current stack top.

6. *Right Child of Top of S*: *DEPREL* of the rightmost right child of the current stack top.

7. *Left Child of Front of I*: *DEPREL* of the leftmost child of the front element of the queue I.

The most important design decision in building an ISBN model is choosing the finite set of relevant previous state vectors for the current decision. By connecting to a previous state, we place that state in the local context of the current decision. This specification of the domain of locality determines the inductive bias of learning with ISBNs. When deciding what information to store in its latent variables, an ISBN is more likely to choose information which is immediately local to the current decision. This stored information then becomes local to any following connected decision, where it again has some chance of being chosen as relevant to that decision. In this way, the information available to a given decision can come from arbitrarily far away in the chain of interconnected states, but it is much more likely to come from a state which is relatively local. Thus, we need to choose the set of local (i.e. connected) states in accordance with our prior knowledge about which previous decisions are likely to be particularly relevant to the current decision.

Similarly to constituent parsing, to choose which previous decisions are particularly relevant to the current decision, we make use of the partial dependency structure which has been decided so far in the parse. Specifically, the current latent state vector is connected to the set of 7 previous latent state vectors (if they exist) according to the following relationships:

1. *Input Context*: the last previous state with the same queue I.

2. *Stack Context*: the last previous state with the same stack S.

3. *Right Child of Top of S*: the last previous state where the rightmost right child of the current stack top was on top of the stack.

4. *Left Child of Top of S*: the last previous state where the leftmost left child of the current stack top was on top of the stack.

5. *Left Child of Front of I* : the last previous state where the leftmost child of the front element of I was on top of the stack.

[3]We refer to the *head* of the queue as the *front*, to avoid unnecessary ambiguity of the word *head* in the context of dependency parsing.

6. *Head of Top of S*: the last previous state where the head word of the current stack top was on top of the stack.

7. *Top of S at Front of I*: the last previous state where the current stack top was at the front of the queue.

Each of these 7 relations has its own distinct weight matrix for the resulting arcs in the ISBN, but the same weight matrix is used at each position where the relation is relevant.

All these relations but the last one are motivated by linguistic considerations. The current decision is primarily about what to do with the current word on the top of the stack and the current word on the front of the queue. The *Input Context* and *Stack Context* relationships connect to the most recent states used for making decisions about each of these words. The *Right Child of Top of S* relationship connects to a state used for making decisions about the most recently attached dependent of the stack top. Similarly, the *Left Child of Front of I* relationship connects to a state for the most recently attached dependent of the queue front. The *Left Child of Top of S* is the first dependent of the stack top, which is a particularly informative dependent for many languages. Likewise, the *Head of Top of S* can tell us a lot about the stack top, if it has been chosen already.

As we discussed in the previous section, a second motivation for including a state in the local context of a decision is that it might contain information which has no other route for reaching the current decision. In particular, it is generally a good idea to ensure that the immediately preceding state is always included somewhere in the set of connected states. This requirement ensures that information, at least theoretically, can pass between any two states in the decision sequence, thereby avoiding any hard independence assumptions. The last relation, *Top of S at Front of I*, is included mainly to fulfill this requirement. Otherwise, after a $\mathbf{Shift}_{w_j}$ operation, the preceding state would not be selected by any of the relationships.

The probability of each elementary decision $d_k^{t'}$ depends both on the current state vector $S^{t'}$ and on the previously chosen elementary action $d_{k-1}^{t'}$ from $D^{t'}$ as described in expression 3.7. It is easy to see why the original decision $\mathbf{Right\text{-}Arc}_r$ (Nivre et al., 2004) had to be decomposed into two distinct decisions: the decision to construct a labeled arc and the decision to shift the word. Use of this composite $\mathbf{Right\text{-}Arc}_r$ would have required the introduction of individual parameters for each pair (w, r), where w is an arbitrary word in the lexicon and r - an arbitrary dependency relation.

5.3 Searching for the Best Tree

Again, as with the constituent parsing, complete search is not feasible. However, the success of the deterministic parsing strategy which uses the same parsing order (Nivre et al., 2006), suggests that it should be relatively easy to find an accurate approximation to the best parse with heuristic search methods. Unlike Nivre et al., we cannot use a lookahead in our generative model, as was discussed in section 5.1, so a greedy method is unlikely to lead to a good approximation. Instead we use a pruning strategy similar to the proposed in (Henderson, 2003) and described in the previous chapter, where it was applied to a considerably harder search problem: constituent parsing with a left-corner parsing order.

We apply fixed beam pruning after each decision $\mathbf{Shift}_{w_j}$, because knowledge of the next word in the queue I helps distinguish unlikely decision sequences. We could have used best-first search between $\mathbf{Shift}_{w_j}$ operations, but this still leads to relatively expensive computations, especially when the set of dependency relations is large. However, most of the word pairs can possibly participate only in a very limited number of distinct relations. Thus, we pursue only a fixed number of most probable relations r after each $\mathbf{Left\text{-}Arc}_r$ and $\mathbf{Right\text{-}Arc}_r$ operation.

Experiments with a variety of post-shift beam widths confirmed that very small validation performance gains are achieved with widths larger than 30, and sometimes even a beam of 5 was sufficient. We found also that allowing 5 different dependency labels after each dependency prediction operation was enough that it had virtually no effect on the validation accuracy. For some experiments for the CoNLL-2007 shared task we used a larger beam, but as we will discuss in the empirical section, again, gains were not large.

5.4 Empirical Evaluation

In this section we evaluate the ISBN model for dependency parsing on three treebanks from the CoNLL-2006 shared task. We compare our generative models with the best parsers from the CoNLL-2006 task, including the SVM-based parser of (Nivre et al., 2006) (the MALT parser), which uses the same parsing algorithm. To test the feature induction abilities of our model we compare results with two feature sets, the feature set tuned individually for each language by (Nivre et al., 2006), and another feature set which includes only obvious local features. This simple feature set comprises only features of the word on top of the stack S and the front word of the queue I. We compare the gain from using tuned features with the similar gain obtained by the MALT parser. To obtain these results we train the MALT parser with the same two feature sets.[4]

In order to distinguish the contribution of ISBN's feature induction abilities from the contribution of our estimation method and search, we perform another experiment. We use the tuned feature set and disable the feature induction abilities of the model by removing all the arcs between latent variables vectors. Comparison of this restricted model with the full ISBN model shows how im-

[4]The tuned feature sets were obtained from http://w3.msi.vxu.se/~nivre/research/MaltParser.html. We removed lookahead features for ISBN experiments but preserved them for experiments with the MALT parser. Analogously, we extended simple features with 3 words lookahead for the MALT parser experiments.

portant the feature induction is. Also, comparison of this restricted model with the MALT parser, which uses the same set of features, indicates whether our generative estimation method and use of beam search is beneficial.

Though the most of this section focus on experiments on the selected languages from CoNLL-2006 shared tasks, towards the end of the empirical section we will present our results in the CoNLL-2007 shared tasks and analyze the efficiency of the model.

5.4.1 Experimental Setup

We used the CoNLL-2006 distributions of Danish DDT treebank (Kromann, 2003), Dutch Alpino treebank (van der Beek et al., 2002) and Slovene SDT treebank (Dzeroski et al., 2006). The choice of these treebanks was motivated by the fact that they all are freely distributed and have very different sizes of their training sets: 195,069 tokens for Dutch, 94,386 tokens for Danish and only 28,750 tokens for Slovene. As it is generally believed that discriminative models do better than generative models with a large amount of training data, so we expected to see similar trend in our results. Test sets are about equal and contain about 5,000 scoring tokens each.

We followed the experimental setup of the shared task and used all the information provided for the languages: gold standard part-of-speech tags and coarse part-of-speech tags, word form, word lemma (lemma information was not available for Danish) and a set of fine-grain word features. As we explained in section 5.1, we treated these sets of fine-grain features as an atomic value when predicting a word. However, when conditioning on words, we treated each component of this composite feature individually, as it proved to be useful on the development set. We used frequency cutoffs: we ignored any property (e.g., word form, feature or even part-of-speech tag[5]) which occurs in the training set less than 5 times. Following (Nivre et al., 2006), we used pseudo-projective transformation they proposed to cast non-projective parsing tasks as projective.

ISBN models were trained using a small development set taken out from the training set, which was used for tuning learning parameters and for early stopping. The sizes of the development sets were: 4,988 tokens for larger Dutch corpus, 2,504 tokens for Danish and 2,033 tokens for Slovene. The MALT parser was trained always using the entire training set. We expect that the incremental mean field approximation should demonstrate better results than neural network (feed-forward) approximation on this task as it is theoretically expected and confirmed on the constituent parsing task in the previous chapter. However, the sizes of testing sets would not allow us to perform any conclusive analysis, so we decided not to perform these comparisons here. Instead we used the mean field approximation for the smaller two corpora and used the neural network approximation for the larger one. Training the mean field approximations on the larger Dutch treebank is feasible, but would significantly reduce the possibilities

[5]Part-of-speech tags for multi-word units in the Danish treebank were formed as concatenation of tags of the words, which led to quite sparse set of part-of-speech tags.

		Danish	Dutch	Slovene
ISBN	TF	85.0	79.6	72.9
	LF	84.5	79.5	72.4
	TF-NA	83.5	76.4	71.7
MALT	TF	85.1	78.2	70.5
	LF	79.8	74.5	66.8
MST		84.8	79.2	73.4
Aver		78.3	70.7	65.2

Table 5.1: Dependency parsing: labeled attachment score on the testing sets of Danish, Dutch and Slovene treebanks.

for tuning the learning parameters on the development set and, thus, would increase the randomness of model comparisons.

All model selection was performed on the development set and a single model of each type was applied to the testing set. We used a state variable vector consisting of 80 binary variables, as it proved sufficient on the preliminary experiments. For the MALT parser we replicated the parameters from (Nivre et al., 2006) as described in detail on their web site.

The labeled attachment scores for the ISBN with tuned features (TF) and local features (LF) and ISBN with tuned features and no arcs connecting latent variable vectors (TF-NA) are presented in table 5.1, along with results for the MALT parser both with tuned and local feature, the MST parser (McDonald et al., 2006), and the average score (Aver) across all systems in the CoNLL-2006 shared task. The MST parser is included because it demonstrated the best overall result in the task, non significantly outperforming the MALT parser, which, in turn, achieved the second best overall result. The labeled attachment score is computed using the same method as in the CoNLL-2006 shared task, i.e. ignoring punctuation. Note, that though we tried to completely replicate training of the MALT parser with the tuned features, we obtained slightly different results. The original published results for the MALT parser with tuned features were 84.8% for Danish, 78.6% for Dutch and 70.3% for Slovene. The improvement of the ISBN models (TF and LF) over the MALT parser is statistically significant for Dutch and Slovene. Differences between their results on Danish are not statistically significant.

5.4.2 Discussion of Results

The ISBN with tuned features (TF) achieved significantly better accuracy than the MALT parser on 2 languages (Dutch and Slovene), and demonstrated essentially the same accuracy on Danish. The results of the ISBN are among the two top published results on all three languages, including the best published results on Dutch. All three models, MST, MALT and ISBN, demonstrate much better results than the average result in the CoNLL-2006 shared task. These results show that our generative model is quite competitive with respect to the

		to root	1	2	3 - 6	> 6
Danish	ISBN	95.1	95.7	90.1	84.1	74.7
	MALT	95.4	96.0	90.8	84.0	71.6
Dutch	ISBN	79.8	92.4	86.2	81.4	71.1
	MALT	73.1	91.9	85.0	76.2	64.3
Slovene	ISBN	76.1	92.5	85.6	79.6	54.3
	MALT	59.9	92.1	85.0	78.4	47.1
Average	ISBN	83.6	93.5	87.3	81.7	66.7
	MALT	76.2	93.3	87.0	79.5	61.0
	Improv	**7.5**	**0.2**	**0.4**	**2.2**	**5.7**

Table 5.2: Dependency parsing: F_1 score of labeled attachment as a function of dependency length on the testing sets of Danish, Dutch and Slovene.

best models, which are both discriminative.[6] We would expect further improvement of ISBN results if we applied discriminative retraining (Henderson, 2004) or reranking with data-defined kernels (Henderson & Titov, 2005), even without introduction of any additional features.[7]

We can see that the ISBN parser achieves about the same results with local features (LF). Local features by themselves are definitely not sufficient for the construction of accurate models, as seen from the results of the MALT parser with local features (and look-ahead). This result demonstrates that ISBNs are a powerful model for feature induction.

The results of the ISBN without arcs connecting latent state vectors is slightly surprising and suggest that without feature induction the ISBN is significantly worse than the best models. This shows that the improvement is coming mostly from the ability of the ISBN to induce complex features and not from either using beam search or from the estimation procedure. It might also suggest that generative models are probably worse for the dependency parsing task than discriminative approaches (at least for larger datasets). This motivates further research into methods which combine powerful feature induction properties with the advantage of discriminative training. Although discriminative reranking of the generative model is likely to help, the derivation of fully discriminative feature induction methods is certainly more challenging.

In order to better understand differences in performance between ISBN and MALT, we analyzed how relation accuracy changes with the length of the head-dependent relation. The harmonic mean between precision and recall of labeled attachment, F_1 measure, for the ISBN and MALT parsers with tuned features is presented in table 5.2. F_1 score is computed for four different ranges of lengths and for attachments directly to root. Along with the results for each of the

[6]Note that the development set accuracy predicted correctly the testing set ranking of ISBN TF, LF and TF-NA models on each of the datasets, so it is fair to compare the best ISBN result among the three with other parsers.

[7]Though in initial experiments we performed during the CoNLL 2007 shared task we did not manage to achieve significant improvement from using reranking.

	Ara	Bas	Cat	Chi	Cze	Eng	Gre	Hun	Ita	Tur	**Ave**
LAS	74.1	75.5	87.4	82.1	77.9	88.4	73.5	77.9	82.3	79.8	**79.90**
UAS	83.2	81.9	93.4	87.9	84.2	89.7	81.2	82.2	86.3	86.2	**85.62**

Table 5.3: Dependency parsing: labeled attachment score (LAS) and unlabeled attachment score (UAS) on the final testing sets for the CoNLL-2007 shared task.

languages, the table includes their mean (Average) and the absolute improvement of the ISBN model over MALT (Improv). It is easy to see that accuracy of both models is generally similar for small distances (1 and 2), but as the distance grows the ISBN parser starts to significantly outperform MALT, achieving 5.7% average improvement on dependencies longer than 6 word tokens. When the MALT parser does not manage to recover a long dependency, the highest scoring action it can choose is to reduce the dependent from the stack without specifying its head, thereby attaching the dependent to the root by default. This explains the relatively low F_1 scores for attachments to root (evident for Dutch and Slovene): though recall of attachment to root is comparable to that of the ISBN parser (82.4% for MALT against 84.2% for ISBN, on average over 3 languages), precision for the MALT parser is much worse (71.5% for MALT against 83.1% for ISBN, on average).

The considerably worse accuracy of the MALT parser on longer dependencies might be explained both by use of a non-greedy search method in the ISBN and the ability of ISBNs to induce history features. To capture a long dependency, the MALT parser should keep a word on the stack during a long sequence of decision. If at any point during the intermediate steps this choice seems not to be locally optimal, then the MALT parser will choose the alternative and lose the possibility of the long dependency.[8] By using a beam search, the ISBN parser can maintain the possibility of the long dependency in its beam even when other alternatives seem locally preferable. Also, long dependencies are often more difficult, and may be systematically different from local dependencies. The designer of a MALT parser needs to discover predictive features for long dependencies by hand, whereas the ISBN model can automatically discover them. Thus we expect that the feature induction abilities of ISBNs have a strong effect on the accuracy of long dependencies. This prediction is confirmed by the differences between the results of the normal ISBN (TF) and the restricted ISBN (TF-NA) model. The TF-NA model, like the MALT parser, is biased toward attachment to root; it attaches to root 12.0% more words on average than the normal ISBN, without any improvement of recall and with a great loss of precision. The F_1 score on long dependencies for the TF-NA model is also negatively affected in the same way as for the MALT parser. This confirms that the ability of the ISBN model to induce features is a major factor in improving

[8]The MALT parser is trained to keep the word as long as possible: if both **Shift** and **Reduce** decisions are possible during training, it always prefers to shift. Though this strategy should generally reduce the described problem, it is evident from the low precision score for attachment to root, that it cannot completely eliminate it.

accuracy of long dependencies.[9]

5.4.3 CoNLL-2007 Shared Task

In the CoNLL-2007 shared task the ISBN parser was evaluated on 10 languages, including Arabic (Hajič et al., 2004), Basque (Aduriz et al., 2003), Catalan (Martí et al., 2007), Chinese (Chen et al., 2003), Czech (Böhmová et al., 2003), English (Marcus et al., 1993; Johansson & Nugues, 2007), Greek (Prokopidis et al., 2005), Hungarian (Csendes et al., 2005), Italian (Montemagni et al., 2003) and Turkish (Oflazer et al., 2003). As with the CoNLL-2006 experiments, the ISBN models were trained using a small development set taken out from the training set, which was used for tuning learning and decoding parameters, for early stopping and very coarse feature engineering. The sizes of the development sets were different: starting from less than 2,000 tokens for smaller treebanks to 5,000 tokens for the largest one. The relatively small sizes of the development sets limited our ability to perform careful feature selection, but this should not have significantly affected the model performance, as discussed in section 5.2, and demonstrated in the previous set of experiments.[10] We used frequency cutoffs: a threshold of 20 for Greek and Chinese and a threshold of 5 for the rest in the same way as for the CoNLL-2006 experiments. Because cardinalities of each of these sets (sets of word forms, lemmas and features) affect the model efficiency, we selected the larger threshold when validation results with the smaller threshold were comparable. For the ISBN latent variables, we used vectors of length 80, based on our previous experience.

Unlike experiments on CoNLL-2006 results, in the shared task we used only the simplest neural network approximation. We would expect better performance with the more accurate incremental mean field approximation. We did not try this because, on larger treebanks it would have taken too long to tune the model with this better approximation, and using different approximation methods for different languages would not be compatible with the shared task rules.

We used the base feature model described in section 5.2 for all the languages but Arabic, Chinese, Czech, and Turkish. For Arabic, Chinese, and Czech, we used the same feature models used in the CoNLL-2006 shared task by (Nivre et al., 2006), and for Turkish we used again the base feature model but extended it with a single feature: the part-of-speech tag of the token preceding the current top of the stack.

To overcome a minor shortcoming of the parsing algorithm of (Nivre et al., 2004) we introduce a simple language independent post-processing step. Nivre's parsing algorithm allows unattached nodes to stay on the stack at the end of

[9]Results presented in (McDonald & Nivre, 2007) suggest that the greedy search also contributes to the lower accuracy of the MALT parser on long dependencies. They show it by comparison with the MST parser which performs global inference. MST achieves significantly higher accuracy on longer dependencies.

[10]Use of cross-validation with our model is relatively time-consuming and, thus, not quite feasible for the shared task.

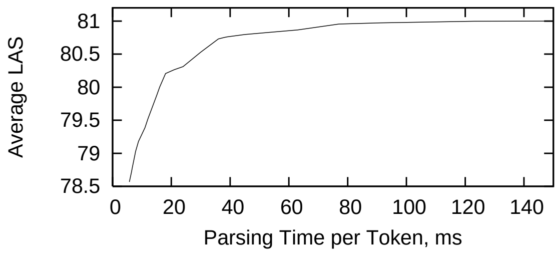


Figure 5.2: Dependency parsing: average labeled attachment score on Basque, Chinese, English, and Turkish development sets as a function of parsing time per token

parsing, which is reasonable for treebanks with unlabeled attachment to root. However, this sometimes happens with languages where only labeled attachment to root is allowed. In these cases (only 35 tokens in Greek, 17 in Czech, 1 in Arabic, on the final testing set) we attached them using a simple rule: if there are no tokens in the sentence attached to root, then the considered token is attached to root with the most frequent root-attachment relation used for its part-of-speech tag. If there are other root-attached tokens in the sentence, it is attached to the next root-attached token with the most frequent relation. Preference is given to the most frequent attachment direction for its part-of-speech tag. This rule guarantees that no loops are introduced by the post-processing.

Results on the final testing set are presented in table 5.3. Detailed results on the shared task is available in (Nivre et al., 2007b), additional analysis and comparison of participating systems are presented in (Nivre, 2007).

The model achieves relatively high scores on each individual language, significantly better than each average result in the shared task. This leads to the third best overall average results in the shared task, both in average labeled attachment score and in average unlabeled attachment score. The absolute error increase in labeled attachment score over the best system is only 0.4%. We attribute ISBN's success mainly to its ability to automatically induce features, as this significantly reduces the risk of omitting any important highly predictive features. This makes an ISBN parser a particularly good baseline when considering a new treebank or language, because it does not require much effort in feature engineering.

It is also important to note that the model is quite efficient. Figure 5.2 shows the tradeoff between accuracy and parsing time as the width of the search beam is varied, on the development set. This curve plots the average labeled attachment score over Basque, Chinese, English, and Turkish as a function of parsing time per token.[11] Accuracy of only 1% below the maximum can be

[11]A piecewise-linear approximation for each individual language was used to compute the average. Experiments were run on a standard 2.4 GHz desktop PC.

achieved with average processing time of 17 ms per token, or 60 tokens per second.[12]

5.5 Summary of the Chapter

We proposed a latent variable dependency parsing model based on Incremental Sigmoid Belief Networks. Unlike state-of-the-art dependency parsers, it uses a generative history-based model. We demonstrated that it achieves state-of-the-art results on a selection of languages from the CoNLL-2006 shared task. The ISBN uses a vector of latent variables to represent an intermediate state and uses relations defined on the output structure to construct the arcs between latent state vectors. These properties make it a powerful feature induction method for dependency parsing, and it achieves competitive results even with very simple explicit features. The ISBN model is especially accurate at modeling long dependencies, achieving average improvement of 5.7% over the state-of-the-art baseline on dependences longer than 6 words. Empirical evaluation demonstrates that competitive results are achieved mostly because of the ability of the model to induce complex features and not because of the use of a generative probability model or a specific search method. Competitive results on the CoNLL-2007 shared task, where it achieved the best results among the single model parsers and the third in overall, confirm the viability of the proposed approach. As with other generative models, it can be further improved by the application of discriminative reranking techniques. Discriminative methods are likely to allow it to significantly improve over the current state-of-the-art in dependency parsing.[13]

[12]For Basque, Chinese, and Turkish this time is below 7 ms, but for English it is 38 ms. English, along with Catalan, required the largest beam across all 10 languages. Note that accuracy in the lowest part of the curve can probably be improved by varying latent vector size and frequency cut-offs. Also, efficiency was not the main goal during the implementation of the parser, and it is likely that a much faster implementation is possible.

[13]The ISBN dependency parser is downloadable from `http://cui.unige.ch/~titov/idp/`

Chapter 6

Data-Defined Kernels for Parse Reranking

Discriminative reranking methods have been shown to be effective in improving the accuracy of generative probabilistic models for natural language parsing (Collins & Koo, 2005; Charniak & Johnson, 2005; Henderson, 2004). As we discussed in the introduction, in discriminative reranking a classifier, a reranker, is trained to choose the best parse tree from the list of candidates provided by a generative probabilistic model. Reranking has advantages: learning can try to optimize measures related directly to expected testing performance, rather than the log-likelihood of the training data. Fully discriminative training for constituent parsing is normally prohibitively expensive (Taskar et al., 2004; Turian & Melamed, 2006), thus making reranking the only feasible approach for using discriminative criteria in constituent parsing. Work on discriminative reranking approaches has been focused both on proposing modifications of machine learning algorithms to make them suitable for parse reranking (or structured prediction in general) (Collins & Koo, 2005; Tsochantaridis et al., 2004; Shen & Joshi, 2005), or on methods to construct an appropriate feature space. The latter goal has often been addressed by proposing an explicit set of linguistically motivated features (e.g. (Collins & Koo, 2005)), or by defining a kernel (Collins & Duffy, 2002; Shen et al., 2003; Suzuki et al., 2004).

Some work in machine learning has taken an alternative approach to explicitly defining feature spaces (or kernels), where the mapping to the feature space, i.e. the feature extractor, is derived from a probabilistic model of the problem (Jaakkola & Haussler, 1998; Tsuda et al., 2002a; Seeger, 2002; Tsuda & Kawanabe, 2002; Tsuda et al., 2002b). Applying this way of defining feature spaces to parsing has advantages. First, linguistic knowledge about parsing is reflected in the design of the probabilistic model. The probabilistic model is already available in the discriminative reranking problem. Therefore, the process is fairly automatic and does not require any background knowledge. Second, the feature extractor is defined using the trained parameters of the probabilistic model. Thus the feature space is in part determined by the training data and is automatically tailored to reflect properties of parse trees which are relevant to parsing. For this reason, we call both the induced features spaces and the corresponding kernels (i.e. the inner products in these feature spaces) data-defined.[1] In this chapter we will use interchangeably the terms “feature extractor” and “kernels”: most of the discussed kernels will have finite kernel functions computed in an explicit way from a probabilistic model.

In this chapter we describe how this approach can be applied to parse reranking. We consider two data-defined feature spaces previously proposed for classification problems and propose their modifications for parse reranking. Though formally these feature extractors can be induced from any probabilistic model, as we will discuss further, they are especially appropriate for non-linear models. In experiments we continue to use ISBNs, or more precisely, the neural network model SSN, which can be viewed as an ISBN approximation (see chapter 3). We rerank the output of this probabilistic model with the Voted Perceptron (Freund & Schapire, 1998) operating in the feature space induced from the probabilistic model. This method achieves significant improvement over the accuracy of the probabilistic model alone. We should note that the proposed approach for construction of the features space is essentially orthogonal to the use of hand-crafted features: the feature space induced from the probabilistic model can be augmented by additional features motivated by linguistic considerations.

The chapter is structured as follows. In section 6.1 we describe the previous work on data-defined kernels for classification problems. Section 6.2 discusses how they can be adapted to the parse reranking problem. In section 6.3 we describe the learning algorithm and provide details on reparametrization of the ISBN model, which simplifies the computation of the feature extractor and thus allows for efficient sparse representation of parse trees in the induced feature space. Section 6.4 describes the experimental setup and results obtained on the parse reranking task. We conclude with discussions and a brief review of approaches for construction of features spaces previously considered in discriminative parse reranking.

6.1 Data-defined kernels

In recent years, several methods have been proposed for constructing kernels from trained probabilistic models. As a rule, these kernels are then used with linear classifiers to learn the desired task. In this section we will discuss the two most frequently considered kernels: the Fisher kernel (Jaakkola & Haussler, 1998) and the TOP kernel (Jaakkola & Haussler, 1998), as well as a number of other, somewhat lesser known kernels. All these kernels have previously been applied to classification problems. As standard for classification problems, the goal here is to construct a feature representation of the input element x $\varphi(x)$ from a probabilistic model, or directly to the construct the inner product

[1] In relevant studies, they are also known as model-based (Jaakkola & Haussler, 1998) or model-driven (Gärtner, 2003).

$K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$. This is different from the parsing set-up, where, as we discussed in the introduction (see expression 1.2), we are interested in finding a joint input-output representation $\varphi(x, y)$, where y is some parse tree for sentence x, and not in a feature representation of the sentence $\varphi(x)$.

6.1.1 Fisher Kernel

The Fisher kernel (Jaakkola & Haussler, 1998) is the best known kernel belonging to the class of data-defined kernels. Let us consider a binary classification task, where the goal is to predict a binary class label y ($y \in \{+1; -1\}$) given an input x. Jaakkola and Haussler (Jaakkola & Haussler, 1998) consider derivation of a feature extractor from a generative probabilistic model $P(x|\hat{\theta})$, where $\hat{\theta}$ is a vector of model parameters. Note that this is a generative model of input x, not of input-output pairs (x, y) as is the case for generative models of trees in parsing. This generative model may still include the classification variable y as a latent variable; therefore, it is often possible to use a plug-in estimate $P(y = +1|x, \hat{\theta})$ to perform the classification. However, the goal of the kernel construction is to capture differences in the generative process between a pair of examples x_i and x_j, rather than just differences in the posterior probabilities of the labels computed separately for each of the inputs x_i and x_j. Jaakkola and Haussler suggest that the derivative of the log-likelihood with respect to a model parameter describes the contribution of that parameter to the generative process. Therefore the Fisher score, i.e. the gradient of the log-likelihood $U_x = \nabla_\theta \log P(x|\hat{\theta})$, could be used as a feature extractor. The parameter space of a probabilistic model is not Euclidean; instead, it has a Riemannian metric structure with a local metric given by the Information matrix $I = E_x[U_x U_x^T]$ (Amari, 1985), where E_x denotes the expectation with respect to the probabilistic model $P(x|\hat{\theta})$. The natural gradient is the gradient in the direction that maximizes $\log P(x|\theta)$ while traversing the minimum distance in the Riemannian manifold of all the probability models of the form $P(x|\theta)$, i.e. the direction of steepest descent in the Riemannian manifold.[2] Therefore, the use of the natural gradient to represent the properties of the generative process is more appropriate than the use of the standard gradient.

The natural gradient is given by $I^{-1}U_x$, and the inner product of the natural gradients in the Riemannian manifold defines the Fisher kernel:

$$K_{FK}(x_i, x_j) = (I^{-1}U_{x_i})^T I (I^{-1}U_{x_j}) = U_{x_i}^T I^{-1} U_{x_j}. \tag{6.1}$$

It has been demonstrated in (Jaakkola & Haussler, 1998) that if the probability model contains the target class as a latent variable, i.e. if the generative probability can be represented as

$$P(x|\theta) = \sum_y P(x|y, \theta) P(y|\theta), \tag{6.2}$$

[2]It has been argued that use of the natural gradient instead of the standard gradient in gradient descent is both faster and yields better generalization (Amari, 1998; Le Roux et al., 2008).

then there exists a linear model in the feature space of the kernel (6.1) such that it performs at least as well as the classifier based on the plug-in estimate. Importantly, the Fisher kernel (6.1) is invariant under one-to-one reparametrizations, which is arguably a desirable property in learning. However, in practice computation of the inverse Information matrix is usually expensive or infeasible and instead the inner product of the Fisher scores $U_{x_i}^T U_{x_j}$ is used as the kernel. In further discussions we will refer to the Fisher scores as feature extractors of the Fisher kernel. Classifier with the Fisher kernel achieve very competitive results in a number of problems involving the classification of structures (Jaakkola & Haussler, 1998; Jaakkola et al., 2000; Pavlidis et al., 2001), where otherwise it is difficult to define an appropriate kernel function. However, we are not aware of any previous application of the Fisher kernel to parsing or structured prediction in general.

6.1.2 TOP Kernel

As described above, the Fisher kernel is derived from the generative models of the input. However, one of the arguments for its construction considered the model $P(x|\theta)$ which uses the label y as a latent variable in the model, expression (6.2). This type of model allows us to obtain an estimator of the posterior probability $P(y|x,\theta)$. (Tsuda et al., 2002a) proposed to use this model to obtain the feature extractor. The feature extractor of their kernel for the input x is defined by

$$\varphi_{\hat{\theta}}(x) = (v(x,\hat{\theta}), \frac{\partial v(x,\hat{\theta})}{\partial \theta_1}, \ldots, \frac{\partial v(x,\hat{\theta})}{\partial \theta_l}), \tag{6.3}$$

where v is the log-odds of the event y:

$$v(x,\hat{\theta}) = \log P(y=+1|x,\hat{\theta}) - \log(1 - P(y=+1|x,\hat{\theta})).$$

The kernel was called the Tangent vector Of the Posterior log-odds (TOP). The first component of the feature extractor is a monotonous function of $P(y = +1|x,\hat{\theta})$; therefore, it is clear that a classifier with this kernel can obtain accuracy at least as good as the accuracy of classification with the plug-in estimate $P(y||x,\hat{\theta})$.

The choice of the TOP kernel feature extractor is motivated by the minimization of the classification error of an optimal linear classifier $w^T\varphi_{\hat{\theta}}(x)+b$. Tsuda et al. have demonstrated that this error is closely related to the estimation error of the posterior probability $P(y=+1|x)$ by the estimator $g(w^T\varphi_{\hat{\theta}}(x)+\ b)$, where g is the logistic sigmoid function $g(t) = 1/(1+\exp(-t))$. We will not provide details of their derivation, but our derivation of the Fisher kernel for reranking will use a similar technique. The TOP kernel has demonstrated better results than the Fisher kernel on a number of classification tasks, see e.g. (Tsuda et al., 2002a).

6.1.3 Other Data-Defined Kernels

There have been several other proposals of feature extractors based on probabilistic models. The *Leave One Out kernel* (LOO) (Tsuda & Kawanabe, 2002) measures similarity between examples x_i and x_j using the probability models $P(x|\theta^{x_i})$ and $P(x_j|\theta^{x_j})$ obtained by removing examples x_i and x_j from the training set of the model. This estimator converges to the Fisher kernel when the number of samples goes to infinity. Clearly, it is not feasible to use LOO with models where estimates of the model parameters θ cannot be easily (analytically) computed. Therefore, application of this kernel to any non-linear (latent variable) model is not practical.

A *Mutual Information* kernel has been proposed in (Seeger, 2002). They consider a Bayesian approach: to compute the similarity between examples x_i and x_j they compare the effects that these examples have on the posterior distribution of model parameter θ. They show that the Fisher kernel can be regarded as a MAP approximation to their Mutual Information kernel. Unfortunately, exact computation of this kernel is not feasible and the proposed variational approximation does not lead to encouraging results.

A general framework of *marginalized kernels* was proposed in (Tsuda et al., 2002b), where the kernel between instances x_i and x_j instances is defined by:

$$K_{MK}(x_i, x_j) = \sum_{h,h'} P(h|x_i, \hat{\theta})P(h'|x_j, \hat{\theta})K((h, x_i), (h', x_j)),$$

where h is some latent variable, e.g. y, and the K is some kernel over pairs (x, h). The Fisher kernel can be represented as a marginalized kernel with a specific kernel K, in this case kernel K will be also derived from the probabilistic model. This framework has been proven to be useful in a number of problems involving structured input spaces, e.g. a marginalized kernel for labeled graphs was proposed in (Kashima et al., 2003), and applied to problems in chemistry, an application of the marginalized kernel to problems in bionformatics was considered in (Tsuda et al., 2002b).

6.2 Data-defined Kernels for Reranking

In this section we return to the parsing problem and discuss how data-defined kernels can be applied to the reranking problem.

6.2.1 Applicability of the Fisher Kernel

The Fisher kernel (Jaakkola & Haussler, 1998) in the exact sense cannot be applied to parse reranking. The idea of the Fisher kernel was that the kernel induced from a generative model of the input $P(x)$ is used in the prediction of some property y. Even if we assume that we can obtain the distribution $P(x)$ by marginalizing out parse trees y, we will obtain only a kernel over sentences $K(x, x')$, which is not what we want in parse reranking.[3] However, nothing prevents us from considering the "Fisher kernel" over input-output pairs $K((x, y), (x', y')) = \nabla_\theta \log P(x, y|\hat{\theta})^T I^{-1} \nabla_\theta \log P(x', y'|\hat{\theta})$. However, in the theoretical sense this kernel is more closely related to the TOP kernel of Tsuda et al. (Tsuda et al., 2002a) than to the original Fisher kernel of Jaakkola and Haussler.[4] We still would expect that it reflects the properties of the generative process of an input-output pair and the information theoretic motivation of the Fisher kernel, described in the previous section, is still valid. To distinguish this kernel from the other kernel considered in this chapter we will refer to it as the Fisher Kernel. Note, that the guarantee that there exists a linear model in the feature space of the kernel (6.1), that it performs at least as well as the MAP classifier, is not longer valid for this kernel anymore. This property is desirable and to enforce it we extend the feature extractor of the kernel with the log-probability of the input-output pair $\log P(x, y|\hat{\theta})$.[5] Accurate approximation of the Information matrix is not feasible for the model considered in our experiments and as has been previously done in practice, we replace it with the identity matrix. The resulting feature extractor is given by

$$\varphi_{\hat{\theta}}^{(FK)}(x, y) = (\log P(x, y|\hat{\theta}), \frac{\partial \log P(x, y|\hat{\theta})}{\partial \theta_1}, \ldots, \frac{\partial \log P(x, y|\hat{\theta})}{\partial \theta_l}). \tag{6.4}$$

Now we provide another motivation for this Fisher kernel, using the method similar to the one used in the derivation of the TOP kernel in (Tsuda et al., 2002a). The difference of our derivation from the one proposed in (Tsuda et al., 2002a) is caused by the difference in the error function appropriate for reranking and binary classification, and by the difference in the form of appropriate probability estimators. We will first show that the error of a linear reranker can be upper bounded by the estimation error of a probability estimator which uses the same feature extractor as the linear reranker. Therefore, we can expect that the feature extractor which minimizes the estimation error of the estimator leads to good reranking performance. Second, we will show that the feature extractor of the Fisher kernel (6.4) is appropriate for probability estimation.

In the task of reranking, a reranker tries to predict the best parse tree from a list of candidates $G(x)$ for a sentence x. Assuming that there exist an unknown true distribution $P^\star(x, y)$ we can write the reranking loss of a linear classifier

[3]However, such a kernel might be useful for sentence classification.

[4]Even if we frame the reranking problem as a binary classification of sentence-tree pairs on correct vs. incorrect, the motivation of the original kernel is still not valid. In this case we need a probabilistic model $P(x, y|\theta)$ which models both correct and incorrect sentence-tree pairs. Essentially it means that we need to train $P(x, y|\theta)$ on both correct and wrong sentences in the candidate list with a latent variable z representing correctness of the pair. This set-up seems very artificial, though it resembles such methods as contrastive estimation (Smith & Eisner, 2005).

[5]Note that almost all the parse reranking methods use the log-probability assigned to the parse tree as an additional feature (Collins & Duffy, 2002; Collins, 2000; Collins & Koo, 2005; Shen & Joshi, 2003; Shen et al., 2003).

as[6]

$$Error(\varphi, w, x) = E_{y \in G(x)}[\Phi(\min_{y' \in G(x)} w^T(\varphi(x, y) - \varphi(x, y')))], \tag{6.5}$$

where $\Phi(t)$ is the step function equal to 1 if $t > 0$, 0 otherwise. The expectation is computed with respect to the distribution $P^\star(y|G(x))$, i.e. the probability of generating y from the true underlying model given that one of the candidates $G(x)$ was generated, $\frac{P^\star(x,y)}{\sum_{y' \in G(x)} P^\star(y',x)}$. Note that if the linear model (w, φ) assigns the highest score to the tree $\hat{y}$ then the functional $Error(\varphi, w, x)$ is equal to the probability of generating other trees in the candidate list, $1 - P^\star(\hat{y}|G(x))$.

We are interested in the feature extractor φ which achieves the minimal expected error $E_x[Error(\varphi, w, x)]$ for the parameter vector w chosen by a learning algorithm. We do not solve this problem directly, but instead consider the risk of a classifier with an optimally chosen w:

$$R(\varphi) = \min_w E_x[Error(\varphi, w, x)] \tag{6.6}$$

in hope that the induced φ will be appropriate for an arbitrary algorithm. The lower bound on this risk is given by the Bayes error $L^\star$:

$$L^\star = E_x[1 - \max_{y \in G(x)} P^\star(y|G(x))].$$

We will upper bound the risk $R(\varphi)$ using the estimation error of an estimator of the conditional probability in the candidate list:

$$h(w, y, G(x), x, \varphi) \approx P^\star(y|G(x)).$$

The only requirement we impose on the functional h is that it should be a monotonically non-decreasing function of the inner product $w^T\varphi(x, y)$. We denote by $D(h, \varphi)$ the estimation error of this estimator h with the feature extractor φ and the optimally chosen parameter vector w:

$$D(h, \varphi) = min_w E_x[\max_{y \in G(x)} |h(w, y, G(x), x, \varphi) - P^\star(y|G(x))|].$$

Note that the estimator error $D(h, \varphi)$ depends on the largest error in the candidate list and not on the expectation under $P^\star(y|G(x))$. Now we can formulate the upper bound on the reranking error of the linear classifier $R(\varphi, w)$.

Lemma 6.2.1. $R(\varphi) - L^\star \leq 2D(h, \varphi)$

Proof. Let us fix a sentence x and the parameter vector w. As we noted above, the loss incurred by selecting a parse tree y is equal to $1 - P^\star(y|G(x))$; therefore,

[6]In this chapter we focus the on the zero-one loss; for the optimization of partial match measures we refer the reader to chapter 8 where we propose methods to minimize the Bayes risk in parsing.

by choosing the parse tree $\hat{y}$ instead of the most probable parse tree $y^\star$ the reranker increases the loss by

$$\begin{aligned} P^\star(y^\star|G(x)) - P^\star(\hat{y}|G(x)) = &(P^\star(y^\star|G(x)) - h(w, y^\star, G(x), x, \varphi)) \\ &- (P^\star(\hat{y}|G(x)) - h(w, y^\star, G(x), x, \varphi))\,. \end{aligned}$$

Because of the monotonicity requirement on $h(w, y^\star, G(x), x, \varphi)$, $h(w, y^\star, G(x), x, \varphi) \leq h(w, \hat{y}, G(x), x, \varphi)$, we can upper bound the loss increment as

$$\begin{aligned} (P^\star(y^\star|G(x)) - h(w, y^\star, G(x), x, \varphi)) - (P^\star(\hat{y}|G(x)) - h(w, \hat{y}, G(x), x, \varphi)) \\ \leq 2 \max_{y \in G(x)} |P^\star(y|G(x)) - h(w, y, G(x), x, \varphi)|. \end{aligned}$$

This inequality is true for any x and w, which immediately implies the statement of the lemma. $\square$

This shows that if we construct an accurate probability estimator we are guaranteed to achieve low reranking error with an optimal linear reranker which uses the same feature extractor as the estimator.

Now it is natural to estimate the conditional probability $P(y|G(x))$ with the log-linear estimator (Johnson et al., 1999; Collins & Koo, 2005):

$$\log h(w, \hat{y}, G(x), x, \varphi) = w^T \varphi(x, y) - \log Z(x), \tag{6.7}$$

where $Z(x)$ is the partition function. Under the technical assumption that there exist such parameters $\theta^\star$ that $P(y|G(x)) = P(y|G(x), \theta^\star)$, where

$$P(y|G(x), \theta^\star) = \frac{P(x, y|\theta^\star)}{\sum_{y' \in G(x)} P(x, y'|\theta^\star)},$$

we can write the true log-probability $\log P(y|G(x))$ as the Taylor expansion of $P(y|G(x), \theta)$ around $\hat{\theta}$ with remainder term in the Lagrange form

$$\begin{aligned} \log P(y|G(x)) = &\log P(x, y|\hat{\theta}) + \nabla_\theta \log P(x, y|\hat{\theta})^T (\theta^\star - \hat{\theta}) \\ &- \log \sum_{y' \in G(x)} P(x, y|\hat{\theta}) - \nabla_\theta \log (\sum_{y' \in G(x)} P(x, y|\hat{\theta}))^T (\theta^\star - \hat{\theta}) \\ &+ \frac{1}{2} (\theta^\star - \hat{\theta})^T \nabla^2_\theta \log P(y|G(x), \widetilde{\theta})(\theta^\star - \hat{\theta}), \end{aligned}$$

where $\widetilde{\theta}$ is a certain point between $\hat{\theta}$ and $\hat{\theta}$ in the Euclidean space, existence of such point is guaranteed by the Taylor theorem. Comparing this expression with the form of $\log h(w, \hat{y}, G(x), x, \varphi)$ in (6.7), we can observe that the first order approximation to $\log P(y|G(x)$ by $\log h(w, \hat{y}, G(x), x, \varphi)$ can be obtained by setting φ to

$$(\log P(x, y|\hat{\theta}) - C(x), \frac{\partial \log P(x, y|\hat{\theta})}{\partial \theta_1}, \ldots, \frac{\partial \log P(x, y|\hat{\theta})}{\partial \theta_l}),$$

where $C(x) = \log Z(x) - \log \sum_{y' \in G(x)} P(x, y|\hat{\theta}) - \nabla_\theta \log \left(\sum_{y' \in G(x)} P(x, y|\hat{\theta})\right)^T (\theta^\star - \hat{\theta})$ and setting w to $(1, (\theta^\star - \hat{\theta})^T)$. The bias term of the estimator $C(x)$ is independent of y and, therefore, irrelevant to the reranking task, because learning algorithms depend on φ only through differences $\varphi(x, y) - \varphi(x, y')$, where y and y' are both from the same candidate list G(x).[7]. Therefore, the feature extractor of the Fisher kernel $\varphi_{\hat{\theta}}^{(FK)}$ (6.4) can be used instead.

6.2.2 TOP Kernel for Reranking

We could try to apply the TOP kernel directly to the parsing problem, by observing that parsing can be framed as the binary classification problem, where we try to classify the pair (x, y) into two categories: category $z = +1$ of correct pairs vs. category $z = -1$ of incorrect pairs. The generative model $P(y, x|\hat{\theta})$ was trained only on correct parse trees and thus it gives an estimate of $P(y, x|\hat{\theta}, z = +1)$. However, we do not have models for $P(y, x|\hat{\theta}, z = -1)$ and $P(z = +1)$ and therefore we cannot apply the Bayes rule to obtain an estimate of $P(z = +1|y, x)$ needed to build the feature extractor (6.3). One way to tackle this problem, as we noted in a footnote earlier, would be to train a probabilistic model on the wrong parse trees from the candidate list and use $P(z = +1) = 1/N$, where N is the size of the candidate list. However, we chose to tackle this problem in a different way.

Instead we will derive a kernel, similar in the algebraic form to the TOP kernel of Tsuda et al. but with somewhat different motivation. In the previous section we defined the kernel motivated by minimization of the estimation error of a probability estimator. However, discriminative methods, such as support vector machines or the perceptron algorithm, do not directly correspond to a probability model. It is easy to see, for example, that the direction of an update in the perceptron algorithm is very different from the direction of the gradient used in learning maximum likelihood estimators, under the assumption of normalized exponential form of the probability estimator (i.e. softmax). These discriminative models instead optimize measures closely related to the reranking error defined as the number of parses ranked higher than the correct parse $y^\star$ (Collins & Koo, 2005):

$$RerankError(\varphi, w, x) = \sum_{y' \in G(x) \setminus \{y^\star\}} \Phi(w^T(\varphi(x, y') - \varphi(x, y^\star))), \qquad (6.8)$$

where Φ is, as before, the step function equal to 1 if the argument is positive and 0 otherwise. Therefore, instead of probability estimation we could consider the task directly related to the *RerankError*: the task of predicting the true rank of a parse tree y in the candidate list, i.e. the number of parses having the true probability higher than y:

$$r(y) = \sum_{y' \in G(x) \setminus \{y\}} \Phi(P^\star(x, y') - P^\star(x, y)).$$

[7]Also, it is easy to show that $C(x) = o((\theta^\star - \hat{\theta})^2)$

This rank $r(y)$ can be approximated with a smooth function as

$$r(y) \leq \sum_{y' \in G(x) \setminus \{y\}} \frac{P^\star(x, y')}{P^\star(x, y)}.$$

Instead of performing linear expansion of $\log P(x, y|\theta)$ around $\hat{\theta}$, as in the previous section, we propose to perform the expansion of the logarithm of this rank approximation $v(x, y, \hat{\theta}) = \log \sum_{y' \in G(x) \setminus \{y\}} P(x, y'|\hat{\theta}) - \log P(x, y|\hat{\theta})$. The resulting feature extractor will be

$$\varphi_{\hat{\theta}}^{(TK)}(x, y) = (v(x, y, \hat{\theta}), \frac{\partial v(x, y, \hat{\theta})}{\partial \theta_1}, \ldots, \frac{\partial v(x, y, \hat{\theta})}{\partial \theta_l}). \tag{6.9}$$

As in the previous section, the linear classifier with this feature extractor $w^T \varphi_{\hat{\theta}}^{(TK)}(x, y)$ induces the first order approximation to the logarithm of the upper bound of the rank

$$\log \sum_{y' \in G(x) \setminus \{y\}} P^\star(x, y') - \log P^\star(x, y),$$

when w is equal to $(1, (\theta^\star - \hat{\theta})^T)$ and under the assumption that $P^\star(x, y) = P(x, y|\theta^\star)$, for some $\theta^\star$.

We call this kernel the TOP Reranking kernel, because as the TOP kernel of (Tsuda et al., 2002a) it induces the Taylor expansion of log-odds of an event, where in this case the event is choosing a parse y from the candidate list $G(x)$. Note that the original motivation of the TOP kernel is probably not very appropriate for the reranking problem. As we mentioned in section 6.1, the feature extractor of the TOP kernel for classification problems was motivated by the minimization of the estimation error of a posterior probability estimator in the form of the logistic sigmoid. The use of logistic sigmoid estimators induces unnormalized probability estimates and, as we have shown in the previous section, more appropriate estimators in the form of the soft-max function motivate instead the Fisher kernel (6.4). We believe that though the Fisher kernel is preferable if the considered task is probability estimation, the TOP reranking kernel may be a better choice for use in discriminative rerankers because its feature extractor more closely corresponds to their optimization criteria.

In a strictly theoretical sense both of the kernels considered here are more related to the TOP kernel for classification than to any other data-defined kernel because they are derived from posterior probabilities. The main difference here is that they induce input-output feature representations, as needed for reranking, rather than feature representation of the input, as is usual in classification problems.

Note that though these kernels can be applied to arbitrary probabilistic models with continuous first derivatives, they are particularly appropriate for non-linear probabilistic models. In the case of a standard PCFG-like model, the feature space induced by these kernels will be essentially the same as the original feature space of the probability model. Under the logarithmic parametrization, where θ_i is equal to log-probability of production rule P_i in the grammar, the corresponding component of the feature extractor of the Fisher kernel for a tree y will be equal to the number of times this production rule P_i appears in the tree y. For the TOP reranking kernel the components of the feature extractor will be equal to the weighted combinations of these counts, where the weights are computed using the probability estimates given by the PCFG model.

6.2.3 Challenges

It is usual in reranking to use the score from the baseline generative probability model as one of the features in the classifier. However, in order to get representative training data, the available training set is usually divided into chunks and each chunk is parsed with a model trained on the remaining data (see e.g. (Collins & Koo, 2005)). This creates two problems. First, it is time-consuming with latent variables models because training of a latent variable model is considerably more computationally expensive than the training of a PCFG-like model where maximum-likelihood estimates of parameters are just normalized counts of events in the training set. The second problem with this approach is even more serious: both of the considered kernels depend on the parametrization and the lack of identifiability makes feature extractors derived from different probabilistic models incomparable. As an example, for ISBNs, there is no reason to expect that two latent variables with index i in the latent state vector from two different ISBN models correspond to the same feature of the parsing history. This problem would probably be solved by using either the inverse Information matrix in the Fisher kernel or by using the Leave-One-Out kernel, but both of these approaches are not tractable.

Another slightly related problem is probably more of theoretical interest only. As was discussed in section 6.1, the Fisher kernel can be viewed as an approximation to the Leave-One-Out (LOO) kernel, i.e. computing the Fisher kernel between two instances x_i and x_j corresponds to computation of the similarity between the probability models $P(x, y|\theta^{x_i})$ and $P(x, y|\theta^{x_j})$ estimated on all the available data except x_i and x_j, respectively. However, this is true as long as the probability model was originally trained on the training set which included both x_i and x_j. Equivalently, if both examples were not used by the probabilistic model at the training time the Fisher kernel between x_i and x_j derived from it can be shown to be equivalent to "Add-One-In" kernel, where instead of removing examples from the training set the instances are added to the training set. However, in practice kernels are often computed in an asymmetric situation where one of the examples was used in the training and another was not, which is not an entirely valid scenario from the LOO kernel perspective. In classification this problem is solvable by using the transductive setting; in reranking this problem cannot be resolved unless we model the distribution of both correct and incorrect parses. We expect that this problem does not significantly affect our results because, to our knowledge, it does not cause serious performance degradation in classification problems.

In the experimental section we will show that even despite these challenges we manage to achieve significant improvement over the baseline probabilistic model.

6.3 Learning Algorithm

Once we have defined a kernel over parse trees, general techniques for linear classifier optimization can be used to learn the given task. The most sophisticated of these techniques (such as Support Vector Machines) are unfortunately too computationally expensive to be used on large datasets like the Penn Treebank (Marcus et al., 1993). Instead we use an online method which has often been shown to be virtually as good, the Voted Perceptron (VP) (Freund & Schapire, 1998) algorithm. The VP algorithm was originally applied to parse reranking in (Collins & Duffy, 2002) with the Tree kernel. We modify the perceptron training algorithm to make it more suitable for parsing, where zero-one classification loss is not the evaluation measure usually employed. We also develop a variant of the kernels, which is more efficient when used with the VP algorithm.

Given a list of candidate trees, we train the classifier to select the tree with the largest constituent F_1 score, i.e. similarity between the tree in question and the gold standard parse. The Voted Perceptron algorithm is an ensemble method for combining the various intermediate models which are produced during training a perceptron. It demonstrates more stable generalization performance than the normal perceptron algorithm (Freund & Schapire, 1998).[8]

We modify the perceptron algorithm by introducing a new classification loss function. This modification enables us to treat differently the cases where the perceptron predicts a tree with a F_1 score much smaller than that of the top candidate and the cases where the predicted and the top candidates have similar score values. The natural choice for the loss function would be $\Delta(y, y^\star) = F_1(y^\star) - F_1(y)$, where $F_1(y)$ denotes the F_1 score value for the parse tree y and $y^\star$ is the best tree in the candidate list $G(x)$, i.e. the tree with largest constituent F_1 score. This approach is very similar to slack variable rescaling for Support Vector Machines (Tsochantaridis et al., 2004). The update of the standard reranking perceptron can be viewed as the gradient of the error function $\lfloor \max_{y \in G(x)} w^T(\varphi(x, y) - \varphi(x, y^\star)) \rfloor$, where $\lfloor z \rfloor$ is z if $z > 0$ and 0 otherwise. Similarly, the update of the proposed perceptron algorithm can be viewed as the gradient of $\sum_{y \in G(x)} \lfloor \Delta(y, y^\star) w^T(\varphi(x, y) - \varphi(x, y^\star)) \rfloor$. The learning algorithm we employed is presented in figure 6.1.

When applying the described feature extractors to ISBNs with a large training corpus, we face efficiency issues because of the large number of weights in the model. Even though we use only the output layer weights, this vector

[8]We could consider the use of the average perceptron algorithm instead of the VP algorithm, as is often done (Collins & Koo, 2005). In the average perceptron the voting is replaced by averaging the scores assigned by the models from the ensemble. The resulting final score can be computed much more efficiently. However, in our case it was feasible to use VP and we preferred it because the theoretical guarantees demonstrated for the VP algorithm are not directly applicable to the averaged version.

```
$w = 0$
for $x \in D$ do
  for $y \in G(x) \setminus \{y^\star\}$ do
    if $w^T\varphi(x, y) > w^T\varphi(x, y^\star)$ then
      $w = w + \Delta(y, y^\star)(\varphi(x, y^\star - \varphi(x, y))$
    end if
  end for
end for
```

Figure 6.1: The modification of the perceptron algorithm.

grows with the size of the vocabulary, and thus can be large.[9] All the kernels presented in section 6.2 lead to non-sparse feature vectors, i.e. without many zero components. This happens because we compute the derivative of the normalization factor used in the estimation of $P(d_k^t|hist(t, k))$. This normalization factor depends on the output layer weights corresponding to all the possible next decisions (3.11) and (3.16). This makes an application of the VP algorithm infeasible in the case of a large vocabulary.

We can address this problem by freezing the normalization factor when computing the feature vector. Note that we can rewrite the model log-probability of the tree as:

$$\log P(y|\hat{\theta}) = \sum_{t,k} \log q_k^t(d) = \sum_{t,k} \sum_j W_{dj}\mu_j^t + V_d \\ - \sum_{t,k} \sum_{d'} \Phi_{h(t,k)}(d') \exp(\sum_j W_{d'j}\mu_j^t + V_{d'}).$$

We treat the parameters used to compute the first term as different from the parameters used to compute the second term, and we define our kernel only using the parameters in the first term. This means that the second term does not affect the derivatives in the formula for the feature vector $\varphi(x, y)$. Thus the feature vector for the kernel will contain non-zero entries only in the components corresponding to the parser actions which present in the candidate derivation for the sentence and in the first vector component. We have applied this technique to the TOP reranking kernel, the result of which we will call the *efficient TOP reranking kernel.*

6.4 Empirical evaluation

As before, we used the Penn Treebank WSJ corpus (Marcus et al., 1993) to perform empirical experiments on the data-defined kernels. We report results

[9]We performed a set of preliminary experiments where we used a complete set of weights. We did not manage to achieve any improvement in a comparison with the reranker which used only the parameters of the decisions CPDs, i.e. parameters in expression (3.7).

for two different vocabulary sizes, varying in the frequency with which tag-word pairs must occur in the training set in order to be included explicitly in the vocabulary. A frequency threshold of 200 resulted in a vocabulary of 508 tag-word pairs and a threshold of 20 resulted in 4215 tag-word pairs. We denote the probabilistic model trained with the vocabulary of 508 by the ISBN-Freq$\geq$200, the model trained with the vocabulary of 4215 by the ISBN-Freq$\geq$20.

We used only the neural network approximation, i.e. the SSN model (Henderson, 2003), in these experiments. This makes our results directly comparable with results published in (Henderson, 2003), as we use the same baseline model. Also NN approximation is considerably faster than the variational IMF approximation, which made it feasible to perform experiments with VP algorithms.[10]

Testing the probabilistic parser requires using a beam search algorithm, as was described in section 4.3. We set the width of the beam to 40 for both testing of the probabilistic model and generating the candidate list for reranking. For training and testing the kernel models, we provided a candidate list consisting of the top 20 parses found by the generative probabilistic model.

We trained the VP model with all three kernels using the 508 word vocabulary (Fisher-Freq$\geq$200, TOP-Freq$\geq$200, TOP-Eff-Freq$\geq$200) but only the efficient TOP reranking kernel model was trained with the vocabulary of 4215 words (TOP-Eff-Freq$\geq$20). The non-sparsity of the feature vectors for other kernels led to the excessive memory requirements and larger testing time. In each case, the VP model was run for only one epoch. We would expect some improvement if running it for more epochs, as has been empirically demonstrated in other domains (Freund & Schapire, 1998).

To avoid repeated testing on the standard testing set, we first compare the different models with their performance on the validation set. Note that the validation set was not used during learning of the kernel models or for adjustment of any parameters.

Standard measures of accuracy are shown in table 6.1. Both the Fisher kernel and the TOP kernels show better accuracy than the baseline probabilistic model, but only the improvement of the TOP kernels is statistically significant.[11] For

[10]Note that to perform testing with the VP algorithm, the inner product should be computed between the feature extractor for each parse tree in $G(x)$ for the considered sentence x and each "support vector" of the perceptron. By the "support vector" of the perceptron we mean all the parse trees which were scored above the best trees during training, triggering the update of the perceptron vector w. For the entire WSJ and the candidate list of 20 sentences it leads to an order of a million inner product computations per testing sentence. Because of the large parameter space it is not quite feasible to pre-compute them all, which means that in a straightforward implementation estimation of $P(x, y|\theta)$ should be performed on-the-fly. This is not feasible even with the neural network approximation. Instead we use an implementation where we compute $P(x, y|\theta)$ for a chunk of "support vectors" and compute the inner product between them and every parse tree in the testing set. Unfortunately, this is still too expensive to be used with the IMF approximation. It is worth noting, however, that the average perceptron is much faster and our preliminary experiments have suggested very similar results, but, as we mentioned, we preferred the VP algorithm because of its better theoretical motivation.

[11]As in previous parsing experiments, we measured significance with the randomized significance test of (Yeh, 2000).

	LR	LP	F_1
NN-Freq≥200	87.2	88.5	87.8
Fisher-Freq≥200	87.2	88.8	87.9
TOP-Freq≥200	87.3	88.9	88.1
TOP-Eff-Freq≥200	87.3	88.9	88.1
NN-Freq≥20	88.1	89.2	88.6
TOP-Eff-Freq≥20	88.2	89.7	88.9

Table 6.1: Experiments with data-defined kernels: percentage labeled constituent recall (LR), precision (LP), and a combination of both (F_1) on validation set sentences of length at most 100.

the TOP kernel, the improvement over the baseline is about the same with both vocabulary sizes. Also note that the performance of the efficient TOP reranking kernel is the same as that of the original TOP reranking kernel for the smaller vocabulary.

For comparison to previous results, table 6.2 lists the results on the testing set for our best model (TOP-Efficient-Freq≥20)[12] and several other statistical parsers (Collins, 1999; Collins & Duffy, 2002; Collins & Roark, 2004; Henderson, 2003; Charniak, 2000; Collins, 2000; Shen & Joshi, 2004; Shen et al., 2003; Henderson, 2004; Petrov & Klein, 2007; Charniak & Johnson, 2005).[13]

Note first of all that the parser based on the TOP efficient kernel has greater accuracy than the baseline SSN model (Henderson, 2003). When compared to other kernel methods, our approach performs better than those based on the Tree kernel (Collins & Duffy, 2002), though this comparison is probably not entirely appropriate as the baseline models and candidate lists are different.

6.5 Other Kernels for Parse Reranking

The original motivation for data-defined kernels was the construction of feature extractors for structured data to be used for classification with kernel machines (Jaakkola & Haussler, 1998). In parallel, a different framework for constructing kernels for structured data based on the notion of convolution was proposed in (Haussler, 1999). Convolution kernels use the decompositions of structured objects into parts. The convolution kernel between composite examples x_i and x_j is computed using kernels measuring the similarities between parts in x_i and parts in x_j. A convolution kernel for parsing has already been

[12]On sentences of length at most 40, TOP-Efficient-Freq≥20 model gets 89.6% recall and 90.5% precision.

[13]Note, though that results of Petrov and Klein 07 (Petrov & Klein, 2007) are probably not directly comparable with other models as they use approximate Bayes risk minimization. As we will discuss in chapter 8, results of virtually any model can be improved by using Bayes Risk minimization, including other ones in the list. We will show, for example, that, when Bayes risk minimization is applied, the model TOP-Eff-Freq≥ 20 achieves the F_1 score of 90.1% instead 89.6%.

	LR	LP	F_1
Collins 99	88.1	88.3	88.2
Collins and Duffy 02	88.6	88.9	88.7
Collins and Roark 04	88.4	89.1	88.8
Henderson 03 (SSN)	88.8	89.5	89.1
Charniak 00	89.6	89.5	89.5
TOP-Eff-Freq≥20	89.1	90.1	89.6
Collins 00	89.6	89.9	89.7
Shen and Joshi 04	89.5	90.0	89.8
Shen et al. 03	89.7	90.0	89.8
Petrov and Klein 07	90.2	89.9	90.0
Henderson 04	89.8	90.4	90.1
Charniak and Johnson 05	-	-	91.0

* F_1 for previous models may have rounding errors.

Table 6.2: Experiments with data-defined kernels: Percentage labeled constituent recall (LR), precision (LP), and F_1 measure on the entire testing set.

proposed by (Collins & Duffy, 2002). The features of a tree with their Tree Kernel are all its connected tree fragments. The Tree kernel does not require explicit enumeration of these fragments since a dynamic programming algorithm was proposed to compute the kernel more efficiently. The VP algorithm was applied to rerank the output of a Collins parser (Collins, 1999) and demonstrated an improvement over the baseline.

In (Shen et al., 2003) it was pointed out that most of the arbitrary tree fragments allowed by the Tree kernel are linguistically meaningless. The authors suggest that the use of the Tree kernel on the derivation tree for Lexicalized Tree Adjoining Grammar (LTAG) (Joshi & Schabes, 1992) leads to a more linguistically appropriate set of features. Though the idea seems appealing, the provided empirical evaluation is not convincing. The authors report results of a model combination where they combine the output of the classifier trained with the LTAG-Tree kernel with the output of the classifier trained with hand-crafted features originally proposed in (Collins, 2000). Importantly, they did not manage to achieve any improvement over the accuracy of the latter classifier alone.[14]

A different approach to get rid of useless features in the convolution kernels for natural languages was proposed in (Suzuki et al., 2004). They use statistical tests to extract important features. Importantly, this reduced kernel can still be computed using dynamic programming. Even though their approach is applicable to the Tree Kernel, they perform empirical evaluation only on question classification and sentence modality identification, where their dimensionality reduction method improves over using the original feature space.

[14]This classifier with the same set of hand-crafted features was originally evaluated in (Shen & Joshi, 2003).

6.6 Summary of the Chapter

This chapter studies methods for deriving kernels for reranking from a probabilistic model, and demonstrates state-of-the-art accuracy when these methods are applied to parse reranking. Contrary to most of the previous research on kernel methods in parsing, linguistic knowledge does not have to be expressed through a list of features, but instead can be expressed through the design of a probability model. The parameters of this probability model are then trained, so that they reflect what features of trees are relevant to parsing. The kernels are then derived from this trained model in such a way as to maximize its usefulness for reranking.

We performed experiments on parse reranking using the non-linear probabilistic model, SSN, which is equivalent to feed-forward approximation to ISBN as shown in chapter 3. We used a modification of the Voted Perceptron algorithm to perform reranking with the kernel. The results were only 0.2% worse than the best perceptron-based parsing method, and close to the best current statistical parsers.

In recent years, probabilistic models have become commonplace in natural language processing. We believe that this approach to defining kernels will simplify the problem of defining kernels for these tasks and could be very useful for many of them. Also, this method should not be necessarily viewed as an alternative method to defining explicit features for reranking. Standard reranking techniques already rely on the probability estimate from the baseline probability model in addition to new features. Here we show that the use of more rich information about the generative process is also beneficial for such models.

In the following chapter we will show how data-defined kernels can be applied to domain adaptation.

Chapter 7

Data-Defined Kernels for Domain Adaptation

In recent years, significant progress has been made in the area of natural language parsing. This research has focused mostly on the development of statistical parsers trained on large annotated corpora, in particular the Penn Treebank WSJ corpus (Marcus et al., 1993). The best statistical parsers have shown good results on this benchmark, but these statistical parsers demonstrate far worse results when they are applied to data from a different domain (Sekine, 1997; Ratnaparkhi, 1999; Gildea, 2001; Roark & Bacchiani, 2003). There are two different scenarios usually considered in domain adaptation. In one, there is a small amount of training data for the target domain. For this scenario the most standard approaches include count merging (Gildea, 2001; Roark & Bacchiani, 2003), i.e. joining the labeled datasets with possibly some weighting, or model interpolation (Roark & Bacchiani, 2003). In another scenario, there is no labeled data available for the target domain, but only a large amount of unlabeled data. In this case, essentially all the successful domain-adaptation methods are based on some form of self-training, where the unlabeled data comes from the target domain (McClosky et al., 2006b; Reichart & Rappoport, 2007). This scenario was also chosen for the domain adaption track of the recent CoNLL-2007 shared task, where adaption of dependency parsers was considered.

It is important to note that these scenarios are complementary, because in practice it is likely to use both small amount of labeled data for the target domain and vast amounts of unlabeled data. We only address the use of the labeled data. In this chapter, we propose using data-defined kernels and large margin methods to specifically address adapting a parser to a new domain using a small amount of labeled data for the target domain. Data-defined kernels are used to construct a new parser which exploits information from a parser trained on a large out-of-domain corpus. Large margin methods are used to train this parser to optimize performance on a small in-domain corpus.

Large margin methods have demonstrated substantial success in applications to many machine learning problems, because they optimize a measure which is closely related to the expected testing performance. Discriminative classifiers need a definition of a feature representation for the instances of a kernel, which determines a mapping to a, possibly infinite, space where linear classification is performed. In the previous chapter we proposed to apply a class of kernels derived from probabilistic models to the natural language parsing problem. The proposed kernels were constructed using the parameters of a trained probabilistic model. This type of kernel was called a data-defined kernel, because the kernel incorporates information from the data used to train the probabilistic model. We propose to exploit this property to transfer information from a large corpus to a statistical parser for a different domain. Specifically, we propose to train a statistical parser on data including the large corpus, and to derive the kernel from this trained model. Then this derived kernel is used in a large margin classifier trained on the small amount of training data available for the target domain.

In our experiments, we consider two different scenarios for parsers adaptation with labeled data. The first scenario is the pure adaptation case, which we call "transferring". Here we only require a probabilistic model trained on the large corpus. This model is then reparameterized so as to extend the vocabulary to better suit the target domain. The kernel is derived from this reparameterized model. The second scenario is a mixture of parser training and adaptation, which we call "focusing". Here we train a probabilistic model on both the large corpus and the target corpus. The kernel is derived from this trained model. In both scenarios, the kernel is used in a SVM classifier (Tsochantaridis et al., 2004) trained on a small amount of data from the target domain. This classifier is trained to rerank the candidate parses selected by the associated probabilistic model. We use the Penn Treebank Wall Street Journal corpus as the large corpus and individual sections of the Brown corpus as the target corpora (Marcus et al., 1993; Francis & Kucera, 1982). The probabilistic model is, as in previous chapters, ISBNs, and the data-defined kernel is the TOP reranking kernel.

With both scenarios, the resulting parser demonstrates improved accuracy on the target domain over the probabilistic model alone. In additional experiments, we evaluate the hypothesis that the primary issue for porting parsers between domains is differences in the distributions of words in structures, and not in the distributions of the structures themselves. We partition the parameters of the probability model into those which define the distributions of words and those that only involve structural decisions, and derive separate kernels for these two subsets of parameters. The former model achieves virtually identical accuracy to the full model, but the later model does worse, confirming the hypothesis.

7.1 Porting with Data-Defined Kernels

In this chapter, we consider adaptation of a parser trained on a large amount of annotated data to a different domain where only a small amount of annotated data is available. We validate our method in two different scenarios, transferring and focusing. Also we verify the hypothesis that addressing differences between the vocabularies of domains is more important than addressing differences between their syntactic structures.

7.1.1 Transferring to a Different Domain

In the transferring scenario, we are given just a probabilistic model which has been trained on a large corpus from a source domain. The large corpus is not available during adaption, and the small corpus for the target domain is not available during training of the probabilistic model. This is the case of pure parser adaptation, because it only requires the source domain parser, not the source domain corpus. Besides this theoretical significance, this scenario has the advantage that we only need to train a single probabilistic parser, thereby saving on training time and removing the need for access to the large corpus once this training is done. Then any number of parsers for new domains can be trained, using only the small amount of annotated data available for the new domain.

Our proposed adaptation method first constructs a data-defined kernel using the parameters of the trained probabilistic model. A large margin classifier with this kernel is then trained to rerank the top candidate parses produced by the probabilistic model. Only the small target corpus is used during training of this classifier. The resulting parser consists of the original parser plus a very computationally cheap procedure to rerank its best parses.

Whereas training of standard large margin methods, like SVMs, is not feasible on a large corpus, it is quite tractable to train them on a small target corpus.[1] Also, the choice of the large margin classifier is motivated by their good generalization properties on small datasets, on which accurate probabilistic models are usually difficult to learn.

This method is related to the use of the Fisher kernel for semi-supervised training in fields other than parsing and structured classification (Seeger, 2000). There, one of the approaches is to train a generative model of the input on a large unlabeled dataset and then train a classifier with the Fisher kernel constructed from this generative model on available annotated data. However, as we discussed in section 6.2, this approach is not applicable in our case because we have to consider only kernels over input-output pairs, which limits us to the use of annotated data only.

We hypothesize that differences in vocabulary across domains is one of the main difficulties with parser adaptation. To address this problem, we propose

[1]In (Shen & Joshi, 2003) it was proposed to use an ensemble of SVMs trained the WSJ corpus, but we believe that the generalization performance of the resulting classifier is compromised in this approach.

constructing the kernel from a probabilistic model which has been reparameterized to better suit the target domain vocabulary. As in other lexicalized statistical parsers, the probabilistic model we use treats words which are not frequent enough in the training set as 'unknown' words. Thus there are no parameters in this model which are specifically for these words. When we consider a different target domain, a substantial proportion of the words in the target domain are treated as unknown words, which makes the parser only weakly lexicalized for this domain.

To address this problem, we reparameterize the probability model so as to add specific parameters for the words which have high enough frequency in the target domain training set but are treated as unknown words by the original probabilistic model. These new parameters all have the same values as their associated unknown words, so the probability distribution specified by the model does not change. However, when a kernel is defined with this reparameterized model, the kernel's feature extractor includes features specific to these words, so the training of a large margin classifier can exploit differences between these words in the target domain. Expanding the vocabulary in this way is also justified for computational reasons; the speed of the probabilistic model we use is greatly affected by vocabulary size, but the large margin method is not.

7.1.2 Focusing on a Subdomain

In the focusing scenario, we are given the large corpus from the source domain. We may also be given a parsing model, but as with other approaches to this problem we simply throw this parsing model away and train a new one on the combination of the source and target domain data. Previous work (Roark & Bacchiani, 2003) has shown that better accuracy can be achieved by finding the optimal re-weighting between these two datasets, but this issue is orthogonal to our method, so we only consider equal weighting. After this training phase, we still want to optimize the parser for only the target domain.

Once we have the trained parsing model, our proposed adaptation method proceeds the same way in this scenario as in transferring. However, because the original training set already includes the vocabulary from the target domain, the reparametrization approach defined in the preceding section is not necessary so we do not perform it. This reparametrization could be applied here, thereby allowing us to use a statistical parser with a smaller vocabulary, which can be more computationally efficient both during training and testing. However, we would expect better accuracy of the combined system if the same large vocabulary is used both by the probabilistic parser and the kernel method.

7.1.3 Vocabulary versus Structure

It is commonly believed that differences in vocabulary distributions between domains affects the adapted parser performance more significantly than the differences in syntactic structure distributions. We would like to test this hypothesis in our framework. The ISBN probabilistic model allows us to distinguish between those parameters responsible for the distributions of individual vocabulary items, and those parameters responsible for the distributions of structural decisions, as described in more details in section 7.2. We train two additional models, one which uses a kernel defined in terms of only vocabulary parameters, and one which uses a kernel defined in terms of only structure parameters. By comparing the performance of these models and the model with the combined kernel, we can draw conclusion on the relative importance of vocabulary and syntactic structures for parser adaptation.

7.2 An Application to ISBNs

Data-defined kernels can be applied to any kind of parametrized probabilistic model, but they are particularly interesting for latent variable models. Without latent variables, as we discussed in the previous chapter the features of the data-defined kernel (except for the first feature) are a function of the counts used to estimate the model. With latent variables, the meaning of the variable (not just its value) is learned from the data, and the associated features of the data-defined kernel capture this induced meaning.

We use the NN approximation to ISBNs as our latent variable model. The complete set of model weights is not used to define the kernel, but instead reparametrization is applied to define a third level of parametrization which only includes the network's output layer weights. As suggested in the previous chapter, use of the complete set of weights does not lead to any improvement of the resulting reranker and makes the reranker training more computationally expensive.

Furthermore, to assess the contribution of vocabulary and syntactic structure differences (see section 7.1.3), we divide set of the parameters into vocabulary parameters and structural parameters. We consider the weights used in the estimation of the probability of the next word given the history representation as vocabulary parameters, and the weights used in the estimation of structural decision probabilities as structural parameters. We define the kernel with structural features as using only structural parameters, and the kernel with vocabulary features as using only vocabulary parameters.

7.3 Experimental Results

We used the Penn Treebank WSJ corpus and the Brown corpus to evaluate our approach. We used the standard division of the WSJ corpus into training, validation, and testing sets. In the Brown corpus we ran separate experiments for sections F (informative prose: popular lore), K (imaginative prose: general fiction), N (imaginative prose: adventure and western fiction), and P (imaginative prose: romance and love story). These sections were selected because they are sufficiently large, and because they appeared to be maximally different from each other and from WSJ text. In each Brown corpus section, we selected every

	testing	training	validation
WSJ	2,416 (54,268)	39,832 (910,196)	1,346 (31,507)
Brown F	1,054 (23,722)	2,005 (44,928)	105 (2,300)
Brown K	1,293 (21,215)	2,459 (39,823)	129 (1,971)
Brown N	1,471 (22,142)	2,797 (42,071)	137 (2,025)
Brown P	1,314 (21,763)	2,503 (41,112)	125 (1,943)

Table 7.1: Domain adaptation: number of sentences (words) for each dataset.

third sentence for testing. From the remaining sentences, we used 1 sentence out of 20 for the validation set, and the remainder for training. The resulting datasets sizes are presented in table 7.1.

For the large margin classifier, we used the SVM-Struct (Tsochantaridis et al., 2004) implementation of SVM, which rescales the margin with F_1 measure of bracketed constituents (see (Tsochantaridis et al., 2004) for details). Linear slack penalty was employed.[2]

7.3.1 Experiments on Transferring across Domains

To evaluate the pure adaptation scenario (transferring), described in section 7.1.1, we trained the neural network approximation to ISBN model (i.e. SSN model of Henderson (Henderson, 2003)) on the WSJ corpus. For each tag, there is an unknown-word vocabulary item which is used for all those words not sufficiently frequent with that tag to be included individually in the vocabulary. In the vocabulary of the parser, we included the unknown-word items and the words which occurred in the training set at least 20 times. This led to the vocabulary of 4,215 tag-word pairs.

We derived the kernel from the trained model for each target section (F, K, N, P) using reparametrization discussed in section 7.1.1: we included in the vocabulary all the words which occurred at least twice in the training set of the corresponding section. This approach led to a smaller vocabulary than that of the initial parser but specifically tied to the target domain (3,613, 2,789, 2,820 and 2,553 tag-word pairs for sections F, K, N and P respectively). There is no sense in including the words from the WSJ which do not appear in the Brown section training set because the classifier won't be able to learn the corresponding components of its decision vector. The results for the original

[2]Training of the SVM takes about 3 hours on a standard desktop PC. Running the SVM, i.e. the linear classifier, is very fast, once the probabilistic model has finished computing the probabilities needed to select the candidate parses.

probabilistic model (NN-WSJ) and for the kernel method (TOP-Transfer) on the testing set of each section are presented in table 7.2.

To evaluate the relative contribution of our adaptation technique versus the use of the TOP kernel alone, we also used this TOP kernel to train an SVM on the WSJ corpus. We trained the SVM on data from the development set and section 0, so that the size of this dataset (3,267 sentences) was about the same as for each Brown section.[3] This gave us a "TOP-WSJ" model, which we tested on each of the four Brown sections. In each case, the TOP-WSJ model did worse than the original NN-WSJ model, as shown in table 7.2. This makes it clear that we are getting no improvement from simply using a TOP kernel alone or simply using more data, and all our improvement is from the proposed adaptation method.

7.3.2 Experiments on Focusing on a Subdomain

To perform the experiments on the approach suggested in section 7.1.2 (focusing), we trained the parser on the WSJ training set joined with the training set of the corresponding section. We included in the vocabulary only words which appeared in the joint training set at least 20 times. Resulting vocabularies comprised 4,386, 4,365, 4,367 and 4,348 for sections F, K, N and P, respectively.[4] Experiments were done in the same way as for the parser transferring approach, but reparametrization was not performed. Standard measures of accuracy for the original probabilistic model (NN-WSJ+Br) and the kernel method (TOP-Focus) are also shown in table 7.2.

For the sake of comparison, we also trained the NN parser on only training data from one of the Brown corpus sections (section P), producing a "NN-Brown" model. This model achieved an F_1 measure of only 81.0% for the P section testing set, which is worse than all the other models and is 3% lower than our best results on this testing set (TOP-Focus). This result underlines the need to adapt parsers from domains in which there are large annotated datasets.

7.3.3 Experiments Comparing Vocabulary to Structure

We conducted the same set of experiments with the kernel with vocabulary features (TOP-Voc-Transfer and TOP-Voc-Focus) and with the kernel with the structural features (TOP-Str-Transfer and TOP-Str-Focus). Average results for classifiers with these kernels, as well as for the original kernel and the baseline, are presented in table 7.3.

[3]We think that using an equivalently sized dataset provides a fair test of the contribution of the TOP kernel alone. It would also not be computationally tractable to train an SVM on the full WSJ dataset without using different training techniques, which would then compromise the comparison.

[4]We would expect some improvement if we used a smaller threshold on the target domain, but preliminary results suggest that this improvement would be small.

	section	LR	LP	F_1
TOP-WSJ	F	83.9	84.9	84.4
NN-WSJ	F	84.4	85.2	84.8
TOP-Transfer	F	84.5	85.6	85.0
NN-WSJ+Br	F	84.2	85.2	84.7
TOP-Focus	F	84.6	86.0	85.3
TOP-WSJ	K	81.8	82.3	82.1
NN-WSJ	K	82.2	82.6	82.4
TOP-Transfer	K	82.4	83.5	83.0
NN-WSJ+Br	K	83.1	84.2	83.6
TOP-Focus	K	83.6	85.0	84.3
TOP-WSJ	N	83.3	84.5	83.9
NN-WSJ	N	83.5	84.6	84.1
TOP-Transfer	N	84.3	85.7	85.0
NN-WSJ+Br	N	85.0	86.5	85.7
TOP-Focus	N	85.0	86.7	85.8
TOP-WSJ	P	81.3	82.1	81.7
NN-WSJ	P	82.3	83.0	82.6
TOP-Transfer	P	82.7	83.8	83.2
NN-WSJ+Br	P	83.1	84.3	83.7
TOP-Focus	P	83.3	84.8	84.0

Table 7.2: Domain adaptation: percentage labeled constituent recall (LR), precision (LP), and F_1 measure on the individual test sets.

7.3.4 Discussion of Results

For the experiments which directly test the usefulness of our proposed adaptation technique (NN-WSJ versus TOP-Transfer), our technique demonstrated improvement for each of the Brown sections (table 7.2), and this improvement was significant for three out of four of the sections (K, N, and P).[5] This demonstrates that data-defined kernels are an effective way to adapt parsers to a new domain.

For the experiments which combine training a new probability model with our adaptation technique (NN-WSJ+Br versus TOP-Focus), our technique still demonstrated improvement over training alone. There was improvement for each of the Brown sections, and this improvement was significant for two out of four of the sections (F and K). This demonstrates that, even when the probability model is well suited to the target domain, there is still room for improvement from using data-defined kernels to optimize the parser specifically to the target

[5]We measured significance in F_1 measure at the 5% level with the randomized significance test of (Yeh, 2000). We think that the reason the improvement on section F was only significant at the 10% level was that the baseline model (NN-WSJ) was particularly lucky, as indicated by the fact that it did even better than the model trained on the combination of datasets (NN-WSJ+Br).

	LR	LP	F_1
NN-WSJ	83.1	83.8	83.5
TOP-Transfer	83.5	84.7	84.1
TOP-Voc-Transfer	83.5	84.7	84.1
TOP-Str-Transfer	83.1	84.3	83.7
NN-WSJ+Br	83.8	85.0	84.4
TOP-Focus	84.1	85.6	84.9
TOP-Voc-Focus	84.1	85.6	84.8
TOP-Str-Focus	83.9	85.4	84.7

Table 7.3: Domain adaptation: average accuracy of the models on chapters F, K, N and P of the Brown corpus.

domain without losing information about the source domain.

One potential criticism of these conclusions is that the improvement could be the result of reranking with the TOP kernel, and have nothing to do with adaptation. The lack of an improvement in the TOP-WSJ results discussed in section 7.3.1 clearly shows that this cannot be the explanation. The opposite criticism is that the improvement could be the result of optimizing to the target domain alone. The poor performance of the NN-Brown model discussed in section 7.3.2 makes it clear that this also cannot be the explanation. Therefore reranking with data defined kernels must be both effective at preserving information about the source domain and effective at specializing to the target domain.

The experiments which test the hypothesis that differences in vocabulary distributions are more important than difference in syntactic structure distributions confirm this belief. Results for the classifier which uses the kernel with only vocabulary features are better than those for structural features in each of the four sections with both the Transfer and Focus scenarios. In addition, comparing the results of TOP-Transfer with TOP-Voc-Transfer and TOP-Focus with TOP-Voc-Focus, we can see that adding structural features in TOP-Focus and TOP-Transfer leads to virtually no improvement. This suggest that differences in vocabulary distributions are the only issue we need to address, although this result could possibly also be an indication that our method did not sufficiently exploit structural differences.

In this chapter we concentrate on the situation where a parser is needed for a restricted target domain, for which only a small amount of data is available. We believe that this is the task which is of greatest practical interest. For this reason we do not run experiments on the task considered in (Gildea, 2001) and (Roark & Bacchiani, 2003), where they are adapting from the restricted domain of the WSJ corpus to the more varied domain of the Brown corpus as a whole. However, to help emphasize the success of our proposed adaptation method, it is relevant to show that even our baseline models are performing better than this previous work on parser adaptation. We trained and tested the

ISBN parser in their "de-focusing" scenario using the same datasets as (Roark & Bacchiani, 2003). When trained only on the WSJ data (analogously to the NN-WSJ baseline for TOP-Transfer) it achieves results of 82.9%/83.4% LR/LP and 83.2% F_1, and when trained on data from both domains (analogously to the NN-WSJ+Br baselines for TOP-Focus) it achieves results of 86.3%/87.6% LR/LP and 87.0% F_1. These results represent a 2.2% and 1.3% increase in F_1 over the best previous results, respectively (see the discussion of (Roark & Bacchiani, 2003) below).

7.4 Related Work

Though most research in the field of parsing is still focused on the Wall Street Journal corpus, following its success in other problems in natural language processing (see e.g. (Blitzer et al., 2006)), domain adaptation has become a hot topic in natural language parsing.

Early work on domain-adaptation was mostly focused on analysis of parser performance and robustness of syntactic features when varying domains. In (Sekine, 1997) it was shown that the distribution of structures differs greatly from domain to domain and that the use of data from the target or similar domains leads to much better parsing accuracy than the use of data from broader or different domains. (Ratnaparkhi, 1999) performed adaptation experiments with a Maximum Entropy parser and demonstrated that the parser trained on WSJ achieves far worse results on the Brown corpus sections. Adding a small amount of labeled data from the target domain improves the results, but accuracy is still much lower than the results on the WSJ. They reported results when their parser was trained on the WSJ training set plus a portion of 2,000 sentences from a Brown corpus section. They achieved 80.9%/80.3% recall/precision for section K, and 80.6%/81.3% for section N.[6] Our analogous method (TOP-Focus) achieved much better accuracy (3.7% and 4.9% better F_1, respectively).

In addition to adaptation experiments with the parsing model of (Collins, 1997), (Gildea, 2001) provided a comprehensive analysis of parser portability. On the basis of this analysis, a technique for parameter pruning was proposed leading to a significant reduction in the model size without a large decrease of accuracy. (Gildea, 2001) only reports results on sentences of 40 or less words on all the Brown corpus sections combined, for which he reports 80.3%/81.0% recall/precision when training only on data from the WSJ corpus, and 83.9%/84.8% when training on data from the WSJ corpus and all sections of the Brown corpus. (Roark & Bacchiani, 2003) performed experiments on supervised and unsupervised PCFG adaptation to the target domain. They propose to use the statistics from a source domain to define priors over weights. However, in their experiments they used only trivial sub-cases of this approach, namely,

[6]The sizes of Brown sections reported in (Ratnaparkhi, 1999) do not match the sizes of sections distributed in the Penn Treebank 3.0 package, so we could not replicate their split. We suspect that a preliminary version of the corpus was used for their experiments.

count merging and model interpolation. They achieved very good improvement over their baseline and over (Gildea, 2001), but the absolute accuracies were still relatively low (as discussed above). They report results with combined Brown data (on sentences of 100 words or less), achieving 81.3%/80.9% when training only on the WSJ corpus and 85.4%/85.9% with their best method using the data from both domains.

All the above mentioned approaches with the exception of unsupervised PCFG adaptation of (Roark & Bacchiani, 2003), were mostly focused on the use of labeled data from the target domain to adapt the parser. Use of unlabeled data from the target data was not seriously considered until very recently because of lack of success with the closely related problem: semi-supervised learning. In semi-supervised learning, a model uses both labeled and unlabeled data, but the distribution of the data is normally assumed to be identical in both parts of the dataset. One of the most common techniques to perform semi-supervised learning is self-training. In self-training, in its basic form, the output of a model on unlabeled data is added to the training set and the model is retrained on this new training set. Different modifications exist, e.g. the process can be repeated iteratively, or only sentences where high-accuracy of the model is expected are added to the training set. A similar technique, co-training (Blum & Mitchell, 1998), uses instead of a single model two or more different statistical models. Output of these models is added to the training set only if the models have high degree of agreement on the considered example. First attempts to apply self-training to parsing were not particularly successful (Charniak, 1997; Steedman et al., 2003), modest improvement, if any, was achieved. Co-training (Blum & Mitchell, 1998), though, was shown to be useful both for the single-domain set-up and domain adaptation. (Steedman et al., 2003) performed portability experiments where they were adapting models trained on the Brown corpus to the WSJ domain. They used a very small amount of labeled data - only 1000 sentences - and achieved an improvement on WSJ of about 3% (from 75.4% to 78.2%).

The first encouraging results on semi-supervised training were demonstrated in (McClosky et al., 2006a), where a non-standard semi-supervised scenario was used. They used a generative parser with a discriminative reranker (Charniak & Johnson, 2005) to select the best parse tree for each sentence in unlabeled data. Then these trees were used to retrain the generative parser, whereas reranker was left trained only on the labeled data. The model achieved 12% error reduction over the supervised baseline and the best reported F_1 measure of 92.1% on the standard WSJ test set. This approach was later successfully applied to the parser-adaptation problem in (McClosky et al., 2006b). They considered adaptation from the WSJ domain to the Brown corpus as a whole. They showed that self-training even on the data from a different domain improves accuracy on the target domain, which can be explained by the fact that self-training helps to solve data-sparsity of the parser. They achieved improvement of 1.7% from 85.8% to 87.5% in F_1 measure on Brown by using 24 millions sentences from North American News Corpus as unlabeled data in self-training. Note, though, that their self-training approach is somewhat similar to co-training because of the crucial importance of using both a reranker and a generative model. However, very recently a generic semi-supervised procedure was shown to be useful both in the standard semi-supervised training and in domain-adaptation (Reichart & Rappoport, 2007). They achieved significant improvement from using unlabeled data but, importantly, unlike (McClosky et al., 2006a; McClosky et al., 2006b), they used very small amount of labeled data: from 100 to 2000 sentences.

(Kawahara & Uchimoto, 2008) considered a different method for semi-supervised learning of dependency parsing models. Unlabeled data is first parsed by an initial supervised model, its output is used to gather statistics about dependency relations. The obtained statistics of structural features are then used as additional features for learning of a new model on labeled data. Authors propose to gather statistics only about short dependencies - parsers are known to produce much more accurate predictions for short relations than for long relations as we discussed in chapter 5. Experiments demonstrate that the semi-supervised model significantly improves over the initial model. Though authors do not perform any experiments on domain-portability problem, their idea can probably be extended to the domain-adaptation problem, where both labeled and unlabeled data is available for the target domain.

A very promising and general domain-adaptation technique based on Alternative Structure Optimization (ASO) (Ando & Zhang, 2005) was proposed in (Blitzer et al., 2006). In (Ando & Zhang, 2005) unlabeled data is used to create many auxiliary learning problems “similar” to the problem of interest. E.g., consider prediction of a word from a context, this problem can be regarded as similar to PoS tagging and the training set for the auxiliary problems can be generated from unlabeled sentences. The auxiliary problems are used to induced a reduced feature representation of the input space useful in all these auxiliary problems. Motivation is that if this feature representation was useful for a number of different problems than it is expected to be useful for the real problem of interest. Therefore, this reduced representation is then used together with the original representation to train models on labeled data for this real problem. In (Blitzer et al., 2006) they apply this approach to domain-adaptation by using unlabeled data both from the source domain and the target domain, in hope to induce similar feature representation for similar entities in target and source domains, even though the entities from the target domain may never appear in the labeled data. Because of the above motivation, they call their approach domain adaptation methods Structured Correspondence Learning (SCL). Another difference of SCL from ASO is that they use a slightly different feature induction procedure, which can be viewed as a single first step of ASO. Though they achieved very encouraging results in PoS tagging experiments, usefulness of the SCL method to more complex structured prediction problems, such as parsing, is yet to be demonstrated. It is not trivial to create these auxiliary tasks on unlabeled data, so that to make them similar to the parsing problem. We are aware of only a single application of this technique to parsing: (Shimizu & Nakagawa, 2007) tried to use auxiliary tasks of predicting existence of a preposition, a determiner or an auxiliary verb between each pair of words in a sentence, they then used the induced feature space in the MST dependency parsing model (McDonald et al., 2005). They did not achieve any improvement over the supervised baseline in adaptation problems, probably because the created auxiliary tasks do not have much in common with the dependency parsing problem.

Domain-adaptation with use of only unlabeled data from the target domain was the topic of a track in CoNLL-2007 shared task on dependency parsing (Nivre et al., 2007a). The considered approaches included the above mentioned SCL method (Shimizu & Nakagawa, 2007), ensemble-methods similar in motivation to co-training (Sagae & Tsujii, 2007; Dredze et al., 2007) and self-training (Watson & Briscoe, 2007). For details we refer the reader to the shared task overview paper (Nivre et al., 2007a), which provides a description of the considered problem, datasets and proposed methods. One criticism of this shared task design (Dredze et al., 2007) was that the annotation guidelines used in the source domain (from which the labeled data was coming) and the target domain were not quite compatible. The authors argued that changes in the conditional probability $P(y|x)$ were more important factor in the performance degradation, then changes in the distribution of input x. Clearly, this "transfer learning" problem cannot be properly addressed without having labeled data from the target domain.

7.5 Summary of the Chapter

This chapter proposes a novel technique for improving portability of non-linear models for natural language parsing, applying parse reranking with data-defined kernels. First a probabilistic model of parsing is trained on all the available data, including a large set of data from the source domain. This model is used to define a kernel over parse trees. Then this kernel is used in a large margin classifier trained on a small set of data only from the target domain. This classifier is used to rerank the top parses produced by the probabilistic model on the target domain. Experiments with the ISBN parser demonstrate that this approach leads to improved parser accuracy on the target domain, without any significant increase in computational cost. This approach uses small amounts of labeled data from target domain and can be used together with semi-supervised methods such as self-training to integrate unlabeled data from the target domain. Also we demonstrated that the primary issue for adapting parsers between domains is differences in the distributions of words in structures, and not in the distributions of the structures themselves.

Chapter 8

Minimum Bayes Risk Decoding

As we discussed in previous chapters, candidate reranking is widely used approach to improve performance of a generative model. Instead of attempting to select the most probable structure, a model can predict a structure which minimizes expected loss. Loss minimization, or minimum Bayes risk (MBR) decoding, was previously studied in the context of PCFGs (Goodman, 1996) with a linear loss function (labeled constituent recall measure), but there was no attempt to apply it to more complex non-linear models and different loss functions. In this chapter we explain how MBR decoding can be implemented with an arbitrary statistical model and different loss functions both for phrase-structure and dependency parsing. The obtained results suggest that even when the loss function used in MBR decoding is only weakly related to the loss function used in evaluation, results are still significantly improved with respect to standard Maximum A-posteriori Probability (MAP) decoding. This conclusion makes the use of MBR decoding attractive for natural language parsing tasks, where the true loss function is not usually known.

The discriminative classifier used in the candidate reranking scenario can often be regarded as a probabilistic model, with the most probable candidate according to this model being selected (i.e. maximum a-posteriori probability decoding). A different use of the n-best lists has been considered in the machine translation and speech processing areas. Here candidate lists are often used to approximate conditional risk (Kumar & Byrne, 2004; Stolcke et al., 1997). Then either the candidate with the smallest approximate conditional risk is selected, or a search is performed for an optimal structure under this approximation. The latter is known as Minimum Bayes Risk (MBR) decoding.

MBR decoding differs from Maximum A-posteriori Probability (MAP) decoding only when the loss function employed differs from 0-1 loss, as is the standard situation in complex structure prediction tasks and, particularly, in parsing. However, the only attempt until recently to optimize Bayes risk in parsing

(Goodman, 1996) only considered the constituent tree parsing task with binarized unlexicalized probabilistic context-free grammars (PCFG). They achieved very low accuracy with their over-simplistic model. Recent proposals, which were parallel or succeeded the publication of research presented in this chapter (Titov & Henderson, 2006b; Titov & Henderson, 2006a), are discussed in section 8.6.

We focus on MBR decoding on the basis of n-best lists, without placing any constraints on what probabilistic model is used. This allows for use of this approach with non linear probabilistic models which do not factorize over components of the tree. We consider both the constituent and dependency tree parsing tasks. A novel algorithm is proposed for prediction of a Minimum Bayes risk (MBR) tree given a candidate list and a standard measure of accuracy, F_1 measure on labeled constituents. In dependency parsing, we show that MBR decoding with standard accuracy measures can be done using previously proposed algorithms. The MBR decoding approach is evaluated on the standard Wall Street Journal (WSJ) parsing task with different generative and discriminative models. In all the cases a significant improvement over MAP decoding is achieved. When we use the best parsing method for this task (Charniak & Johnson, 2005), our approach for MBR decoding demonstrates the best published result for WSJ constituent parsing task, a F_1 score of 91.7%.[1]

8.1 Bayes risk minimization

The Bayes risk of a model $y = h(x)$ is defined as:

$$R(h) = E_{x,y'}\Delta(y', h(x)),$$

where the expectation is taken over all the possible sentences x and trees y' and $\Delta(y', y)$ denotes a loss incurred by selecting a tree y for a sentence x when the correct tree is y'. It follows that an optimal (Bayes) classifier $h^\star$ is one which chooses the tree y that minimizes the conditional risk:

$$h^\star(x) = \arg\min_y \sum_{y'} P(y'|x)\Delta(y', y),$$

where minimization is performed over all possible trees for sentence x.

In order to estimate the conditional risk for a tree y on the basis of a candidate list, we have to make the assumption that the risk is proportional to the average loss in respect to the elements of the candidate list:

$$\sum_{y'} P(y'|x)\Delta(y, y') \approx \alpha \frac{\sum_{y' \in G(x)} P(y'|x)\Delta(y', y)}{\sum_{y' \in G(x)} P(y'|x)},$$

[1]We compare only fully supervised approaches here. Semi-supervised model of McClosky et al. (McClosky et al., 2006a) achieved the F_1 score of 92.1% using additionally 1.75 mln of unlabeled sentences fron North American News Text Corpus (Graff, 1995), as it was discussed in the previous chapter.

where $G(x)$ denotes a candidate list provided by a baseline model for the input x and α is some constant value. Replacing the true probabilities with their estimates $P_\theta(y'|x)$, we can define the model

$$\hat{h}(x) = \arg\min_{y} \sum_{y' \in G(x)} P_\theta(y'|x)\Delta(y', y). \tag{8.1}$$

Probability estimates can be provided by the baseline model, which can be either conditional $P_\theta(y|x)$ or joint $P_\theta(x, y)$, since the normalization factor does not affect $\hat{h}(x)$. As will be discussed in section 8.4, the probability estimates can also be computed from the most widely used non-probabilistic discriminative reranking models.

In the following two sections, we will consider how MBR decoding (equation (8.1)) can be performed for standard loss functions in the constituent and dependency tree parsing tasks. For more complex loss functions or very tight constraints on the running time, a simpler approach can be used, where a parse tree is selected from the candidate list:

$$\bar{h}(x) = \arg\min_{y \in G(x)} \sum_{y' \in G(x)} P_\theta(y'|x)\Delta(y, y'). \tag{8.2}$$

We will refer to this approach in equation (8.2) as MBR reranking, and to the complete decoding approach in equation (8.1) as MBR prediction.

A statistically more motivated approach would be to use samples from the baseline model instead of a distribution in the list of top candidates. This would guarantee that the estimator of the risk is unbiased. Nevertheless, the convergence rate of the risk estimator might still be worse when using samples than when using top candidates. Importantly, it is not feasible to obtain samples from many generative models, including ISBNs considered in the previous chapters, and also it would not allow us to use scores from discriminative rerankers.[2] However if one is interested in minimizing risk using the standard PCFG model as a baseline probabilistic model, then sampling algorithms considered in (Goodman, 1998; Finkel et al., 2006) can be used to obtain the list of parses.

8.2 Constituent tree parsing

Accuracy of a parser is evaluated by comparison of a tree y predicted by the parser to a gold-standard tree $y^\star$ provided by a human annotator. As we described in section 2.1.1, the standard measure of accuracies of constituent parsing involve the number of equally labeled brackets. The Labeled Recall (LR) measure is defined as $|y \bigcap y^\star|/|y^\star|$, Labeled Precision (LP) as $|y \bigcap y^\star|/|y|$ and

[2]We could train a discriminative reranker on parse trees sampled from a generative model and use scores assigned to the tree by the discriminative reranker to compute probability estimates. However, these trees would not be samples from the probability models defined by these estimates, and therefore estimators still may be biased.

their harmonic mean $F_1(y^\star, y) = 2|y \bigcap y^\star|/(|y^\star| + |y|)$. Individual minimization of the LP measure is not possible and individual minimization of the LR measure will result in a strong preference towards large trees.[3] It follows that the natural choice is to use the loss function $\Delta(y, y') = (1 - F_1(y, y'))$ in MBR decoding methods. Then the expression (8.1) for this loss function can be rewritten as

$$\hat{h}(x) = \arg\max_y \sum_{y' \in G(x)} P_\theta(y'|x) \frac{|y \bigcap y'|}{|y'| + |y|}.$$

We can further transform this into:

$$\hat{h}(x) = \arg\max_y \sum_{b \in y} w_{|y|}(b), \tag{8.3}$$

where the bracket weight $w_{|y|}(b)$ is defined as

$$w_m(b) = \sum_{y' \in G(x)} I_{y'}(b) \frac{P_\theta(y'|x)}{|y'| + m},$$

and I_X denotes the indicator function for a set X. Thus, for a fixed number of brackets m, the task of finding an optimal tree with m brackets is equivalent to the search for a set of m non-crossing brackets with maximum weight, where the weight of a set is the sum of weights of its elements $w_m(b)$.[4] We can solve this task with dynamic programming. Then the complete maximization in expression (8.3) can be done by considering an appropriate range of values for m.

Let $C[p, l, s]$ be a dynamic programming table that stores the score of the maximal subforest with p internal nodes (brackets) spanning a subsequence from position s to position $s + l - 1$. We denote as v_j a word at position j in the sentence x. For each split of the subsequence $v_s, ..., v_{s+l-1}$ defined by index k, we can distinguish 3 sets of brackets: brackets in the forest spanning the left subsequence $v_s, ..., v_{s+k-1}$, brackets in the forest spanning the right subsequence $v_{s+k}, ..., v_{s+l-1}$ and brackets of the form $(X, s, s + l - 1)$ spanning the whole subsequence $v_s, ..., v_{s+l-1}$. All the possible sizes of these sets which sum up to p should be considered. If we consider outputting t brackets of the form $(X, s, s + l - 1)$, then the optimal choice is to select from the candidates trees the t brackets with the largest weights $w_m(b)$. The recursive computation of $C[p, l, s]$ can be summarized in the following expression:

$$C[p, l, s] = \max_{\substack{0 \le t \le min(N[l,s],p) \\ 0 \le u \le p-t \\ 1 \le k \le l-1}} \Big(\sum_{1 \le i \le t} w_m(b[l, s, i]) + C[u, k, s] + C[p - u - t, l - k, s + k] \Big),$$

[3]In (Goodman, 1996) optimization of a recall measure was considered, but their recall measure was based on PCFG binary production rules rather then on labeled brackets. It does not directly correspond to the usual measures used in parsing, but allows to simplify decoding.

[4]Note that any set of m non-crossing brackets can be used to define a tree by adding a root bracketing which spans the whole sentence. The root bracketing is ignored in the standard loss measures, and we ignore it when measuring the size of a tree.

where u is the number of brackets in the a subforest spanning the left subsequence, $b[l, s, 1]$,..., $b[l, s, N[l, s]]$ is the set of brackets of the form $(X, s, s+l-1)$ sorted in decreasing order of their weights, and $N[l, s]$ is the size of this set. The weight of the optimal forest of m elements is given by $C[m, n, 1]$ where n is length of the sentence. Pseudo-code for the algorithm is given in Figure 8.1. A simple modification of this algorithm that keeps track of optimal values for k, t and u, can be used to recover a tree. If we note that $N[l, s]$ is bounded by the number of possible node labels, then it is easy to see that the algorithm has a complexity of $O(m^2 n^3)$.

It can be shown that to perform maximization across values of m in (8.3) only a fixed range of values needs to be considered:

$$\lceil \frac{\beta}{2-\beta} \min_{y' \in G(x)} |y'| \rceil \leq m \leq \lfloor \frac{(2-\beta)}{\beta} \max_{y' \in G(x)} |y'| \rfloor, \quad \beta = \frac{\max_{y' \in G(x)} P_\theta(y'|x)}{\sum_{y' \in G(x)} P_\theta(y'|x)}. \quad (8.4)$$

Therefore, the total runtime of the algorithm for finding an optimal tree is $O(n^6 \log^3 n)$. In practice this algorithm is quite tractable, because a multiplicative constant is small. This complexity would be prohibitive for a parser with a large number of node labels, but MBR decoding does not need to choose them from the full set for each span but only from a small set of labels appearing in the candidates. Also, in practice a much smaller range of m values than stated in (8.4) can be considered. In all the experiments discussed in section 8.5, average MBR decoding time on a standard desktop PC was always below 0.2 seconds per sentence.[5]

8.3 Dependency tree parsing

As we discussed in section 2.1.2, it is common to distinguish project and non-projective dependency parsing problems. In this section we will briefly discuss how Bayes risk minimization can be performed both with the projectivity requirement and without it.

Standard measures of accuracy for dependency parsing are labeled and unlabeled relation accuracies. The labeled accuracy is the fraction of relations where both edges and labels match, and the unlabeled accuracy is defined similarly but without requiring labels to match. We will consider optimization of unlabeled accuracy but it is trivial to optimize the labeled accuracy in the same way. If we consider a tree as a set of edges, then expression (8.1) can be rewritten as

$$\hat{h}(x) = \arg\max_y \sum_{y' \in G(x)} P_\theta(y'|x) |y \bigcap y'|.$$

[5]A simple modification of this algorithm can be used for MBR-decoding with a PCFG model, in this case there is no need to use a candidate list. We do not described this modification here because it is not relevant to the problem of decoding with non-linear models parsing models studied in this chapter.

```
$C[p, l, s] = 0 \ \forall \ p, l, s$
**for** $p = 1$ to $m$ **do**
  **for** $l = 1$ to $n$ **do**
    **for** $s = 1$ to $n - l + 1$ **do**
      $w_{loc} = 0$, $t = 0$
      **while** $t \leq N[l, s]$ and $t \leq p$ **do**
        $w_{loc} := w_{loc} + w_m(b[l, s, t])$
        **for** $k = 1$ to $l - 1$ **do**
          $C[p, l, s] := \max_{0 \leq u \leq p-t} (C[p, l, s], w_{loc} + C[u, k, s]$
                                $+ C[p - u - t, l - k, s + k])$
        **end for**
        $t := t + 1$
      **end while**
    **end for**
  **end for**
**end for**
**return** $C[m, n, 1]$
```

Figure 8.1: An algorithm for finding the weight of an optimal forest with m internal nodes.

We can represent a score of a dependency tree y as a linear function

$$\hat{h}(x) = \arg\max_{y} \sum_{e \in y} w(e),$$

where $w(e)$, a score for a relation e, is defined as

$$w(e) = \sum_{y' \in G(x)} P_\theta(y'|x) I_{y'}(e).$$

Consequently, the score for a tree y is decomposed into a sum of scores of individual relations. The case of linear models with feature representation decomposable into a sum over individual relations was studied in (McDonald et al., 2005). For non-projective dependency trees, decoding is equivalent to searching for a Minimum Spanning Tree in directed graphs, a problem for which $O(n^2)$ algorithms are known (Tarjan, 1977). For projective dependency trees, as pointed out in (McDonald et al., 2005), the parsing algorithm of (Eisner, 1996), with a runtime of $O(n^3)$, can be used.

8.4 Probability estimation with discriminative classifiers

As we mentioned in the introduction, n-best lists in parsing are often used to learn a discriminative classifier, called a reranker, to predict the best candidate in a list. Consequently, we might expect better performance if we used probability estimates from rerankers in (8.1), rather than using estimates from baseline models. This is trivial when a discriminative reranker defines a probabilistic model, but less so if voted models or maximal margin classifiers are used. In this section we will briefly review applicable techniques.

If an SVM is used to learn the classier, we suggest using the normalized exponential form to estimate probabilities of candidates in this list. This form can be viewed as a direct generalization of the approach proposed in (Platt, 1999), where the logistic sigmoid of the SVM output is used as the probability estimator for binary classification problems. Therefore, the appropriate form of the probability estimate for parsing and other structured classification problems is given by the normalized exponential of the SVM output, resulting in the prediction rule

$$\hat{h}(x) = \arg\min_{y} \sum_{y' \in G(x)} \exp(A \hat{w}^T \phi(y')) \Delta(y, y'), \tag{8.5}$$

where $\hat{w}$ is a decision vector learned during classifier training, $\phi(y')$ is a feature representation of the tree y' and the scalar parameter A should be tuned on a development set.

This form also agrees with the motivation of the Fisher kernel described in chapter 6. As we have shown there the Fisher kernel defines a feature space which is appropriate for estimating the discriminative probability in the candidate list in the normalized exponential form (i.e. softmax). From the motivation of the Fisher kernel it also follows that the optimal value of A should be close to $1/\hat{w}_1$.

As was suggested in chapter 6, the construction of the TOP reranking kernel is appropriate for estimation of the expression

$$\log \sum_{y' \in G(x) \setminus \{y\}} P(x, y') - \log P(x, y),$$

which is an upper bound to the logarithm of the candidate rank. Estimation of this expression is performed by a linear model $w^T \varphi_{\hat{\theta}}^{(TK)}(x, y)$. It follows that the conditional distribution can be estimated as

$$\frac{P(x, y)}{\sum_{y' \in G(x)} P(x, y')} \approx \sigma(w^{\star T} \phi_(x, y))$$

for some choice of the decision vector $w = w^{\star}$ with the first component equal to one, where σ is the logistic sigmoid. Then the appropriate form of the loss minimizing classifier is

$$\hat{h}_{TK}(x) = \arg\min_{y' \in G(x)} \sum_{y \in G(x)} \sigma(A \hat{w}^T \phi_{\hat{\theta}}^{TK}(x, y')) \Delta(y, y'),$$

where the scalar parameter A should be selected on the development set. As for the Fisher kernel, the optimal value of A should be close to $1/\hat{w}_1$.

Another type of models popular in natural language processing are voted models (Collins & Duffy, 2002; Shen & Joshi, 2003). For such models, which combine classifiers using votes, the number of votes cast for each candidate can be used to define this discriminative probability. The discriminative probability of a candidate is simply the number of votes cast for that candidate normalized across candidates. Intuitively, we can think of this method as treating the votes as a sample from the discriminative distribution.

8.5 Experimental evaluation

To perform empirical evaluations of the proposed methods, we consider the task of constituent tree parsing of the Penn Treebank Wall Street Journal corpus (Marcus et al., 1993). First, we perform experiments with SVM Struct (Tsochantaridis et al., 2004) as the learner. However, use of SVM Struct for large scale parsing experiments is computationally expensive[6], so here we use only a small portion of the available training data to perform evaluation. In the other three sets of experiments, described below, we test our best model on the standard Wall Street Journal parsing benchmark (Collins, 1999) with the Voted Perceptron (VP) algorithm (Collins & Duffy, 2002) and maximum entropy reranker (Charniak & Johnson, 2005) as the learners.

8.5.1 Experiments with SVM Struct and ISBNs

For our first set of experiments, we used the SSN neural network models, which as we have shown in chapter 3 can be viewed an approximation to ISBNs models considered in this thesis. As a discriminative reranker we use a SVM modification (Tsochantaridis et al., 2004) with the TOP Reranking Kernel (TOP) derived from the SSN probabilistic model, as was described in the previous chapter. This combination allows us to demonstrate that results of a parser based on a generative latent variable model can be further improved by using Bayes Risk minimization.

Both the neural network probabilistic model and the discriminative classifier were trained on section 0 (1,921 sentences, 40,930 words). Section 24 (1,346 sentences, 29,125 words) was used as the validation set during the neural network learning and for choosing parameters of the models. Section 23 (2,416 sentences, 54,268 words) was used for the final testing of the models. For training and testing of the kernel models, we provided a candidate list consisting of the top 20 parses found by the probabilistic model. For the testing set, selecting the candidate with an oracle results in an F_1 score of 89.1%.

We used the SVM-Struct software package (Tsochantaridis et al., 2004) to train the SVM with the TOP kernel, with slack rescaling and linear slack penalty. The loss function is defined as $\Delta(y, y') = 1 - F_1(y, y')$, where F_1 denotes F_1 measure on bracketed constituents. This loss function was used for

[6]Proposals have been made for addressing this problem (Shen & Joshi, 2003), but since this issue is orthogonal to those addressed in this chapter, we do not consider them here.

	NN	NN-Rerank	NN-Predict	TOP	TOP-Rerank	TOP-Predict
F_1	81.3	81.8	82.1	81.7	82.1	82.3
LP	81.7	82.3	82.6	82.4	82.8	83.1
LR	80.9	81.4	81.6	81.1	81.5	81.6
CM	18.3	18.3	18.7	18.2	18.6	18.9

Table 8.1: Bayes risk minimization, experiments with data-defined kernels and SVM: percentage labeled constituent recall (LR), precision (LP), F_1 measure and percentage complete match (CM) on the testing set.

rescaling the slacks in all the SVM models, as well as in the definition of the Bayes risk. Model parameters, including the parameter A in (8.6), were adjusted on the basis of validation set accuracy. As we hypothesized in section 8.4, the optimal value on the validation set appeared to be close to $1/\hat{w}_1$, confirming the motivation of the kernel.

Standard measures of parsing accuracy, plus complete match accuracy (0-1 error), on the final testing set are shown in table 8.1. As the baselines, the table includes the results of the standard SVM classifier with the TOP kernel, and the baseline probabilistic model (NN). NN-Rerank and NN-Predict are decoding methods that use probability estimates from the SSN probabilistic model, whereas TOP-Rerank and TOP-Predict are based on probability estimates from the SVM. Both NN-Rerank and TOP-Rerank are MBR reranking approaches, as in (8.2), whereas in NN-Predict and TOP-Predict full MBR decoding is performed with the dynamic programming algorithm proposed in section 8.2.

All the proposed MBR approaches show better F_1 accuracy than the MAP decoding with the corresponding model. Both full MBR decoding methods demonstrate better accuracy than the corresponding MBR reranking approaches (NN-Predict vs NN-Rerank, TOP-Predict vs TOP-Rerank). All these differences are statistically significant. It should also be noted that, surprisingly, exact match for NN-Predict and TOP-Predict is also improved, even though the F_1 loss function was optimized.

These experimental results demonstrate that the MBR decoding approaches considered in this chapter demonstrate significant improvement over the baseline MAP decoding approaches. The relative error reduction over MAP decoding approaches is larger than the error reduction that was previously achieved by reranking with data-defined kernels over the baseline NN model.

8.5.2 Experiments with Voted Perceptron and ISBNs

The above experiments with the SVM Struct demonstrate empirically the viability of our approaches. The aim of experiments on the entire WSJ is to test whether the risk minimization methods still achieve significant improvement when more accurate generative models are used, and also to show that discriminative models other than SVM can be use to construct sufficiently accurate MBR decoders. We apply the MBR-decoding to the same list of candidates considered in the experimental section of chapter 6. As estimates of candidates

	R	P	F_1
NN	88.8	89.5	89.1
TOP	89.1	90.1	89.6
TOP-Rerank	89.5	90.5	90.0
TOP-Predict	89.6	90.7	90.1

Table 8.2: Bayes risk minimization, experiments with data-defined kernels and perceptron: percentage labeled constituent recall (R), precision (P), F_1 measure on the testing set.

probabilities we use normalized counts of votes obtained by applying the Voted Perceptron algorithms with TOP Reranking Kernel, as described in chapter 6. We only note here that the candidate list has 20 candidates, and, for the testing set, selecting the candidate with an oracle results in an F_1 score of 95.4%.

The resulting accuracies of this model are presented in table 8.2, together with results of the VP with TOP Reranking kernel and MAP-decoding with the SSN probabilistic model. These two models (NN and TOP) are the same as described and evaluated in chapter 6. Both MBR approaches achieve significantly better results than the previously proposed models. Again, the relative error reduction of TOP-Predict over TOP is about the same as that of TOP over SSN. The resulting system, consisting of the generative model and the reranker with data-defined kernel, achieves results at the state-of-the-art level.

8.5.3 Experiments with Voted Perceptron and Tree Kernels

In this experiment we show that the proposed MBR approaches are general enough and work well with other models. We replicated the parse reranking experimental setup used for the evaluation of the Tree Kernel in (Collins & Duffy, 2002), where the candidate list was provided by the generative probabilistic model (Collins, 1999) (model 2). A list of on average 29 candidates was used, with an oracle F_1 score on the testing set of 95.0%. We trained the VP algorithm using the same parameters for the Tree Kernel and probability feature weighting as described in (Collins & Duffy, 2002). A publicly available efficient implementation of the Tree Kernel was utilized to speed up computations (Moschitti, 2004). As in the previous experiment, votes of the perceptron were used to define the probability estimate.

The results for our Bayes risk reranking and prediction approaches (TK-Rerank and TK-Predict, respectively), along with the standard Tree Kernel VP results (TK) (Collins & Duffy, 2002) and the probabilistic baseline (Collins, 99) (Collins, 1999) are presented in table 8.3. The Bayes risk minimization models improve in F_1 score over the standard VP results. Differences between them and the TK model are statistically significant. The achieved error reduction over the TK model is larger than the error reduction of the standard Tree Kernel VP over MAP decoding with the probabilistic model (Collins, 99), and

	LR	LP	F_{1*}	CB	0C	2C
(Collins, 99)	88.1	88.3	88.2	1.06	64.0	85.1
TK	88.6	88.9	88.7	0.99	66.5	86.3
TK-Rerank	89.0	89.5	89.2	0.91	66.6	87.4
TK-Predict	89.1	89.5	89.3	0.89	66.9	87.6

* F_1 for previous models may have rounding errors.

Table 8.3: Bayes risk minimization, experiments with the Tree kernel and perceptron: percentage labeled constituent recall (LR), precision (LP), F_1 measure, an average number of crossing brackets per sentence (CB), percentage of sentences with 0 and ≤ 2 crossing brackets (0C and 2C, respectively).

this improvement is achieved without adding any additional linguistic features, but by using only a different decoding method. It is interesting to note that the model improves in other accuracy measures as well.

8.5.4 Experiments with a Maximum Entropy Reranker

In the last set of experiments we considered the parsing model which achieves the best results on WSJ parsing task (Charniak & Johnson, 2005). It consists of a 50-best generative parser and a maximum entropy reranker. We used probability estimates given by the reranker to define the risk approximation. The experimental setup of (Charniak & Johnson, 2005) was replicated. However the maximum entropy reranker results appeared to be considerably better than the results published in (Charniak & Johnson, 2005).[7] Selecting the candidate with an oracle results in an F_1 score of 96.8%. Accuracy measure for the maximum entropy reranking (MER) and our MBR decoding approaches (ME-Predict, ME-Rerank) are presented in table 8.4.

Both MBR decoding methods improve over MAP decoding (ME) and ME-Predict improvement is statistically significant and corresponds to about 3% error reduction.[8] We would expect a larger improvement in practice, because normally a parser trained on the WSJ is applied to the data from a different domain. In this case a probabilistic model is generally more uncertain, i.e. the entropy of the distribution in the list is larger. Therefore, the difference between MAP and MBR predictions is also expected to be larger.

	LR	LP	F_1	CB	0C	2C
ME	91.0	91.8	91.4	0.73	73.3	89.7
ME-Rerank	91.2	92.1	91.6	0.68	73.7	90.9
ME-Predict	91.3	92.1	91.7	0.67	74.5	91.1

Table 8.4: Bayes risk minimization, experiments with Charniak parser: percentage labeled constituent recall (LR), precision (LP), F_1 measure, an average number of crossing brackets per sentence (CB), percentage of sentences with 0 and ≤ 2 crossing brackets (0C and 2C, respectively).

8.6 Related Work

Even though Bayes risk minimization was popular in speech recognition community (Kumar & Byrne, 2004), there has not been much research on application of Bayes risk minimization to natural language parsing or natural language processing in general, apart from the Labeled recall method of Goodman (Goodman, 1996) for binarized PCFG, which we mentioned in the introduction to this chapter.

The methods and results presented in this chapter were first published in (Titov & Henderson, 2006b; Titov & Henderson, 2006a). The parallel and independent work of Sagae and Lavie (Sagae & Lavie, 2006) considered a method somewhat similar to Bayes risk minimization but applied to parser combination. Whereas, in our case candidate trees are obtained from a single model, they use the list of parse tree, where each of these parse trees is obtained from a different parser, and a voting scheme is used to define a weighting (i.e. distribution) over these parse trees. Then they use the standard MST parsing algorithms (McDonald et al., 2005) (for dependency parsing) or the chart parsing algorithm (for constituent parsing) to obtain a combined (or 'average') parse tree. Their goal can be viewed as minimization of risk where a normalized vote is regarded as a candidate probability. Their constituent "reparsing" algorithm does not find the tree with an optimal F1 score, but instead, similar to (Goodman, 1996), minimizes recall and not precision. Unlike (Goodman, 1996) they do not use strictly binarized grammar without unary production, therefore favoring recall would seriously harm precision. They balance precision and recall by using thresholding. Prior to application of the parsing algorithm they discard all the constituents which have scores smaller than a threshold value, where the threshold can be tuned to obtain the best F_1 score on a development set. Our algorithm instead is guaranteed to find the optimal candidate and does not require tuning of any parameters.

Bayes risk minimization both for dependency and constituent parsing was

[7] The software implementation of (Charniak & Johnson, 2005) is publicly available. We used the release of May, 2006.

[8] If we use the upper bound of 97% suggested in (Charniak & Johnson, 2005), rather than 100%, then the error reduction is 5%. 97% is motivated by inter-annotator agreement for the WSJ dataset.

also considered in papers which succeeded publication of the results presented in this chapter. In (McDonald & Satta, 2007) it was shown that the Matrix-Tree theorem (Williams, 1996) can be used to compute an estimate of the probability that an edge presents in the output tree. In this work the MST dependency parsing model (McDonald & Satta, 2007) was used to define the probability model. Therefore, MBR-decoding with the MST model can be performed directly without use of candidate lists. In the PCFG models with latent annotation, use of some form of MBR-decoding is crucial (Petrov et al., 2006). As we discussed in the introduction, these models use PCFGs where the original treebank non-terminals are extended with latent annotations. However, finding the most likely tree in the original annotation is not feasible. Instead, the problem can be viewed as MBR decoding with different loss functions. As was observed in (Petrov & Klein, 2007), the approximate algorithm of Matsuzaki et al. (Matsuzaki et al., 2005) is equivalent to the Labeled recall MBR-decoding of Goodman (Goodman, 1996), but in (Matsuzaki et al., 2005) the recall is defined not over labeled constituents but rather over productions in the grammar. In (Petrov & Klein, 2007) authors examine different techniques for MBR-decoding with latent PCFGs. In (Johnson, 2007) the MBR-decoding was applied to parsing with Projective Bilexical Dependency Grammars. The paper presented reductions of these dependency grammars to CFGs, which allowed to use CFG parsing techniques to perform MBR decoding. However, the results of the MBR decoding were essentially the same as the results of the MAP decoding. This failure to improve the accuracy was attributed to the fact that the weights obtained by the online discriminative classifier, used in their experiments, do not define a probability distribution.

8.7 Summary of the Chapter

This chapter considers the application of MBR decoding approaches to the natural language parsing task. We proposed techniques for full MBR decoding on the basis of risk estimates from candidate lists in both the constituent and dependency parsing tasks. Our approaches do not place any constraints on the probabilistic models used, thus allowing it to be used with any generative or discriminative parsing method. This make the proposed approach especially appropriate for non-linear probabilistic models. The proposed decoding methods achieve significant improvement over MAP decoding.

We would expect even better results with MBR-decoding if larger n-best lists were used. Fast n-best parsing algorithm, such as algorithms of (Jimenez & Marzal, 2000; Huang & Chiang, 2005) can be used to efficiently produce candidate lists as large as 10^6 parse trees with the model of (Collins, 1999). It is also worth noting that runtime of MBR-decoding algorithms is not significantly affected by the size of candidate lists.

Chapter 9

Conclusions

In this chapter we will summarize the contribution of the thesis, discuss directions for the future work and draw some broader conclusions.

9.1 Summary of the Thesis

In this thesis we considered different aspects of construction, parsing and reranking with non-linear statistical models for natural language parsing. As we discussed in the introduction chapter, non-linearity is required for the construction of latent variable models which do not rely on complex hand-crafted features but induce them automatically.

We considered a very general class of graphical models for natural language - Incremental Sigmoid Belief Networks (ISBNs) (Titov & Henderson, 2007a; Titov & Henderson, 2007c). We formally showed that the neural network model SSN (Henderson, 2003) can be regarded as an approximation to ISBNs and constructed a more accurate variational approximation, Incremental Mean Field method (IMF). The IMF method achieved higher accuracy than the neural network approximation both in artificial and natural language parsing experiments, thus confirming that ISBNs are an appropriate model for parsing. The resulting generative model achieved results close to the best results achieved by generative models in constituent parsing.

The particularly attractive properties of ISBNs are that they do not require explicit independence assumptions and do not enforce constraints on model structure. This makes it easy to apply ISBNs to different structured prediction tasks. In addition to the ISBN for constituent parsing, which replicates the structure of the SSN model (Henderson, 2003), we constructed an ISBN model for dependency parsing. We showed that with only very limited feature engineering the model achieves state-of-the-art results on many natural languages mostly because of its ability to induce latent features. This model achieved the third result in the shared task on multilingual dependency parsing and the best result among single-model systems.

Though ISBNs achieve high parsing accuracy, all the best models for constituent parsing use a discriminative component. The candidate list provided by a generative models is reranked by a discriminative reranker (Collins & Koo, 2005; Charniak & Johnson, 2005; Henderson, 2004). However, to perform such reranking a new feature space or a kernel should be constructed. We showed how to automatically induce feature space from latent variable models. We derived a modification of the Fisher kernel (Jaakkola & Haussler, 1998) and TOP kernels (Tsuda et al., 2002a) for the reranking problem, and our best reranker achieved significant improvement on the parse-reranking task over results of the latent variable model alone.

We showed that these data-defined kernels can be used in domain adaptation. We used the available labeled data to induce a feature representation and then we trained on the labeled data for the target-domain a discriminative linear classifier operating in the induced feature space. We demonstrated that this set-up allows us to achieve significant improvement over the baseline model.

Instead of selecting the most probable parse tree it is often preferable to select trees with a minimal expected error (i.e. Bayes risk). Though Bayes risk minimization was previously considered in parsing in the context of minimization of recall on production rules of binarized PCFGs (Goodman, 1996), the algorithm proposed there is not applicable to non-linear models and minimization of this recall measure is not the best strategy even with standard PCFG-like models. We proposed an approach which does not place any restrictions on the probabilistic model used and, thus, is especially attractive for non-linear models. We showed how this approach can be applied in both dependency and constituent tree parsing with loss functions standard for these tasks. We derived a dynamic programming algorithm for Bayes risk minimization in constituent parsing and showed that previously considered decoding algorithms can be used for the dependency parsing problem. We evaluated these methods empirically on the Wall Street Journal parsing task and demonstrated significant improvement in the considered error measures.

9.2 Future Work

There are many directions we plan on investigating in future for the problem of parsing natural language with latent variable models. In this section we consider some of them.

9.2.1 More Accurate Approximations

The proposed IMF approximation to ISBN, albeit more accurate than the neural network approximation, uses very strong independence assumptions. The fact that the more accurate approximation to ISBNs leads to higher parsing accuracy encourages construction of new approximation techniques. Graphical models of ISBNs tend to be large and densely interconnected, so inference even with coarse approximations can be prohibitively expensive. Therefore, the main challenge here is the need to maintain balance between the accuracy of an approximate model and tractability.

Nevertheless, it does seem possible to construct more accurate approximations without over-complicating the inference. Some of the assumptions made in the IMF model can be relaxed without significant increase of computational expenses. For example, instead of a fully factorisable distribution of latent variables, we could consider a mixture of factorisable distributions. This approach was previously shown to lead to significant improvement in accuracy for static graphical models (Jaakkola & Jordan, 1997). Another approach would be to explore a modification of the Gibbs sampling method (Geman & Geman, 1984), which, similarly to the IMF method, resamples variables only in the recent part of the model, whereas distributions of the previous variables are modeled using estimated means.

9.2.2 Data-Defined Kernels

There are several problems which can be addressed in studying data-defined kernels for parse reranking. The first problem is that parameters corresponding to different parts of the model should receive different weighting, and this weighting should be motivated by the approximation of the inverse information matrix. We have not attempted this, but it should be possible to find a diagonal approximation to the inverse information matrix where the diagonal elements are different for different types of parameters. We hypothesize that this is the problem preventing us from getting any additional improvement by using parameters other than parameters of the distribution of output variables in ISBN.

9.2.3 Domain Adaptation

Data-defined kernels allow us to perform reparametrization of models before the induction of the feature space. It would be interesting to exploit possibilities for inducing this reparametrization to retrain only the components of the model which need to be retrained on the new domain.

9.2.4 Beyond Natural Language

In this thesis we considered application of non-linear probabilistic models to natural language parsing, and, in particular, studied modifications of ISBN models for parsing. However, flexibility of ISBN models, i.e. the lack of constraints on the model structure, should make it possible to apply ISBNs to structured prediction problems in other domain, e.g. protein structure prediction in bioinformatics or hierarchical scene decomposition in computer vision. Therefore methods and approximations developed in this thesis have potential applicability to problems outside of natural language processing.

9.3 Conclusions

This thesis considered non-linear probabilistic models for natural language parsing and it was primarily focused on the class of models which do not impose strict constraints on the the structure of statistical dependencies. The main general goal has been to demonstrate that such latent variable models are appropriate for natural language parsing tasks and provide advantages over the use of standard 'linear' probabilistic models. We demonstrated this, first, by showing that there exist accurate approximations to these models which achieve state-of-the-art results on a variety of natural language parsing problems. Second, we showed that the powerful feature induction abilities simplify construction of parsers for new problems and domains. Also we demonstrated that the latent space induced by the model can be exploited in discriminative rerankers and that this leads to significant improvement both in the standard parsing set-up and in domain adaptation. Further, we show that the non-factorizability of these models does not prevent us from using different decoding criteria and proposing methods for Bayes risk minimization applicable to arbitrary probabilistic models for parsing.

Appendix A

An example of constituent parsing with ISBNs

In this appendix we consider an example of parsing a sentence with the ISBN model for constituent parsing described in chapter 4. We show how the structure of the ISBN graphical model changes after each parsing decision.

We consider a sequence of decisions which construct the tree in figure A.1. For convenience we distinguished nodes NP_1 and NP_2 in the tree even though their labels are the same, NP. The following sequence of figures A.2-A.13 represents the incremental construction of the ISBN model. At each position a decision vector corresponding to the current composite decision is not known and, thus, it is not shaded in the figure. After choosing a decision, the corresponding vector becomes visible and, therefore, it is shaded on the subsequent figures. Incoming arcs for the latent vector corresponding to the considered decision are labeled with relation types. Note that each arc can have several corresponding relations. As can be observed from expression (3.6), in this case the sum of components of the corresponding weight matrices is used to define CPD of the latent vector. This graphical model uses 4 types of relations between latent variables:

1. *Stack Context*: the last previous state with the same stack S.

2. *Sub-Top of S*: the last previous state where the node under the current top of the stack was on top of the stack.

3. *Left Child of Top of S*: the last previous state where the leftmost child of the current stack top was on top of the stack.

4. *Right Child of Top of S*: the last previous state where the rightmost child of the current stack top was on top of the stack.

We denote these relations in the figures as S, S_{-1}, LC, RC, respectively. For simplicity we do not show conditioning of the latent variable vectors on previous

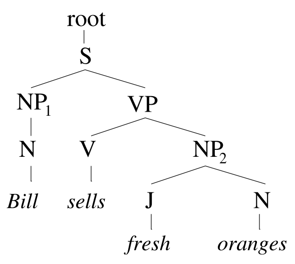


Figure A.1: An example constituent tree.

decisions, but only show conditioning on previous latent variable vectors. The caption for each figure describes states of the stack and the queue before choosing the decision. On the right in each figure we present the partial constituent structure built by the preceding sequence of decisions.

Parts of the graphical model are labeled in the figures with the index number of the step on which they were introduced. Labels of the tree nodes in the graphical model structure represent the constituents which were on top of stack S when the corresponding part of the graphical model was introduced. E.g. in figure A.13 both states 6 and 11 were introduced when VP was on top of the stack. We can view these parts as associated with the corresponding nodes in the constituent tree. Therefore, it is easy to see that the ISBN graphical model is structured similarly to the constituent tree. There are no arc in the graphical model which connect states associated with constituents distant in the tree. However, there are many arcs which connect vectors corresponding to decisions far apart in the sequence of decisions. E.g., the first and the last states (state 12) correspond to neighboring nodes in the constituent tree (*root* and S, respectively) and they are connected in the graphical model. This property allows us to argue that ISBN models exploit locality in the output structure rather than in the input structure, as it is usual for most of graphical models of structured data, e.g. HMMs.

Figure A.2: Step 1, chosen decision $\textbf{Shift}_{Bill/N}$, stack S : $[root]$, queue I : $[Bill/N, ...]$.

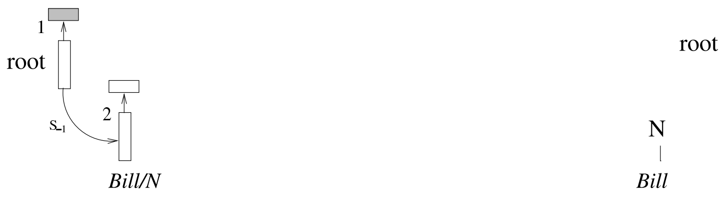


Figure A.3: Step 2, chosen decision $\textbf{Project}_{NP}$, stack $S : [root, Bill/N]$, queue $I : [sells/V, ...]$.

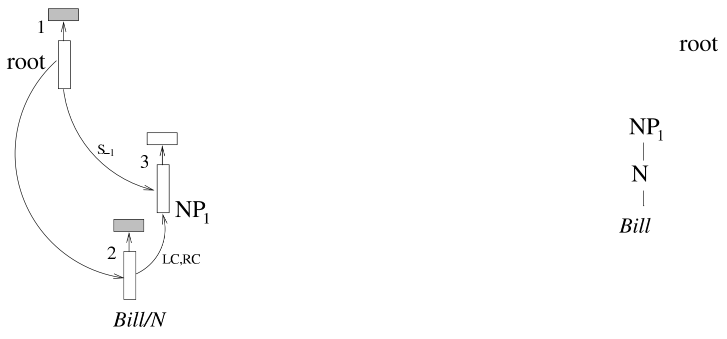


Figure A.4: Step 3, chosen decision $\textbf{Project}_{S}$, stack $S : [root, NP]$, queue $I : [sells/V, ...]$.

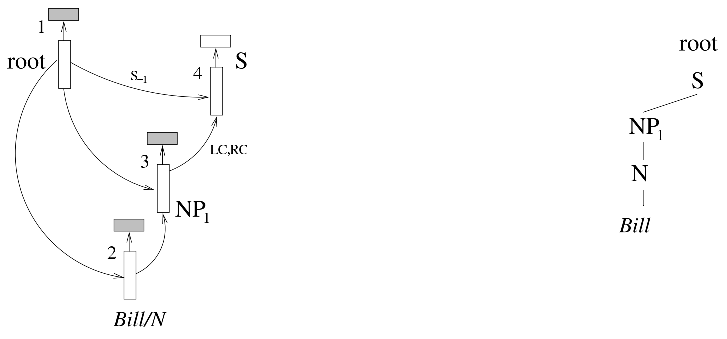


Figure A.5: Step 4, chosen decision $\textbf{Shift}_{sells/V}$, stack $S : [root, S]$, queue $I : [sells/V, ...]$.

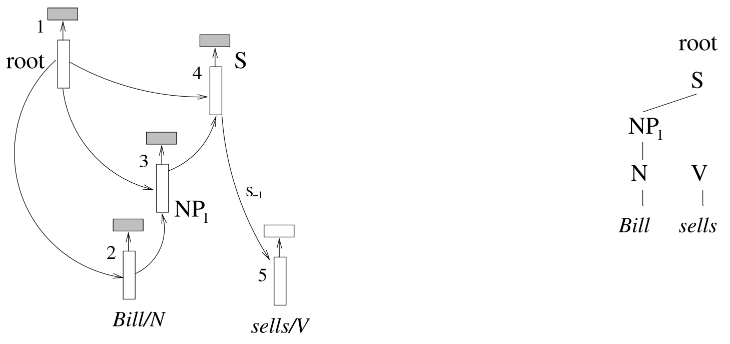


Figure A.6: Step 5, chosen decision $\textbf{Project}_{VP}$, stack $S : [root, S, sells/V]$, queue $I : [fresh/J, ...]$.

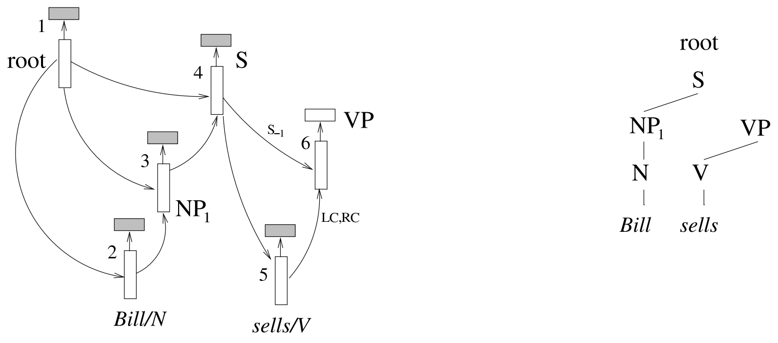


Figure A.7: Step 6, chosen decision $\textbf{Shift}_{fresh/J}$, stack $S : [root, S, VP]$, queue $I : [fresh/J, ...]$.

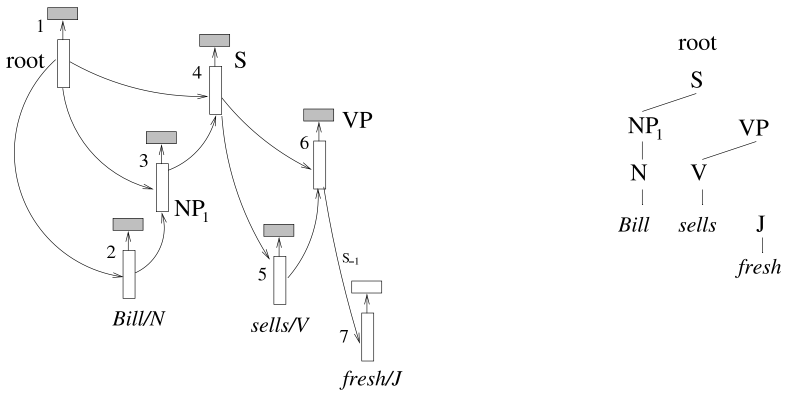


Figure A.8: Step 7, chosen decision $\textbf{Project}_{NP}$, stack $S : [root, S, VP, fresh/J]$, queue $I : [oranges/N]$.

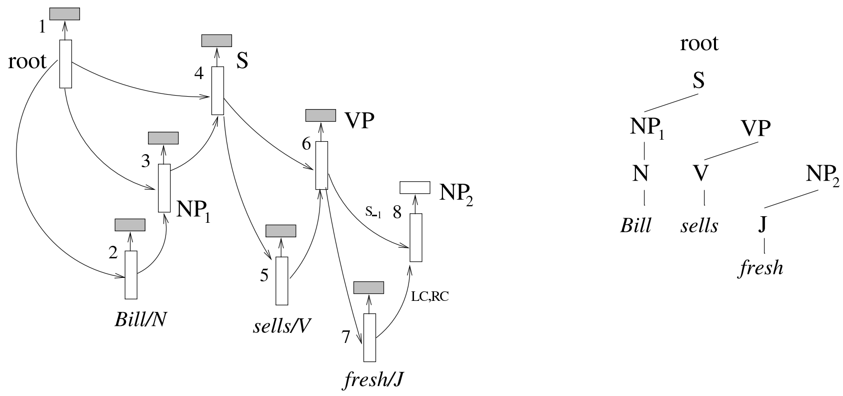


Figure A.9: Step 8, chosen decision $\textbf{Shift}_{oranges/N}$, stack $S : [root, S, VP, NP_2]$, queue $I : [oranges/N]$.

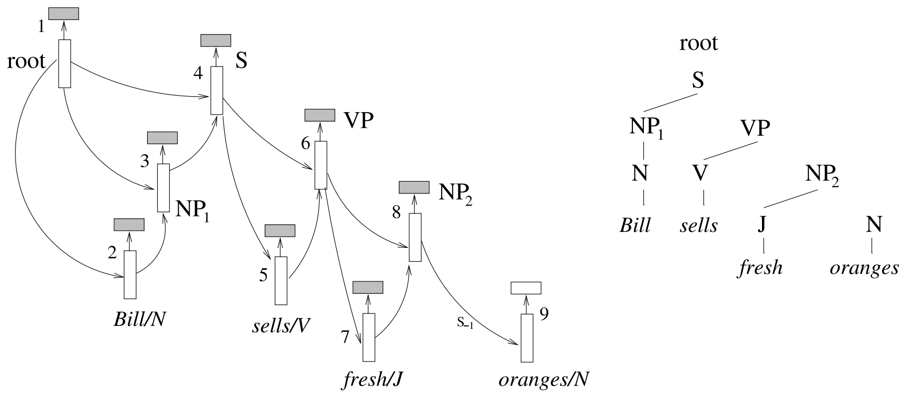


Figure A.10: Step 9, chosen decision **Attach**, stack S : $[root, S, VP, NP_2, oranges/N]$, queue $I : [\,]$.

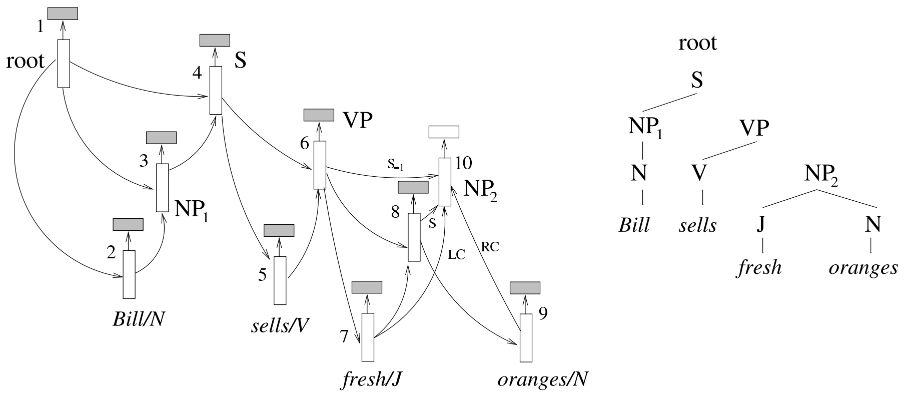


Figure A.11: Step 10, chosen decision **Attach**, stack S : $[root, S, VP, NP_2]$, queue I : [ ].

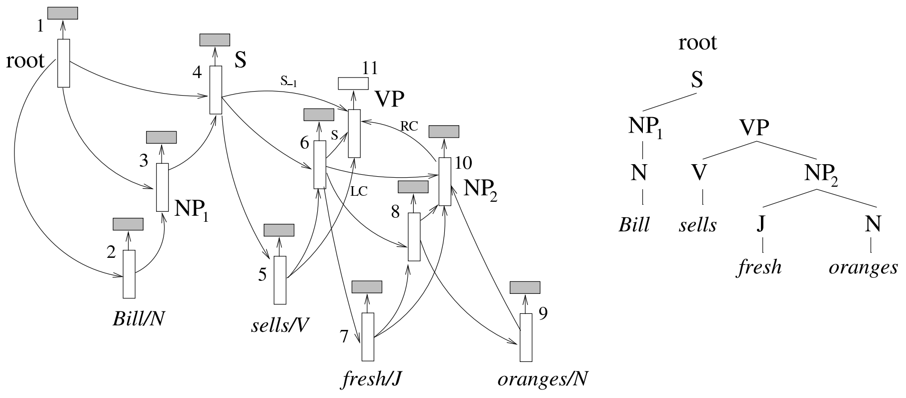
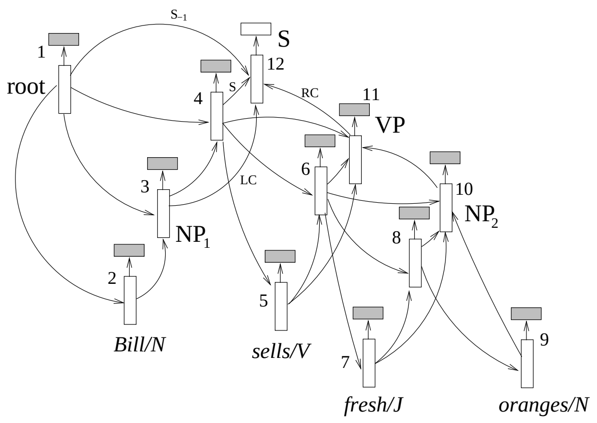


Figure A.12: Step 11, chosen decision **Attach**, stack S : $[root, S, VP]$, queue I : [ ].



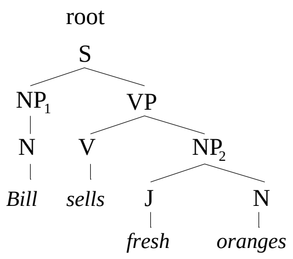


Figure A.13: Step 12, chosen decision **Attach**, stack $S : [root, S]$, queue $I : [\,]$.

Appendix B

An example of dependency parsing with ISBNs

In this appendix we consider evolution of the ISBN structure when constructing a dependency tree. We focus on the ISBN model for dependency parsing proposed in chapter 5.

We consider construction of a tree presented in fig 5.1. The corresponding sequence of parser decisions is also given in the figure. The following sequence of figures B.1-B.10 represents the incremental construction of the ISBN model. As in the previous appendix, at each position a decision vector corresponding to the current composite decision is hidden and, thus, it is not shaded in the figure. After choosing a decision, the corresponding vector becomes visible and, therefore, shaded. As previously, incoming arcs are labeled with relation types. This graphical model uses 7 types of relations defined in chapter 5, see section 5.2 for details:

1. *Input Context* ($I = I$): the last previous state with the same queue I.
2. *Stack Context* ($S = S$): the last previous state with the same stack S.
3. *Right Child of Top of S* (RCS): the last previous state where the rightmost right child of the current stack top was on top of the stack.
4. *Left Child of Top of S* (LCS): the last previous state where the leftmost left child of the current stack top was on top of the stack.
5. *Left Child of Front of I* (LCI): the last previous state where the leftmost child of the front element of I was on top of the stack.
6. *Head of Top* (HS): the last previous state where the head word of the current stack top was on top of the stack.
7. *Top of S at Front of I* ($Q = I$): the last previous state where the current stack top was at the front of the queue.

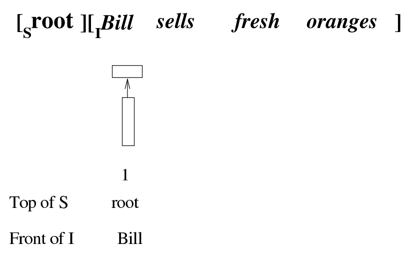


Figure B.1: Step 1, chosen decision $\textbf{Shift}_{Bill}$.

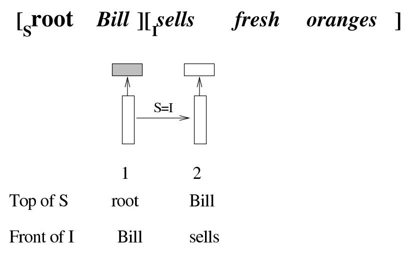
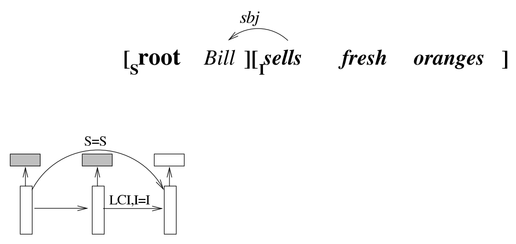


Figure B.2: Step 2, chosen decision $\textbf{Left-Arc}_{sbj}$.

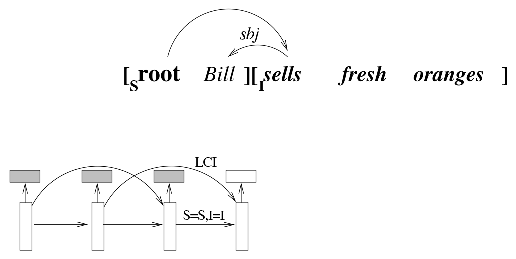
In the figures, above the structure of the ISBN we present the partial dependency structure built by the preceding sequence of decisions. Also in this structure we marked the current position of the stack ($[_S\ ]$) and the queue ($[_I\ ]$). Highlighted words are the words still present in the stack or the queue, the words without highlighting were popped from the stack before the current decision. The caption for each figure indicates the decision selected on the corresponding step.

Below the figure for each constructed latent vector of the ISBN we show the element in front of the queue and on top of the stack at time of the latent vector construction. This information is helpful for understanding placement of new arcs in the model. Also this makes it easy to see that the arcs, as in the case of the constituent model, are local in the output structure, though not necessary local in the decisions sequence. E.g. the latent vector for decision 2 is connected to the last decision vector 10 because the token on top of the stack at time step 2 (word *Bill*) is the leftmost left child of word *sells*, which was on top of the stack at time step 10.



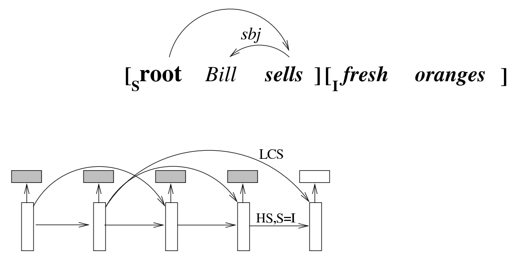
	1	2	3
Top of S	root	Bill	root
Front of I	Bill	sells	sells

Figure B.3: Step 3, chosen decision **Right-Arc**$_{-}$.



	1	2	3	4
Top of S	root	Bill	root	root
Front of I	Bill	sells	sells	sells

Figure B.4: Step 4, chosen decision **Shift**$_{sells}$.



	1	2	3	4	5
Top of S	root	Bill	root	root	sells
Front of I	Bill	sells	sells	sells	fresh

Figure B.5: Step 5, chosen decision **Shift**$_{fresh}$.

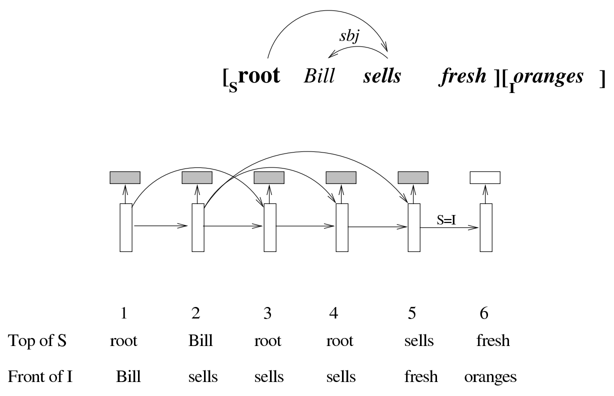

sbj
[S root Bill sells fresh][I oranges]
S=I
1 2 3 4 5 6
Top of S root Bill root root sells fresh
Front of I Bill sells sells sells fresh oranges

Figure B.6: Step 6, chosen decision **Left-Arc**$_{nmod}$.

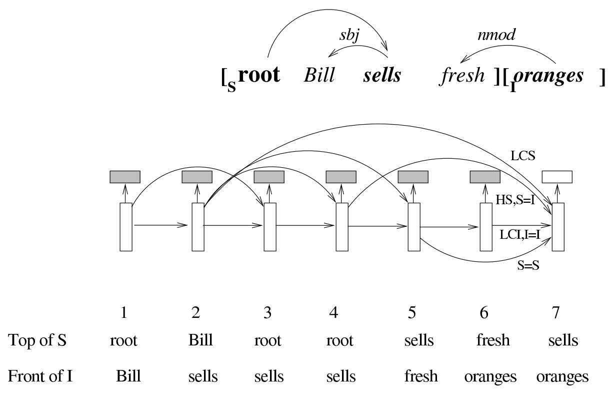
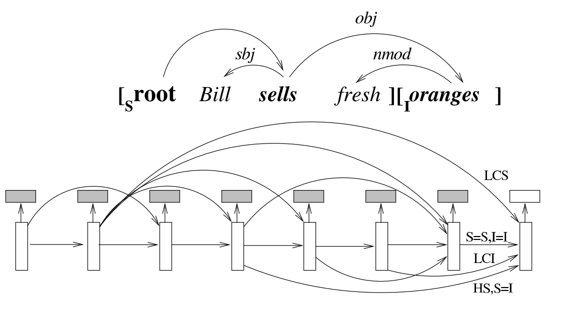
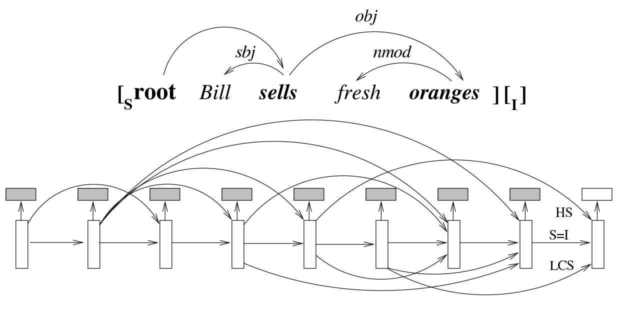

sbj
nmod
[S root Bill sells fresh][I oranges]
LCS
HS,S=I
LCI,I=I
S=S
1 2 3 4 5 6 7
Top of S root Bill root root sells fresh sells
Front of I Bill sells sells sells fresh oranges oranges

Figure B.7: Step 7, chosen decision **Right-Arc**$_{obj}$.



	1	2	3	4	5	6	7	8
Top of S	root	Bill	root	root	sells	fresh	sells	sells
Front of I	Bill	sells	sells	sells	fresh	oranges	oranges	oranges

Figure B.8: Step 8, chosen decision **Shift**$_{oranges}$.



	1	2	3	4	5	6	7	8	9
Top of S	root	Bill	root	root	sells	fresh	sells	sells	oranges
Front of I	Bill	sells	sells	sells	fresh	oranges	oranges	oranges	-

Figure B.9: Step 9, chosen decision **Reduce**.

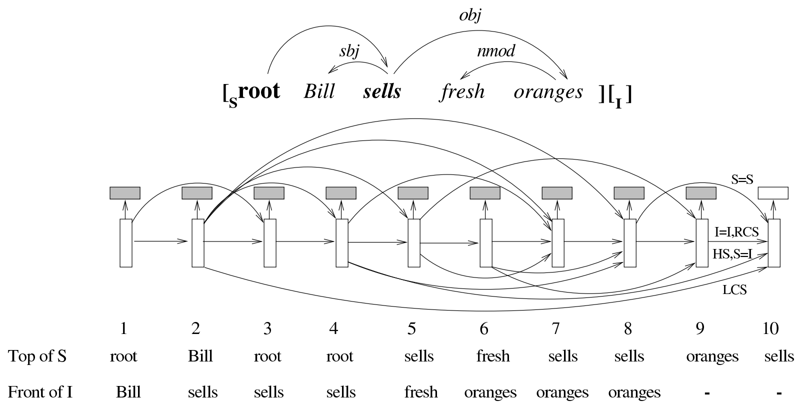

obj
sbj
nmod
[S root Bill sells fresh oranges][I]
S=S
I=I,RCS
HS,S=I
LCS
1 2 3 4 5 6 7 8 9 10
Top of S root Bill root root sells fresh sells sells oranges sells
Front of I Bill sells sells sells fresh oranges oranges oranges - -

Figure B.10: Step 10, chosen decision **Reduce**.

Bibliography

Abeillé, A. (Ed.). (2003). *Treebanks: Building and using parsed corpora.* Kluwer.

Aduriz, I., Aranzabe, M. J., Arriola, J. M., Atutxa, A., de Ilarraza, A. D., Garmendia, A., & Oronoz, M. (2003). Construction of a Basque dependency treebank. *Proc. of the 2nd Workshop on Treebanks and Linguistic Theories (TLT)* (pp. 201–204).

Aho, A. V., Sethi, R., & Ullman, J. D. (1986). *Compilers: Principles, techniques and tools.* Addison Wesley.

Amari, S. (1985). *Differential-geometrical methods in statistics.* New York: Springer-Verlag.

Amari, S. (1998). Natural gradient works efficiently in learning. *Neural Computation*, *10*, 251–276.

Ando, R. K., & Zhang, T. (2005). A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, *6*, 1817–1853.

Attardi, G. (2006). Experiments with a multilanguage non-projective dependency parser. *Proc. 10th Conference on Computational Natural Language Learning (CoNLL-X)* (pp. 166–170). New York, USA.

Bartlett, P. L. (1997). For valid generalization the size of the weights is more important than the size of the network. *Advances in Neural Information Processing Systems 10* (pp. 134–140).

Berger, A., Pietra, S. D., & Pietra, M. D. (1996). A maximum entropy approach to natural language. *Computational Linguistics*, *22*.

Bikel, D. M. (2004). Intricacies of Collins' parsing model. *Computational Linguistics*, *30*.

Bishop, C. M. (1995). *Neural networks for pattern recognition.* Oxford, UK: Oxford University Press.

Black, E., Abney, S., Flickinger, D., Gdaniec, C., Grishman, R., Harrison, P., Hindle, D., Ingria, R., Jelinek, F., Klavans, J., Liberman, M., Marcus, M., Roukos, S., Santorini, B., & Strzalkowski, T. (1991). A procedure for quantitatively comparing the syntactic coverage of english grammars. *Proc. of Speech and Natural Language Workshop* (pp. 306–311). Pacific Grove, CA.

Black, E., Lafferty, J., & Roukos, S. (1992). Development and evaluation of a broad-coverage probabilistic grammar of english language computer manuals. *Proc. 30th Meeting of Association for Computational Linguistics* (pp. 185–192).

Blitzer, J., McDonal, R., & Pereira, F. (2006). Domain adaptation with structural correspondence learning. *Proc. Conference on Empirical Methods in Natural Language Processing.* Sydney, Australia.

Blum, A., & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. *COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers* (pp. 209–214).

Bod, R. (2003). An efficient implementation of a new DOP model. *Proc. 10th Conf. of European Chapter of the Association for Computational Linguistics.* Budapest, Hungary.

Böhmová, A., Hajič, J., Hajičová, E., & Hladká, B. (2003). The PDT: a 3-level annotation scenario. In (Abeillé, 2003), chapter 7, 103–127.

Booth, T. L. (1969). Probabilistic representation of formal languages. *Proc. 10th Annual Symposium on Switching and Automata Theory* (pp. 74–81).

Bottou, L. (1991). *Une approche théoretique de l'apprentissage connexionniste: Applications à la reconnaissance de la parole.* Doctoral dissertation, Université de Paris XI, Paris, France.

Brown, P., Pietra, V. D., deSouza, P., Lai, J. C., & Mercer, R. L. (1992). Class-based n-gram models for natural language. *Computational Linguistics*, *18*, 467–479.

Buchholz, S., & Marsi, E. (2006). CoNLL-X shared task on multilingual dependency parsing. *Proc. of the Tenth Conference on Computational Natural Language Learning.* New York, USA.

Carreras, X. (2007). Experiments with a higher-order projective dependency parser. *Proc. of the CoNLL 2007 Shared Task. EMNLP-CoNLL* (pp. 957–961). Prague, Czech Republic.

Charniak, E. (1997). Statistical parsing with a context-free grammar and word statistics. *Proc. 14th National Conference on Artificial Intelligence.* Providence, RI: AAAI Press/MIT Press.

Charniak, E. (2000). A maximum-entropy-inspired parser. *Proc. 1st Meeting of North American Chapter of Association for Computational Linguistics* (pp. 132–139). Seattle, Washington.

Charniak, E., & Johnson, M. (2005). Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. *Proc. 43rd Meeting of Association for Computational Linguistics* (pp. 173–180). Ann Arbor, MI.

Chelba, C., & Jelinek, F. (2000). Structured language modeling. *Computer Speech and Language*, *14*, 283–332.

Chen, F.-Y., Tsai, P.-F., Chen, K.-J., & Hunag, C.-R. (1999). Construction of sinica treebank. *Computational Linguistics and Chinese Language Processing*, *4*, 87–104.

Chen, K., Luo, C., Chang, M., Chen, F., Chen, C., Huang, C., & Gao, Z. (2003). Sinica treebank: Design criteria, representational issues and implementation. In (Abeillé, 2003), chapter 13, 231–248.

Cocke, J., & Schwartz, J. T. (1970). *Programming languages and their compilers: Preliminary notes* (Technical Report). Courant Institute of Mathematical Sciences, New York University.

Cohn, H., Kleinberg, R., Szegedy, B., & Umans, C. (2005). Group-theoretic algorithms for matrix multiplication. *Proceedings of the 46th Annual Symposium on Foundations of Computer Science*. Pittsburgh, PA, USA.

Collins, M. (1997). Three generative, lexicalized models for statistical parsing. *Proc. 35th Meeting of Association for Computational Linguistics and 8th Conf. of European Chapter of Association for Computational Linguistics* (pp. 16–23). Somerset, New Jersey.

Collins, M. (1999). *Head-driven statistical models for natural language parsing*. Doctoral dissertation, University of Pennsylvania, Philadelphia, PA.

Collins, M. (2000). Discriminative reranking for natural language parsing. *Proc. 17th Int. Conf. on Machine Learning* (pp. 175–182). Stanford, CA.

Collins, M., & Duffy, N. (2002). New ranking algorithms for parsing and tagging: Kernels over discrete structures and the voted perceptron. *Proc. 40th Meeting of Association for Computational Linguistics* (pp. 263–270). Philadelphia, PA.

Collins, M., & Koo, T. (2005). Discriminative reranking for natural language parsing. *Computational Linguistics*, *31*, 25–69.

Collins, M., & Roark, B. (2004). Incremental parsing with the perceptron algorithm. *Proc. 42th Meeting of Association for Computational Linguistics*. Barcelona, Spain.

Csendes, D., Csirik, J., Gyimóthy, T., & Kocsor, A. (2005). *The Szeged Treebank*. Springer.

Cui, H., Sun., R., Li, K., Kan, M.-Y., & Chua, T.-S. (2005). Question answering passage retrieval using dependency relations. *Proceedings of the 28th ACM-SIGIR Conference* (pp. 400–407).

Daelemans, W., & van den Bosch, A. (2005). *Memory-based language processing.* Cambridge, UK: Cambridge University Press.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithms. *Journal of the Royal Statistical Society. Series B (Methodological)*, *39*, 1–38.

DeNeefe, S., Knight, K., Wang, W., & Marcu, D. (2007). What can syntax-based MT learn from phrase-based MT? *Proc. 45th Meeting of Association for Computational Linguistics (ACL).* Prague, Czech Republic.

Dowding, J., Gawron, J. M., Appelt, D., Bear, J., Cherny, L., Moore, R., & Moran, D. (1993). Gemini: A natural language system for spoken-language understanding. *Proc. 31st Meeting of Association for Computational Linguistics (ACL)* (pp. 340–345). Columbus, Ohio.

Dredze, M., Blitzer, J., Talukdar, P. P., Ganchev, K., Graca, J., & Pereira, F. (2007). Frustratingly hard domain adaptation for dependency parsing. *Proc. of the CoNLL shared task. Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL).* Prague, Czech Republic.

Duan, X., Zhao, J., & Xu, B. (2007). Probabilistic parsing action models for multi-lingual dependency parsing. *Proc. of the CoNLL 2007 Shared Task. EMNLP-CoNLL.* Prague, Czech Republic.

Dzeroski, S., Erjavec, T., Ledinek, N., Pajas, P., Zabokrtsky, Z., & Zele, A. (2006). Towards a Slovene dependency treebank. *Proc. Int. Conf. on Language Resources and Evaluation (LREC).* Genoa, Italy.

Eisner, J. (1996). Three new probabilistic models for dependency parsing: An exploration. *Proceedings of the 16th International Conference on Computational Linguistics.* Copenhagen, Denmark.

Eisner, J., & Satta, G. (1999). Efficient parsing for bilexical context-free grammards and head-automation grammars. *Proc. 37th Meeting of Association for Computational Linguistics* (pp. 457–464). College Park, MD, US.

Evgeniou, T., Pontil, M., & Poggio, T. (2000). Regularization networks and support vector machines. *Advances in Computational Mathematics*, *13*, 1–50.

Finkel, J. R., Grenager, T., & Manning, C. D. (2007). The infinite tree. *Proc. 45th Meeting of Association for Computational Linguistics (ACL).* Prague, Czech Republic.

Finkel, J. R., Manning, C. D., & Ng, A. Y. (2006). Solving the problem of cascading errors: Approximate bayesian inference for linguistic annotation pipelines. *Proc. Conference on Empirical Methods in Natural Language Processing*. Sydney, Australia.

Francis, W. N., & Kucera, H. (1982). *Frequency analysis of english usage: lexicon and grammar*. Boston: Houghton Mifflin.

Freund, Y., & Schapire, R. E. (1998). Large margin classification using the perceptron algorithm. *Proc. of the 11th Annual Conf. on Computational Learning Theory* (pp. 209–217). Madisson WI.

Galley, M., Graehl, J., Knight, K., Marcu, D., DeNeefe, S., Wang, W., & Thayer, I. (2006). Scalable inference and training of context-rich syntactic translation models. *Proc. of the Annual Meeting of the ACL and the International Conference on Computational Linguistics*. Sydney, Australia.

Gärtner, T. (2003). A survey of kernels for structured data. *SIGKDD Explorations*.

Geman, S., & Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *6*, 721–741.

Ghahramani, Z., & Jordan, M. I. (1996). Factorial hidden Markov models. *Advances in Neural Information Processing Systems, NIPS* (pp. 472–478). MIT Press.

Gildea, D. (2001). Corpus variation and parser performance. *Proc. Conf. on Empirical Methods in Natural Language Processing*. Pittsburgh, PA.

Goodman, J. (1996). Parsing algorithms and metrics. *Proc. 34th Meeting of the Association for Computational Linguistics* (pp. 177–183). Santa Cruz, CA.

Goodman, J. (1998). *Parsing inside-out*. Doctoral dissertation, Harvard, Cambridge, MA.

Graff, D. (1995). North american news text corpus. *Linguistic Data Consortium (LDC95T21)*.

Hajic, J. (1998). Building a syntactically annotated corpus: The prague dependency treebank. In *Issues of valency and meaning*, 106–132. Praha, Czech Republic: Karolinum.

Hajič, J., Smrž, O., Zemánek, P., Šnaidauf, J., & Beška, E. (2004). Prague Arabic dependency treebank: Development in data and tools. *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools* (pp. 110–117).

Hall, J., Nilsson, J., Nivre, J., Eryigit, G., Megyesi, B., Nilsson, M., & Saers, M. (2007a). Single malt or blended? a study in multilingual parser optimization. *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007* (pp. 933–939).

Hall, K., Havelka, J., & Smith, D. A. (2007b). Log-linear models of non-projective trees, k-best mst parsing and tree-ranking. *In Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*. Prague, Czech Republic.

Haussler, D. (1999). *Convolution kernels on discrete structures.* (Technical Report). University of Santa Cruz.

Henderson, J. (2003). Inducing history representations for broad coverage statistical parsing. *Proc. joint meeting of North American Chapter of the Association for Computational Linguistics and the Human Language Technology Conf.* (pp. 103–110). Edmonton, Canada.

Henderson, J. (2004). Discriminative training of a neural network statistical parser. *Proc. 42nd Meeting of Association for Computational Linguistics.* Barcelona, Spain.

Henderson, J., & Titov, I. (2005). Data-defined kernels for parse reranking derived from probabilistic models. *Proc. 43rd Meeting of Association for Computational Linguistics* (pp. 181–188). Ann Arbor, MI.

Hinton, G., Dayan, P., Frey, B., & Neal, R. (1995). The wake-sleep algorithm for unsupervised neural networks. *Science*, *268*, 1158–1161.

Huang, L., & Chiang, D. (2005). Better k-best parsing. *Proc. 9th Int. Workshop on Parsing Technologies.* Vancouver, Canada.

Huang, L., Knight, K., & Joshi, A. (2006). Statistical syntax-directed translation with extended domain of locality. *7th biennial conference of the Association for Machine Translation in the Americas.* Cambridge, MA.

Hudson, R. (1984). *Word grammar.* Oxford: Basil Blackwell.

Jaakkola, T., & Jordan, M. (1997). Improving the mean field approximation via the use of mixture distributions. *Proceedings of the NATO ASI on Learning in Graphical Models.*

Jaakkola, T. S., Diekhans, M., & Haussler, D. (2000). A discriminative framework for detecting remote protein homologies. *Journal of Computational Biology*, *7*, 95–114.

Jaakkola, T. S., & Haussler, D. (1998). Exploiting generative models in discriminative classifiers. *Advances in Neural Information Processes Systems 11.*

Jijkoun, V., & de Rijken, M. (2004). Information extraction for question answering: Improving recall through syntactic patterns. *Proceedings of the 20th International Conference on Computational Linguistics (COLING).* Geneva, Switzerland.

Jimenez, V., & Marzal, A. (2000). Computation of the n best parse trees for weighted and stochastic context-free grammars. *Proc. of the Joint IAPR International Workshop on Advances in Pattern Recognition.* Alicante, Spain.

Johansson, R., & Nugues, P. (2007). Extended constituent-to-dependency conversion for English. *Proc. of the 16th Nordic Conference on Computational Linguistics (NODALIDA).*

Johnson, M. (1998). PCFG models of linguistic tree representations. *Computational Linguistics*, *24*, 613–632.

Johnson, M. (2007). Transforming projective bilexical dependency grammars into efficiently-parsable cfgs with unfold-fold. *Proc. 45th Meeting of Association for Computational Linguistics (ACL)* (pp. 168–175). Prague, Czech Republic.

Johnson, M., Geman, S., Canon, S., Chi, Z., & Riezler, S. (1999). Stochastic "unification-based" grammars. *Proc. 37th Meeting of Association for Computational Linguistics.*

Jordan, M. I., Z.Ghahramani, Jaakkola, T. S., & Saul., L. K. (1999). An introduction to variational methods for graphical models. In M. I. Jordan (Ed.), *Learning in graphical models.* Cambridge, MA: MIT Press.

Joshi, A. K., & Schabes, Y. (1992). Tree-adjoining grammars and lexicalized grammars. In *Tree automata and languages*, 409–432. North-Holland.

Kashima, H., Tsuda, K., & Inokuchi, A. (2003). Marginalized kernels between labeled graphs. *Proc. International Conference on Machine Learning.* Washington, DC.

Katz, B., & Lin, J. (2003). Selectively using relations to improve precision in question answering. *Proceedings of the EACL-2003 Workshop on the Natural Language Processing for Question Answering.* Budapest, Hungary.

Kawahara, D., & Uchimoto, K. (2008). Learning reliability of parses for domain adaptation of dependency parsing. *Proceedings of The Third International Joint Conference on Natural Language Processing.* Hyderabad, India.

Klein, D., & Manning, C. D. (2003). Accurate unlexicalized parsing. *Proc. of the 41st Annual Meeting of the Association for Computational Linguistics.* Sapporo, Japan.

Knight, K., & Marcu, D. (2000). Statistical-based summarization - step one: Sentence compression. *AAAI, National Conference on Artificial Intelligence.* Austin, Texas.

Koo, T., & Collins, M. (2005). Hidden-variable models for discriminative reranking. *Proc. Conf. on Empirical Methods in Natural Language Processing.* Vancouver, B.C., Canada.

Koo, T., Globerson, A., Carreras, X., & Collins, M. (2007). Structured predictcion models via the matrix-tree theorem. *Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL).* Prague, Czech Republic.

Kromann, M. T. (2003). The Danish dependency treebank and the underlying linguistic theory. *Proceedings of the 2nd Workshop on Treebanks and Linguistic Theories (TLT).* Vaxjo, Sweden.

Kumar, S., & Byrne, W. (2004). Minimum bayes-risk decoding for statistical machine translation. *Proceedings of the Human Language Technology Conference and Meeting of the North American Chapter of the Association for Computational Linguistics.* Boston, MA.

Kurihara, K., & Sato, T. (2004). An application of the variational bayesian approach to probabilistic context-free grammars. *International Joint Conference on Natural Language Processing, Workshop: Beyong Shallow Analysis.* Hainan Island, China.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proc. 18th International Conf. on Machine Learning* (pp. 282–289). Morgan Kaufmann, San Francisco, CA.

Le Roux, N., Manzagol, P.-A., & Bengio, Y. (2008). Topmoumoute online natural gradient algorithm. *Advances in Neural Information Processing Systems (NIPS).*

Lecerf, Y. (1960). Programme des conflits, modèle des conflits. *Bulletin bimestriel de l'ATALA*, *1*, 11–18.

Liang, P., Petrov, S., Jordan, M., & Klein, D. (2007). The infinite PCFG using hierarchical dirichlet processes. *Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL).* Prague, Czech Republic.

Marcus, M. P., Santorini, B., & Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, *19*, 313–330.

Martí, M. A., Taulé, M., Màrquez, L., & Bertran, M. (2007). CESS-ECE: A multilingual and multilevel annotated corpus. Available for download from: http://www.lsi.upc.edu/∼mbertran/cess-ece/.

Matsuzaki, T., Miyao, Y., & Tsujii, J. (2005). Probabilistic CFG with latent annotations. *Proceedings of the 43rd Annual Meeting of the ACL.* Ann Arbor, MI.

McClosky, D., Charniak, E., & Johnson, M. (2006a). Effective self-training for parsing. *Proceedings of the Conference on Human Language Technology and North American chapter of the Association for Computational Linguistics (HLT-NAACL 2006).* New York, USA.

McClosky, D., Charniak, E., & Johnson, M. (2006b). Reranking and self-training for parser adaptation. *Proc. of the Annual Meeting of the ACL and the International Conference on Computational Linguistics.* Sydney, Australia.

McDonald, R. (2006a). *Discriminative learning and spanning tree algorithms for dependency parsing.* Doctoral dissertation, University of Pennsylvania, Philadelphia, PA.

McDonald, R. (2006b). Discriminative sentence compression with soft syntactic constraints. *11th Conf. of European Chapter of the Association for Computational Linguistics.* Trento, Italy.

McDonald, R., Lerman, K., & Pereira, F. (2006). Multilingual dependency analysis with a two-stage discriminative parser. *Proc. of the Tenth Conference on Computational Natural Language Learning.* New York, USA.

McDonald, R., & Nivre, J. (2007). Characterizing the errors of data-driven dependency parsing models. *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL).* Prague, Czech Republic.

McDonald, R., Pereira, F., Ribarov, K., & Hajic, J. (2005). Non-projective dependency parsing using spanning tree algorithms. *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing* (pp. 523–530). Vancouver, British Columbia, Canada.

McDonald, R., & Satta, G. (2007). On the complexity of non-projective data-driven dependency parsing. *Proc. 10th Int. Conference on Parsing Technologies (IWPT).* Prague, Czech Republic.

Montemagni, S., Barsotti, F., Battista, M., Calzolari, N., Corazzari, O., Lenci, A., Zampolli, A., Fanciulli, F., Massetani, M., Raffaelli, R., Basili, R., Pazienza, M. T., Saracino, D., Zanzotto, F., Nana, N., Pianesi, F., & Delmonte, R. (2003). Building the Italian Syntactic-Semantic Treebank. In (Abeillé, 2003), chapter 11, 189–210.

Moschitti, A. (2004). A study on convolutional kernels for shallow semantic parsing. *Proc. 42nd Meeting of the Association for Computational Linguistics.* Barcelona, Spain.

Murphy, K. P. (2002). *Dynamic belief networks: Representation, inference and learning.* Doctoral dissertation, University of California, Berkeley, CA.

Nakagawa, T. (2007). Multilingual dependency parsing using global features. *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007* (pp. 952–956).

Neal, R. (1992). Connectionist learning of belief networks. *Artificial Intelligence*, *56*, 71–113.

Ng, A. Y. (2004). Feature selection, l1 vs l2 regularization, and rotational invariance. *Proc. 21st International Conference on Machine Learning.* Banff, Alberta, Canada.

Ng, A. Y., & Jordan, M. I. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in Neural Information Processing Systems 14.* Cambridge, MA: MIT Press.

Nivre, J. (2007). Data-driven dependency parsing across languages and domains: Perspective from the CoNLL 2007 shared task. *Proc. 10th Int. Conference on Parsing Technologies (IWPT).* Prague, Czech Republic.

Nivre, J., Hall, J., Kübler, S., McDonald, R., Nilsson, J., Riedel, S., & Yuret, D. (2007a). The CoNLL 2007 shared task on dependency parsing. *Proc. of the CoNLL 2007 Shared Task. Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL).*

Nivre, J., Hall, J., Kübler, S., McDonald, R., Nilsson, J., Riedel, S., & Yuret, D. (2007b). The CoNLL 2007 shared task on dependency parsing. *Proc. of the Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL).*

Nivre, J., Hall, J., & Nilsson, J. (2004). Memory-based dependency parsing. *Proc. of the Eighth Conference on Computational Natural Language Learning* (pp. 49–56). Boston, USA.

Nivre, J., Hall, J., Nilsson, J., Eryigit, G., & Marinov, S. (2006). Pseudo-projective dependency parsing with support vector machines. *Proc. of the Tenth Conference on Computational Natural Language Learning* (pp. 221–225). New York, USA.

Nivre, J., & Nilsson, J. (2005). Pseudo-projective dependency parsing. *Proc. 43rd Meeting of Association for Computational Linguistics* (pp. 99–106). Ann Arbor, MI.

Oflazer, K., Say, B., Hakkani-Tür, D. Z., & Tür, G. (2003). Building a Turkish treebank. In (Abeillé, 2003), chapter 15, 261–277.

Pavlidis, P., Furey, T., Liberto, M., Haussler, D., & Grundy, W. (2001). Promoter region-based classification of genes. *Proceedings of the Pacific Symposium on Biocomputing* (pp. 151–163).

Pereira, F., & Schabes, Y. (1992). Inside-outside reestimation from partially bracketed corpora. *Proceedings of the 30th Annual Meeting of the ACL.* Newark, Deleware, USA.

Peshkin, L., & Savova, V. (2005). Dependency parsing with dynamic Bayesian network. *AAAI, 20th National Conference on Artificial Intelligence.* Pittsburgh, Pennsylvania.

Petrov, S., Barrett, L., Thibaux, R., & Klein, D. (2006). Learning accurate, compact, and interpretable tree annotation. *Proc. of the Annual Meeting of the ACL and the International Conference on Computational Linguistics.* Sydney, Australia.

Petrov, S., & Klein, D. (2007). Improved inference for unlexicalized parsing. *Proceedings of the Conference on Human Language Technology and North American chapter of the Association for Computational Linguistics (HLT-NAACL 2007).* Rochester, USA.

Platt, J. C. (1999). Probabilistic outputs for support vector machines and comparision to regularized likelihood methods. In A. Smola, P. Bartlett, B. Scholkopf and D. Schuurmans (Eds.), *Advances in large margin classifiers*, 61–74. MIT Press.

Prescher, D. (2005). Head-driven PCFGs with latent-head statistics. *Proc. 9th Int. Workshop on Parsing Technologies.* Vancouver, Canada.

Press, W., Flannery, B., Teukolsky, S., & Vetterling, W. (1996). *Numerical recipes.* Cambridge, UK: Cambridge University Press.

Prokopidis, P., Desypri, E., Koutsombogera, M., Papageorgiou, H., & Piperidis, S. (2005). Theoretical and practical issues in the construction of a Greek dependency treebank. *Proc. of the 4th Workshop on Treebanks and Linguistic Theories (TLT)* (pp. 149–160).

Quirk, C., Menezes, A., & Cherry, C. (2005). Dependency treelet translation: Syntactically informed phrasal SMT. *Proc. 43rd Meeting of Association for Computational Linguistics.* Ann Arbor, MI.

Ratnaparkhi, A. (1996). A maximum entropy model for part-of-speech tagging. *Proc. Conf. on Empirical Methods in Natural Language Processing* (pp. 133–142). Univ. of Pennsylvania, PA.

Ratnaparkhi, A. (1999). Learning to parse natural language with maximum entropy models. *Machine Learning*, *34*, 151–175.

Reichart, R., & Rappoport, A. (2007). Self-training for enhancement and domain adaptation of statistical parsers trained on small datasets. *Proc. 45th Meeting of Association for Computational Linguistics (ACL)* (pp. 616–623). Prague, Czech Republic.

Riezler, S., King, T. H., Crouch, R., & Zaenen, A. (2003). Statistical sentence condensation using ambiguity packing and stochastic disambiguation methods for lexical-functional grammar. *Proceedings of the Human Language Technology Conference and Meeting of the North American Chapter of the Association for Computational Linguistics*. Edmionton, Canada.

Riezler, S., King, T. H., Kaplan, R. M., Crouch, R., Maxwell, J. T., & Johnson, M. (2002). Parsing the Wall Street Journal using a Lexical-Functional Grammar and discriminative estimation techniques. *Proc. 40th Meeting of Association for Computational Linguistics*. Philadelphia, PA.

Roark, B., & Bacchiani, M. (2003). Supervised and unsuperised PCFG adaptation to novel domains. *Proceedings of the Human Language Technology Conference and Meeting of the North American Chapter of the Association for Computational Linguistics*. Edmionton, Canada.

Roark, B., & Johnson, M. (1999). Efficient probabilistic top-down and left-corner parsing. *Proc. 37th Meeting of Association for Computational Linguistics* (pp. 421–428).

Rosenkrantz, D., & Lewis, P. (1970). Deterministic left corner parsing. *Proc. 11th Symposium on Switching and Automata Theory* (pp. 139–152).

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland (Eds.), *Parallel distributed processing, vol 1*, 318–362. Cambridge, MA: MIT Press.

Sagae, K., & Lavie, A. (2006). Parser combination by reparsing. *Proceedings of the Conference on Human Language Technology and North American chapter of the Association for Computational Linguistics (HLT-NAACL 2006)* (pp. 129–132). New York, USA.

Sagae, K., & Tsujii, J. (2007). Dependency parsing and domain adaptation with lr models and parser ensembles. *Proc. of the CoNLL shared task. Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Prague, Czech Republic.

Sallans, B. (2002). *Reinforcement learning for factored markov decision processes*. Doctoral dissertation, University of Toronto, Toronto, Canada.

Saul, L. K., Jaakkola, T., & Jordan, M. I. (1996). Mean field theory for sigmoid belief networks. *Journal of Artificial Intelligence Research*, *4*, 61–76.

Saul, L. K., & Jordan, M. I. (1999). A mean field learning algorithm for unsupervised neural networks. In M. I. Jordan (Ed.), *Learning in graphical models*, 541–554. Cambridge, MA: MIT Press.

Seeger, M. (2000). *Learning with labeled and unlabeled data* (Technical Report). Institute for ANC, Edinburgh, UK. See `http://www.cs.berkeley.edu/~mseeger`.

Seeger, M. (2002). Covariance kernels from Bayesian generative models. *Advances in Neural Information Processing Systems 14*.

Sekine, S. (1997). The domain dependence of parsing. *Proceedings of the Fifth Conference on Applied Natural Language Processing* (pp. 96–102). Washington D.C.

Shen, L., & Joshi, A. K. (2003). An SVM based voting algorithm with application to parse reranking. *Proc. of the 7th Conf. on Computational Natural Language Learning* (pp. 9–16). Edmonton, Canada.

Shen, L., & Joshi, A. K. (2004). Flexible margin selection for reranking with full pairwise samples. *Proc. of the 1st Int. Joint Conf. on Natural Language Processing*. Hainan Island, China.

Shen, L., & Joshi, A. K. (2005). Ranking and reranking with perceptron. *Machine Learning, Special Issue on Learning in Speech and Language Technologies*, *60*, 73–96.

Shen, L., Sarkar, A., & Joshi, A. K. (2003). Using LTAG based features in parse reranking. *Proc. of Conf. on Empirical Methods in Natural Language Processing*. Sapporo, Japan.

Shimizu, N., & Nakagawa, H. (2007). Structural correspondence learning for dependency parsing. *Proc. of the CoNLL shared task. Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Prague, Czech Republic.

Sima'an, K. (1992). Computational complexity of probabilistic disambiguation. *Grammars*, *5*, 125–151.

Smith, D. A., & Smith, N. A. (2007). Probabilistic models of nonprojective dependency trees. *Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Prague, Czech Republic.

Smith, N. A., & Eisner, J. (2005). Contrastive estimation: training log-linear models on unlabeled data. *Proc. 43rd Meeting of Association for Computational Linguistics* (pp. 354–362). Ann Arbor, MI.

Soisalon-Soininen, E., & Ukkonen, E. (1979). A method for transforming grammars into LL(k) form. *Acta Informatica*, *12*, 339–369.

Steedman, M., Sarkar, A., Osborne, M., Hwa, R., Clark, S., Hockenmaier, J., Ruhlen, P., Baker, S., & Crim, J. (2003). Boostrapping statistical parsers from small datasets. *Proceedings of Conf. of European Chapter of the Association for Computational Linguistics.*

Stolcke, A., Konig, Y., & Weintraub, M. (1997). Explicit word error minimization in n-best list rescoring. *Proc. of 5th European Conference on Speech Communication and Technology* (pp. 163–165). Rhodes, Greece.

Suzuki, J., Isozaki, H., & Maeda, E. (2004). Convolutional kernels with feature selection for natural language processing tasks. *Proc. 42nd Meeting of Association for Computational Linguistics.* Barcelona, Spain.

Tarjan, R. E. (1977). Finding optimal branchings. *Networks*, *7*, 25–35.

Taskar, B., Klein, D., Collins, M., Koller, D., & Manning, C. (2004). Max-margin parsing. *Proc. Conf. on Empirical Methods in Natural Language Processing.* Barcelona, Spain.

Tesniére, L. (1959). *Eléments de syntaxe structurale.* Paris: Editions Klincksieck.

Titov, I., & Henderson, J. (2005). Deriving kernels from MLP probability estimators for large categorization problems. *International Joint Conference on Neural Networks* (pp. 937–942). Montreal, Canada.

Titov, I., & Henderson, J. (2006a). *Bayes risk minimization in natural language parsing* (Technical Report). Department of Computer Science, University of Geneva.

Titov, I., & Henderson, J. (2006b). Loss minimization in parse reranking. *Proc. Conference on Empirical Methods in Natural Language Processing* (pp. 560–567). Sydney, Australia.

Titov, I., & Henderson, J. (2006c). Porting statistical parsers with data-defined kernels. *Proc. 10th Conference on Computational Natural Language Learning (CoNLL-X)* (pp. 6–13). New York, USA.

Titov, I., & Henderson, J. (2007a). Constituent parsing with incremental sigmoid belief networks. *Proc. 45th Meeting of Association for Computational Linguistics (ACL).* Prague, Czech Republic.

Titov, I., & Henderson, J. (2007b). Fast and robust multilingual dependency parsing with a generative latent variable model. *Proc. of the CoNLL shared task. Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL).* Prague, Czech Republic.

Titov, I., & Henderson, J. (2007c). Incremental Bayesian networks for structure prediction. *Proc. 24th International Conference on Machine Learning (ICML).* Corvallis, OR.

Titov, I., & Henderson, J. (2007d). A latent variable model for generative dependency parsing. *Proc. 10th Int. Conference on Parsing Technologies (IWPT)*. Prague, Czech Republic.

Tsochantaridis, I., Hofmann, T., Joachims, T., & Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. *Proc. 21st Int. Conf. on Machine Learning* (pp. 823–830). Banff, Alberta, Canada.

Tsuda, K., & Kawanabe, M. (2002). The leave-one-out kernel. *International Conference on Artificial Neural Networks*. Madrid, Spain.

Tsuda, K., Kawanabe, M., Ratsch, G., Sonnenburg, S., & Muller, K. (2002a). A new discriminative kernel from probabilistic models. *Neural Computation*, *14(10)*, 2397–2414.

Tsuda, K., Kin, T., & Asai, K. (2002b). Marginalized kernels for biological sequences. *Bioinformatics*, *18*, 268–275.

Turian, J., & Melamed, D. (2006). Advances in discriminative parsing. *Proc. of the Annual Meeting of the ACL and the International Conference on Computational Linguistics*. Sydney, Australia.

Turian, J., Wellington, B., & Melamed, D. (2006). Scalable discriminative learning for natural language parsing and translation. *Proc. 20th Conf. on Neural Information Processing Systems*. Vancouver, Canada.

Turner, J., & Charniak, E. (2005). Supervised and unsupervised learning for sentence compression. *Proc. 43rd Meeting of Association for Computational Linguistics*. Ann Arbor, MI.

van der Beek, L., Bouma, G., Daciuk, J., Gaustad, T., Malouf, R., van Noord, G., Prins, R., & Villada, B. (2002). The Alpino dependency treebank. *Computational Linguistic in the Netherlands (CLIN)*.

Vapnik, V. N. (1979). *Estimation of dependencies based on empirical data (in russian)*. Moscow: Nauka. English translation: New York: Springer-Verlag, 1982.

Vapnik, V. N. (1995). *The nature of statistical learning theory*. New York: Springer-Verlag.

Vapnik, V. N. (1998). *Statistical learning theory*. New York: John Wiley and Sons.

Watson, R., & Briscoe, T. (2007). Adapting the RASP system for the CoNLL07 domain-adaptation task. *Proc. of the CoNLL shared task. Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Prague, Czech Republic.

Williams, R. (1996). *Introduction to graph theory.* Longman.

Xu, P., Emami, A., & Jelinek, F. (2003). Training connectionist models for the structured language model. *Proceedings of the 2003 conference on Empirical methods in natural language processing* (pp. 160–167). Sapporo, Japan.

Yamada, H., & Matsumoto, Y. (2003). Statistical dependency analysis with support vector machines. *In proceedings of 8th International Workshop on Parsing Technologies* (pp. 195–206). Nancy, France.

Yeh, A. (2000). More accurate tests for the statistical significance of the result differences. *Proc. 17th International Conf. on Computational Linguistics* (pp. 947–953). Saarbruken, Germany.