Generative models for syntactic and semantic structure prediction using latent variables

Garg, Nikhil

UNIVERSITÉ DE GENÈVE
Département d'informatique

FACULTÉ DES SCIENCES
Dr. James Henderson
Prof. Stéphane Marchand-Maillet

Generative Models for Syntactic and Semantic Structure Prediction using Latent Variables

THÈSE

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

par

Nikhil Garg

de
l'Inde

Thèse N° 4977

GENÈVE
2015

La Faculté des sciences, sur le préavis de Monsieur S. MARCHAND-MAILLET, professeur et directeur de thèse (Département d'informatique), Monsieur J. HENDERSON, docteur et codirecteur de thèse (Département d'informatique), Monsieur A. KALOUSIS, docteur (Département d'informatique) et Madame M. LAPATA, professeure (Institute for Language, Cognition and computation, School of Informatics, University of Edinburgh, Scotland, United-Kingdom), autorise l'impression de la présente thèse, sans exprimer d'opinion sur les propositions qui y sont énoncées.

Genève, le 9 novembre 2015

Le Doyen: Jérome LACOUR

Thèse N° 4977

To my parents

Abstract

Linguistic structures have been defined to capture varying degrees of information in natural language text, for instance, from simple per word part-of-speech tags to more complicated phrases bearing semantic roles. In Natural Language Processing (NLP), it is common to use a pipeline approach, where simpler tasks or structures are used as an input to more complicated problems. This thesis explores the Semantic Role Labeling pipeline which is composed of part-of-speech tagging, syntactic parsing, semantic argument identification, and semantic role classification. In particular, we propose generative models for dependency parsing and semantic role classification, which are two important problems in NLP. These generative models use inter-connected latent variables to encode the parsing history in the first model, and role correlations in the second model.

Dependency parsing has gained a big research interest in recent years due to its structure being relatively simple and close to semantics. At the same time, machine learning models that automatically induce high-dimensional hidden features have been gaining popularity due to their expressiveness and reduced feature engineering requirements. In recent years, several such models have been proposed for NLP problems such as language modeling, parts-of-speech tagging, and constituency parsing. Temporal Restricted Boltzmann Machine (TRBM) is one such model that consists of a network of inter-connected Restricted Boltzmann Machines (RBMs). The efficient inference mechanism available for TRBMs makes them a good tool for complex parsing models. In this work, we propose a model based on TRBM for transition-based dependency parsing. Without much feature engineering, the model achieves good parsing performance comparable to the state-of-the-art models.

Semantic Role Labeling (SRL) has emerged as an important task in NLP due to its applicability in information extraction, question answering, and other NLP tasks. However, due to limited availability of labeled data, unsupervised methods have gained significant interest recently. Research in unsupervised models has benefited immensely from the intensive research in supervised models but has not been able to incorporate complex features used in supervised models, in part due to modeling and tractability difficulties. Global role correlations, such as repetition constraints and role ordering information, are features that have been shown to help the supervised models but have not been modeled well in the unsupervised models. In this work, we propose a Bayesian model for unsupervised semantic role induction that models local constituent features as well as global role correlations in a unified framework. The model outperforms a strong baseline for this task, and the results highlight the importance of including role correlations in the model. We further propose an extension to this model that connects together monolingual models in different languages with the help of latent variables to capture cross-lingual semantic information. In particular, the joint model is trained on large amounts of parallel text in addition to the monolingual data available for each language. The aligned constituents in different languages are interconnected via latent variables that encourage soft role agreement, and hence result in better parameter estimates for the model. The results of this extension show a small improvement over the base monolingual models.

Résumé

Diverses structures linguistiques ont été proposées pour capturer les différents degrés d'information contenus dans des textes en language naturel, de simples étiquetage morpho-syntaxique aux syntagmes complexes attachés à des rôles sémantiques. En traitement naturel du langage (Natural Language Processing, NLP), il est commun d'avoir recours à une approche séquentielle, où les tâches et les structures les plus simples sont utilisées comme entrées pour résoudre des problèmes plus complexes. Cette présente thèse explore l'approche séquentielle de l'étiquetage de role sémantique (Semantic Role Labeling, SRL) qui comprend tout d'abord l'étiquetage morpho-syntaxique, puis l'analyse syntaxique, l'identification des arguments sémantiques et enfin la classification de rôle sémantique. En particulier, nous proposons deux nouveaux modèles génératifs pour l'analyse de dépendance et la classification de rôle sémantique, respectivement, qui constituent deux problèmes complexes et importants en NLP. Ces modèles génératifs ont recours à des variables latents inter-dépendentes pour représenter l'historique de l'analyse syntaxique pour le permier modèle, et les corrélations entre les rôles, pour le second modèle.

Ces dernières années, l'analyse de dépendence a connu un grand gain d'intérêt, grâce à sa formulation relativement simple et proche de l'analyse sémantique. Au même moment, les modèles d'apprentissage machine capables d'induire automatiquement des caractéristiques sous-jacentes de haute dimensionalité ont crû en popularité tout en diminuant les besoins d'ingénierie. Ces modèles ont donc naturellement été appliqués aux problèmes de NLP tels que la modélisation du langage, l'étiquetage morpho-syntaxique et l'analyse grammaticale. Un exemple de tel modèle est la Temporal Restricted Boltzmann Machine (TRBM), qui consiste en un réseau de RBMs

inter-connectés. Le méchanisme d'inférence efficace existants pour les TRBMs ont font un bon outil pour les modèles complexes d'analyse syntaxique. Dans nos travaux, nous proposons un modèle basé sur un TRBM pour l'analyse de dépendence basée sur les transitions. Notre modèle obtient des performances d'analyses proche de l'état de l'art tout en réduisant le besoin de dévelopement de caractéristiques textuelles optimisées.

L'étiquetage de rôle sémantique (SRL) est apparue comme une tâche importante en NLP par sa potentielle application à l'extraction de contenu et pour les systèmes de réponse aux questions. À cause de la disponibilité limitée de données labellisées, les méthodes non supervisées récemment ont vu leur intérêt augmenter significativement. La recherche dans ces méthodes ont grandement bénéficié des recherches effectuées dans les méthodes supervisées, sans pour autent réussir à intégrer les caractéristiques textuelles avancées qui sont utilisées dans ces dernières, en partie à cause des difficultés à les modéliser et à les inférer. Les corrélations globales de rôle sémantique, telles que les contraintes de répétition et les contraintes d'ordre des rôles, sont par exemple des caractéristiques qui ont fait preuve de leurs bénéfices dans les modèles supervisés mais qui n'ont pas encore été correctement modélisées dans les modèles non supervisés. Dans nos travaux, nous proposons un modèle bayésien pour l'induction non supervisée de rôle sémantique qui modèles les caractéristiques locales de groupe ainsi que les corrélations globales de rôle dans un cadre unifié. Notre modèle s'avère plus performant qu'une alternative forte de référence pour cette tâche, ce qui met en avant l'importance d'inclure les corrélations de rôle dans le modèle. Nous proposons de plus une extension de ce modèle qui joint plusieurs modèles unilingues pour plusieurs langues à travers des variables latentes pour capturer des informations sémantiques inter-langue. Notre modèle joint est appris sur de larges données de texte multi-lingue aligné en sus de données unilingues disponibles pour chaque langue. Les groupes syntaxiques alignés entre les différentes langues sont connectés via des variables latentes qui encourage l'entente entre les rôles, ce qui résulte en des meilleures estimations pour les paramètres du modèle. Les expériences montrent que ce modèle joint obtient des résultats légèrement meilleurs que les modèles monolingues équivalents.

Acknowledgements

First of all, I would like to thank my thesis supervisor, James Henderson, whose guidance was crucial at all stages of this work. I am grateful for not just his regular project advice, but also for the freedom he allowed me to pursue my own research direction. I would like to thank the thesis committee: Stéphane Marchand-Maillet, Alexandros Kalousis, and Mirella Lapata for their valuable feedback and support on the thesis.

I am also very grateful to Paola Merlo who along with James took a leading role in the CLCL group. Our weekly meetings and reading groups created a good research environment and sparked some great discussions. The community of fellow PhD students and Post-Doc researchers also played a crucial role in shaping the research environment. I would like to express my thanks to Andrea Gesmundo, Kristina Gulordava, Tanja Samardžić, Effi Georgala, Joel Lang, Lonneke van der Plas, Jun Wang, Adam Woznica, and Phong Nguyen for countless discussions.

Finally, I would like to thank my parents for their support during my studies.

Contents

Chapter 1

Introduction

Common linguistic structures such as part-of-speech tags, syntactic parse trees, and semantic roles are often used in various Natural Language Processing (NLP) applications. These structures provide varying levels of linguistic analyses on a sentence, and are often used as components of an NLP pipeline. For instance, part-of-speech tagging serves as a pre-processing step for syntactic parsing, which in turn is a precursor for semantic role labeling (SRL). In this thesis, we explore methods for syntactic parsing and SRL, which are two of the fundamental problems in NLP.

Syntactic parsing is usually done either as constituency parsing, or as dependency parsing. Figure 1.1 shows the dependency parse tree of an English sentence. As shown in the figure, dependency parsing creates a tree structure on top of a sentence with individual words as nodes, and directed edges going from the parent node to the child nodes. The label on an edge corresponds to the linguistic relation between the parent (head) and the child (modifier).

Semantic Role Labeling (SRL) is defined as the problem of finding predicate-argument structure in a sentence. For instance, consider the following sentence:

$[_{Arg0}$ John] READ $[_{Arg1}$ the book]

The predicate (here, the verb) *read* has two arguments: *John* as *Arg0* or the person doing the reading, and *the book* as *Arg1* or the object being read. The labels *Arg0*

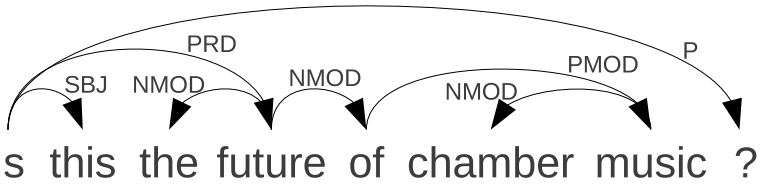


Figure 1.1: Dependency parse tree of an English sentence. Words are the tree nodes and labeled edges go from the head to the modifier of a dependency.

and *Arg1* correspond to the predicate argument relations as defined in the PropBank corpus (Palmer et al., 2005). Consider the semantic roles in another variation of the above sentence:

$[_{Arg1}$ The book] was READ by $[_{Arg0}$ John]

The SRL labeling in the two sentences remains the same even though the syntax is different. (Palmer et al., 2005) write that an SRL system should provide consistent argument labels across different syntactic realizations of the same verb.

In this work, we explore machine learning models for dependency parsing and SRL. In particular, we propose a Temporal Restricted Boltzmann Machine based model for supervised dependency parsing, and a Bayesian model for unsupervised semantic role induction. Both models are generative latent variable based models, where inter-connected latent variables provide context information for a local classification decision. The connections between latent variables encode the order of parsing decisions in dependency parsing, and encode the global role correlations in SRL.

There has been a significant interest recently in machine learning models that use high-dimensional hidden vectors that are either an abstract representation of the observations themselves, or abstract features that help in predicting the observation in generative models. In NLP, such methods have been used for language modeling (Bengio et al., 2003; Mnih and Hinton, 2007), syntactic parsing (Titov and Henderson, 2007b; Socher et al., 2011), topic modeling for documents (Salakhutdinov and Hinton, 2009), and multi-task learning (Collobert and Weston, 2008). Adding to this line of work, we demonstrate the use of Restricted Boltzmann Machines (RBMs) to capture

high-dimensional history based features in a transition-based dependency parsing model. RBMs allow us to efficiently infer both the probability of the decision given the hidden variables and the probability of the hidden variables given the decision. In addition to improved parsing performance, the better inference procedure with RBMs also enables us to achieve good performance with reduced feature engineering compared to similar previous models. This is especially helpful for large complex models used in NLP where feature engineering takes up a significant part of the modeling effort.

Unsupervised methods for SRL have gained significant traction recently due to limited amounts of annotated corpora and high cost of building such resources. SRL systems typically take the syntactic parse tree of a sentence as an input and label the constituents of that tree with semantic roles. Most of the previous work in SRL, supervised as well as unsupervised, has used local classifiers that assign a role label to each constituent independently, and modeled only limited correlations among different roles. However, some previous work in supervised SRL has shown that role repetition constraints (Punyakanok et al., 2004) and role sequence information (Toutanova et al., 2008) can help increase the accuracy of the system. In this work, we incorporate such role correlations in a Bayesian model for unsupervised semantic role induction. Our probabilistic framework allows the unsupervised learning process to identify two sets of semantic roles: *primary* and *secondary* such that the role repetition constraints and the global role sequence preferences are applied only on the primary roles, whereas a limited context information is used for the secondary roles. This modeling enables an unsupervised system to use the role correlation information that was found to be helpful in supervised systems but so far unutilized in unsupervised systems.

In recent work, multilingual learning has been explored to improve the parameter estimates in a given model using word alignments in large amounts of parallel text (Naseem et al., 2009; Titov and Klementiev, 2012a). The multilingual models incorporate some way of reducing the ambiguity in one language using the information from other languages. For instance, consider the following English sentence and its

German translation:

$EN:$ $[_{Arg0}$ John] has READ $[_{Arg1}$ the book]

$DE:$ $[_{Arg0}$ John] hat $[_{Arg1}$ das Buch] GELESEN

Given that 'book' is aligned to 'Buch', the system might be able to predict the role *Arg1* for 'Buch', even if there is insufficient information in the monolingual German data to learn this assignment. The above example illustrates that in languages where the resources are sparse or not good enough, or the distributions are not informative, SRL systems could be made more accurate by using parallel data with resource rich or more amenable languages. We extend our monolingual SRL model to multilingual learning by incorporating soft role agreement in parallel sentences whenever there is a word alignment. This is done by adding multilingual (bilingual in case of two languages) latent variables that couple the aligned constituents in parallel sentences. The mechanism for coupling the aligned constituents generalizes naturally to more than two languages, and to scenarios where there is some labeled data available for some of the languages. The multilingual model has the advantage that it can be trained jointly on parallel text, in addition to training each of its monolingual components on non-parallel text.

In the remaining part of this thesis, Chapter 2 describes our TRBM based parsing model. Chapter 3 describes our Bayesian model for unsupervised semantic role induction, and Chapter 4 describes its bilingual extension. Finally, Chapter 5 presents the conclusions of this thesis.

Chapter 2

TRBM Parsing

Syntactic Parsing is one of the basic tasks in linguistics and NLP. Syntactic parse trees are often used as a preprocessing step in various NLP tasks such as machine translation, semantic role labeling, information extraction, and others. The most common syntactic representations used in NLP are constituent parse trees (or phrase structure trees) and dependency parse trees. Figures 2.1 and 2.2 show the constituency and the dependency parse tree of a sentence respectively. As shown in the figures, a constituent parse tree groups syntactically related words together to form *constituents*, which are then recursively combined together to form higher level constituents till a top level grammar symbol S is formed. On the other hand, a dependency parse tree has individual words as tree nodes and directed edges representing head-modifier relationships. Dependency parse trees have become increasingly popular in NLP due to their structure being close to the predicate-argument semantic structure. Dependency parses are also simpler than constituency parses in the sense that no new nodes are created in a dependency structure, which makes the parsing task somewhat simpler. In this work we propose a new machine learning model for dependency parsing.

As the number of parse trees for a given sentence grows exponentially with the length of the sentence, a parsing algorithm must decompose the score of the full parse tree into a function of scores of smaller components such that the search space can be explored tractably. Further, as natural language could be ambiguous, the parser must learn from the training examples about the possible tree structures depending

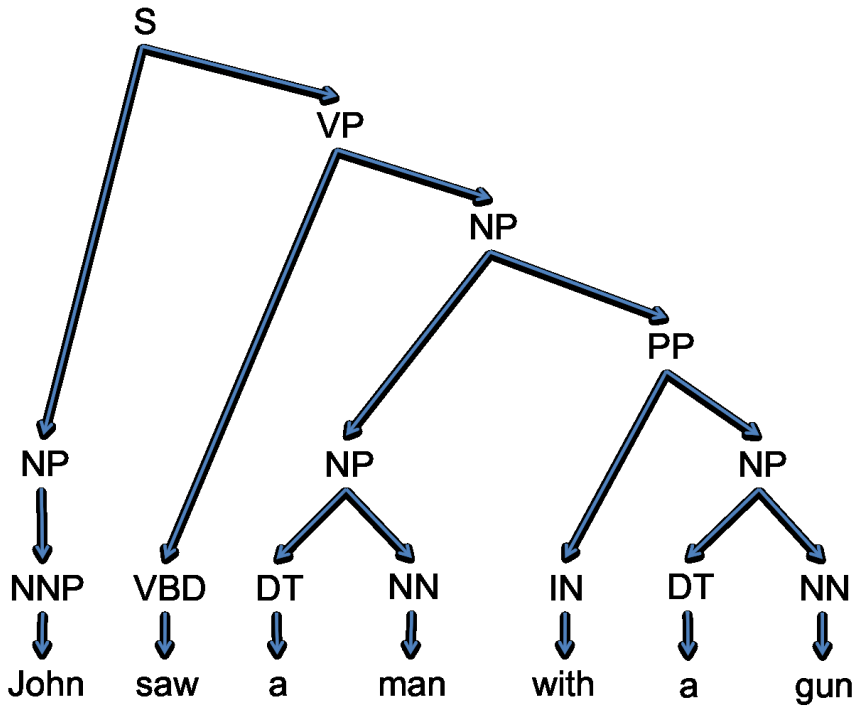


Figure 2.1: A constituency parse tree.

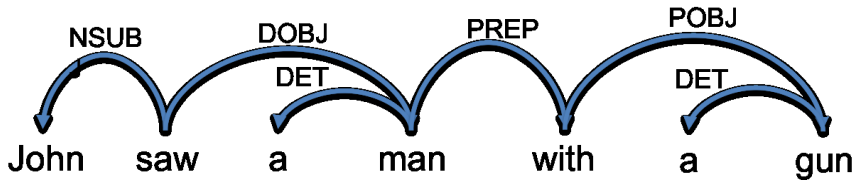


Figure 2.2: A dependency parse tree.

on the words in the sentence. Figure 2.1 illustrates a classic preposition attachment ambiguity. From this parse tree, we can deduce that John saw a man who was carrying a gun, as the phrase "a man with a gun" is labeled as a noun phrase which is a part of the verb phrase with the verb "saw". However, if we replace "gun" with "telescope", then it might become more probable that John actually used a telescope to see the man, as opposed to the likelihood that the man was carrying a telescope. We would get a different parse tree in this case. Hence, the parsing algorithm must be capable of learning such preferences that are highly dependent on the words. Data-driven parsing models that have been proposed for dependency parsing can be broadly classified into two major categories: *graph-based* models and *transition-based* models. Graph-based algorithms typically score a parse tree as the sum of local scores of small parts that are combined together to form a tree. The small parts can be combined together using different methods including a bottom up dynamic programming algorithm (Eisner, 2000), and a maximum spanning tree algorithm (McDonald et al., 2005). On the other hand, transition-based methods typically process the sentence from left to right, constructing edges along the way based on the state of the parser (Yamada and Matsumoto, 2003; Nivre and Scholz, 2004). The state of the parser encodes the parsing history of the algorithm, and conditioning of the next derivation step on the parser state decomposes the probability of the whole tree into smaller parts. Graph-based methods have the advantage that they are able to find a global optimum using a dynamic programming approach. However, the features used to score each of the sub parts must be local in their scope, which limits the parser's ability to use more complex features. In contrast, transition-based parsers can take advantage of the arbitrary complex history based features as they process the sentence sequentially. Unfortunately, the search space of such parsers is often intractable and pruning strategies must be used for efficient parsing. However, with appropriate pruning or even deterministic parsing in some cases, it is possible to achieve a good parsing performance and at the same time maintain reasonable efficiency. Both kinds of parsing approaches have been able to achieve state-of-the-art accuracies on the benchmark datasets. We use a transition-based algorithm in our model.

Transition-based parsing models consist of a series of derivation steps, and use a classifier at each step to determine the next parsing action that takes them from one state to another. Previous work has proposed *memory-based classifiers* (Nivre et al., 2004), SVMs (Nivre et al., 2006b), and Sigmoid Belief Networks (SBNs) (Titov and Henderson, 2007b) for this classification task. These classifiers use some features constructed from the derivation history to predict the next derivation step. Of these approaches, only SBNs make use of high-dimensional hidden representations to encode the parse history. (Titov and Henderson, 2007b) call them Incremental Sigmoid Belief Networks (ISBNs) as the model structure is built incrementally using directed connections between hidden variables of different derivation steps. ISBNs give excellent parsing performance but suffer from either very approximate or slow inference procedures. The advantage of their hidden representations is that they are able to automatically induce complex history features without much feature engineering. We propose to address the problem of inference in a high-dimensional hidden space by using an undirected graphical model, Restricted Boltzmann Machines (RBMs), to model the individual parsing decisions. Unlike the Sigmoid Belief Networks (SBNs) used in ISBNs, RBMs have tractable inference procedures for both forward and backward reasoning, which allows us to efficiently infer both the probability of the decision given the hidden variables and vice versa. The key structural difference between the two models is that the directed connections between hidden and decision vectors in SBNs become undirected in RBMs. A complete parsing model consists of a sequence of RBMs interlinked via directed edges, which gives us a form of Temporal Restricted Boltzmann Machines (TRBMs) (Taylor et al., 2007), but with the incrementally specified model structure required by parsing. In this work, we analyze and contrast ISBNs with TRBMs and show that the latter provide an accurate and theoretically sound model for parsing with high-dimensional hidden variables.

The rest of this chapter is organized as follows. Section 2.1 overviews the related work in this field. Section 2.2 gives a brief outline of a Shift-Reduce parsing algorithm. Next, we describe two baselines: a basic model with no hidden variables in section 2.3, and the ISBN model in section 2.4. Section 2.5 gives the details of our proposed model. We describe the experiments in section 2.6, and a closing discussion in section

2.7.

2.1 Related Work

Graph Based Parsing Models Graph-based models typically factor the dependency trees into a set of parts that can be scored locally, and then search for the global optimum. (Eisner, 1996; Eisner, 2000) introduced the idea of bottom-up dependency parsing based on the notion of spans. Their model was a first-order model in the sense that the probability of the parse tree was factored into probabilities of individual *head-modifier* dependencies. Second-order extensions to this model were proposed by (McDonald and Pereira, 2006) adding *sibling* relationships, and by (Carreras, 2007) further adding *grandchild* relationships. (Koo and Collins, 2010) extended the second-order parsing further to third-order parsing by introducing *grand-sibling* and *tri-sibling* relationships.

(McDonald et al., 2005) proposed an alternative method for first-order dependency parsing by formalizing it as a search for a maximum spanning tree (MST) in a directed graph. This approach had an advantage over bottom up parsing approach of (Eisner, 2000) in that it could perform non-projective parsing. (McDonald and Pereira, 2006) further extended this approach to second-order parsing by adding sibling relationships.

Transition Based Parsing Models Transition-based models build the parse tree incrementally by moving from one state to another, each state representing a partially built tree. (Yamada and Matsumoto, 2003) proposed the first transition based parser for English which processed the sentence from left to right, and used a sequence of *Shift*, *Left* and *Right* actions to construct the unlabeled parse tree. An SVM was used to choose the best action at any given stage using features extracted from the local context. (Nivre and Scholz, 2004) proposed a similar *Shift-Reduce* parser for English that used a *memory-based learning* algorithm, and could perform labeled dependency parsing. (Titov and Henderson, 2007c) extended this model to non-deterministic parsing by using a

generative history based model that integrated word predictions into the set of parser actions, and used a beam search for decoding.

Neural/Deep Networks in NLP Although less common than other modeling methods, neural networks and other deep networks have been used in a few works for different NLP tasks. (Bengio et al., 2003) proposed a neural network model for language modeling. Each word was given a distributed vector representation, and the vector representations of the context words served as an input to a neural network whose output was a high dimensional vector giving the probability for each word. The vector representations for the words and the neural network weights were both learned simultaneously. Following this approach, quite recently, a number of neural network based models have been proposed to induce distributed representations of words or phrases (Huang et al., 2012; Mikolov et al., 2013a; Mikolov et al., 2013b). (Collobert and Weston, 2008) proposed a deep neural network architecture for multi-task learning. Each word was first mapped to a vector representation, and the word representations of a sentence were fed into the neural network of a task such as language modeling, POS tagging, SRL, etc. Every task had its own specific neural network but the word representations were shared, which enabled the model to learn feature representations that were more general compared to task specific representations. (Lane and Henderson, 2001; Henderson, 2003) proposed to use *Simple Synchrony Networks* (SSNs), a neural network based model, for constituency parsing. The model gave close to state-of-the-art performance but there was no principled interpretation of the hidden representations that were encoding the parsing derivation history. (Titov and Henderson, 2007a) proposed a constituent parsing model based on *Incremental Sigmoid Belief Networks* (ISBN), a dynamic Bayesian network that was approximated as a feed-forward neural network. They further showed that the SSN model can be viewed as a coarse approximate to inference with ISBNs. The ISBN model is closely related to our work and we describe it in section 2.4. ((Socher et al., 2011), (Socher et

al., 2013)) proposed a Recursive Neural Network based model called *Compositional Vector Grammars* for constituent parsing. This model represented the constituents or phrases as a distributed feature vector. The training process learned weight vectors that combined two input feature vectors corresponding to two child constituents, to predict the parent feature vector representing the parent constituent. A separate weight vector predicted the label of the constituent given its feature vector. Recently, neural networks have also been used for word alignment (e.g. (Tamura et al., 2014)), and for trained bilingual word embeddings (or distributed representations) for phrase based machine translation (e.g. (Zou et al., 2013)).

RBM Based Models An RBM is a hidden variable model that offers simple learning and inference procedures, and serves as a building block for learning in higher order models such as Deep Belief Networks (DBNs), as well as a building block for constructing sequential models such as TRBMs. RBM based models have been used with great success in image and video processing tasks. (Hinton et al., 2006) proposed a deep belief network for recognition of hand-written digits. The top two layers of this network were an RBM, and the bottom layer was the observed pixel data. The rest of the layers had directed connections in both upwards and downwards directions, for inference and generation respectively. (Taylor et al., 2007) proposed a *Conditional RBM* (or *Temporal RBM*) model for modeling motion capture data. This was a directed time series model which had an RBM at each time step to model the observed motion data as the visible layer, and the internal state of the model as the hidden layer. The directed connections from some previous visible layers to the current hidden and visible layers provided the context information for inference.

Despite their success, RBMs have seen limited use in the NLP community. (Salakhutdinov and Hinton, 2009) proposed an undirected topic model called *replicated softmax* for modeling topics of text documents. The model was essentially an RBM with the document words as the visible layer, and a hidden layer representing a distributed topic representation. (Mnih and Hinton, 2007)

proposed *Temporal Factored RBM* for language modeling where each time step had an RBM corresponding to the current word, and directed connections from the previous to the current time step provided context information. The authors argued that the hidden to hidden connections in this model could incorporate dependencies longer than n-grams.

2.2 Shift-Reduce Dependency Parsing

(Nivre et al., 2004) describe a transition-based model for projective dependency parsing. At any given time step, the parser state consists of a Stack S, a queue I, and a partially constructed labeled dependency structure. Initially, S is empty and I consists of all the words in the sentence. At any given state, let the top of S be word w_i and the front of I be word w_j. The parser has four kinds of transitions:

1. ***Left-Arc*** Adds an edge from w_j to w_i and selects an edge-label r. Pops w_i.

2. ***Right-Arc*** Adds an edge from w_i to w_j and selects an edge-label r.

3. ***Reduce*** Pops w_i.

4. ***Shift*** Shifts w_j from I to S.

Parsing is complete when both I and S are empty. Table 2.1 shows the sequence of transitions for an example sentence.

The models used for assigning probabilities to the parser actions are an important factor and several methods have been proposed including *memory-based classifiers* (Nivre et al., 2004), SVMs (Nivre et al., 2006b), and ISBNs (Titov and Henderson, 2007b). We follow the hidden variable approach of (Titov and Henderson, 2007b), exploring the design issues in related models, and show that TRBMs provide an accurate and theoretically sound model for parsing with hidden variables. We build up the model in three incremental steps: (i) Model 0 has no hidden variables, (ii) Model 1 introduces hidden variables in an ISBN structure, and (iii) Model 2 modifies the ISBN structure to convert it to a TRBM structure.

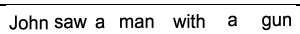
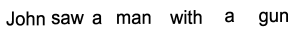
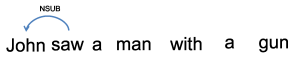
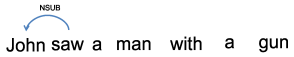
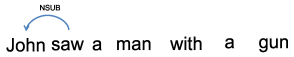
	Stack	Queue	Partial Parse Tree	Next Op
1.	⟨empty⟩	John saw a man with a gun		Shift
2.	John	saw a man with a gun		Left-Arc
3.	⟨empty⟩	saw a man with a gun		Shift
4.	saw	a man with a gun		Shift
5.	saw a	man with a gun		Left-Arc
6.	saw	man with a gun		Right-Arc
7.	saw	man with a gun		Shift
8.	saw man	with a gun		Right-Arc
9.	saw man	with a gun		Shift
10.	saw man with	a gun		Shift
11.	saw man with a	gun		Left-Arc
12.	saw man with	gun		Right-Arc
13.	saw man with	gun		Shift
14.	saw man with gun	⟨empty⟩		Reduce
15.	saw man with	⟨empty⟩		Reduce
16.	saw man	⟨empty⟩		Reduce
17.	saw	⟨empty⟩		Reduce
18.	⟨empty⟩	⟨empty⟩		-

Table 2.1: Shift-Reduce parsing transitions for an example sentence: 'John saw a man with a gun'. The table shows the stack and queue data structures, as well as the the partial parse tree built so far and the next operation (next op). The top of the stack and the front of the queue are underlined.

2.3 Model 0: No Hidden Variables

As shown in Figure 2.3, Model 0 is composed of a series of time steps (the big unshaded rectangles), each corresponding to one of the four decisions: *Left-Arc*, *Right-Arc*, *Reduce* or *Shift*. Each time step in turn consists of a sequence of related decision steps (the small shaded rectangles). The context information or *parser history* is provided by having directed connections from a selected set of past time steps to the current step. The probability of a parse tree is composed of probabilities of individual derivation steps, i.e.

$$P(\text{tree}) = \prod_t P(\mathbf{v}^t | \mathbf{v}^1, ..., \mathbf{v}^{t-1})$$

where each $\mathbf{v}^t$ is a parser decision of the type *Left-Arc*, *Right-Arc*, *Reduce* or *Shift*. Let us denote the past parser decisions $\mathbf{v}^1, ..., \mathbf{v}^{t-1}$ by $history^t$. Each decision is further decomposed into sub-decisions as follows. A *Left-Arc* time step is modeled by two decision variables: the first generates the action to create a Left-Arc with probability $P(\textit{Left-Arc}|history^t)$, and the second generates the dependency label with probability $P(\textit{Label}|\textit{Left-Arc}, history^t)$. The *Right-Arc* step is modeled in a similar way. The *Reduce* step has just one decision variable having a probability $P(\textit{Reduce}|history^t)$. The *Shift* step has three decision variables corresponding to generating the action Shift with probability $P(\textit{Shift}|history^t)$, generating the part-of-speech tag with probability $P(\textit{Part-of-Speech}|\textit{Shift}, history^t)$ and generating the word with probability $P(\textit{Word}|\textit{Shift},\textit{Part-of-Speech}, history^t)$. Modeling each decision as a one-hot binary vector, the decision probability at time step t can be written using a soft-max function:

$$p(\mathbf{v}^t|history^t) = \frac{1}{Z} exp\left(\sum_{i \in \text{visible}} a_i v_i^t + \sum_{\substack{c \in \text{connections} \\ l \in \text{visible}}} v_i^t w_{li}^{(c)} v_l^{(c)} \right)$$

where $\mathbf{v}^t$ denotes the one-hot visible vector at time step t, v_i^t is the ith variable in

that vector, and a_i denotes the weight of the bias term for v_i^t. $v_l^{(c)}$ denotes a visible variable in the past time step, and $w_{lk}^{(c)}$ denotes the weight of the corresponding connection. Z is the normalization term summing over all possible values of the current decision vector.

To capture the parse history, we use the same set of seven kinds of connections as used in (Titov and Henderson, 2007b) which take into account the *structural locality* in the partial dependency structure while connecting two time steps. The same weight matrix is used for the same kind of connections and these weight parameters are learned by back-propagating likelihood gradients. For completeness, we list the set of connections used below. Let us denote the Stack and Queue at the current time step by S_c and I_c respectively, and at a past time step by S_p and I_p respectively. We connect the closest past time step to the current time step in each of the following cases:

1. Front of I_c is the same as front of I_p
2. Top of S_c is the same as top of S_p
3. Rightmost right child of the top of S_c is the same as the top of S_p
4. Leftmost left child of the top of S_c is the same as the top of S_p
5. Head of the top of S_c is the same as the top of S_p
6. Leftmost child of I_c is the same as top of S_p
7. Top of S_c is the same as front of I_p

For Model 0, these connections give information about words, POS tags, dependency labels, etc. (Figure 2.3) of the structure in the neighborhood of the current derivation step.

Note that Figure 2.3 only shows one possible derivation path. Different parsing transitions possible at every step give rise to a huge search space illustrated in Figure 2.4. Every configuration allows a set of possible transitions, each with a certain probability. The parser is expected to search for the highest probability derivation

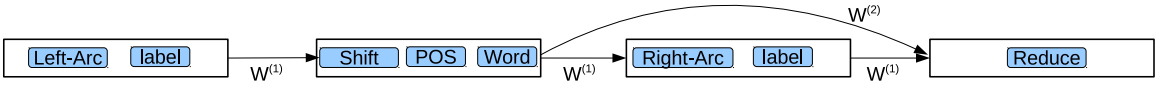


Figure 2.3: Model 0. $W^{(c)}$ denotes the weight matrix for directed connection of type c between two vectors, where $c \in [1, 7]$ corresponding to the 7 kinds of connections described in section 2.3.

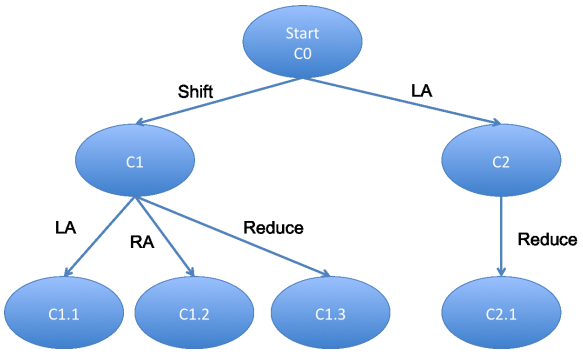


Figure 2.4: An illustration of the search space of transition-based parsing.

starting from the root of the search tree and ending at one of the leaves. As it is intractable to explore all possible derivations, we prune the search space after every shift operation by keeping only α highest probability configurations. This is known as beam search and α is known as the beam size. We used $\alpha = 20$ and increasing it further did not result in much improvement.

2.4 Model 1: Incremental Sigmoid Belief Networks (ISBN)

ISBNs are dynamic bayesian networks where the model structure is specified incrementally by the decision sequence. Figure 2.5 illustrates an ISBN network for our parsing algorithm. For dependency parsing, the ISBN could be considered as an extension to Model 0 (Section 2.3), where we add a hidden layer to every time step, with directed connections to the corresponding decision variables. Further, the temporal connections are now between two hidden layers instead of two visible layers. The advantage of having hidden to hidden connections is that hidden variables can

encode much more complex information (as hidden features) about the history and this information can propagate forward through directed temporal connections. Note that an ISBN network is built incrementally, that is, at any given time step, we only know about the graph structure of the present and the past time steps, and the future graph structure is unknown. This means that there are no visible variables in the future time steps that have to be considered while doing inference at the current step, and thus there is no need to marginalize over all possible future graph structures.

As the decision probabilities are conditioned on the history, once a decision is made the corresponding variable becomes observed, or visible. In an ISBN, the directed edges to these visible variables and the large numbers of heavily inter-connected hidden variables make exact inference of decision probabilities intractable. (Titov and Henderson, 2007a) proposed a couple of approximation procedures for inference in ISBNs. The first was a feed forward approximation where hidden variables were allowed to depend only on their parent variables. In other words, this becomes a neural network style model where current hidden variables are completely determined by the temporal connections without taking into account the current or future observations. Due to this limitation, the authors proposed to make hidden variables conditionally dependent also on a set of explicit features derived from the parsing history, specifically, the base features defined in (Nivre et al., 2006b). As shown in the experiments section, this addition results in a big improvement and works well for the parsing task. These features are:

1. Word at the top of the stack
2. Word at the front of the queue
3. Head word of the top of the stack
4. Lemma of the top of the stack
5. Lemma of the top of the queue
6. Part-of-speech tag of the top of the stack
7. Part-of-speech tag of the front of the queue

8. Part-of-speech tag of one element below the top of the stack
9. Dependency label of the top of the stack
10. Dependency label of the leftmost dependent of the top of the stack
11. Dependency label of the rightmost dependent of the top of the stack
12. Dependency label of the leftmost dependent of the front of the queue

The feed forward approximation described above could also be viewed as a slightly complex version of a Recurrent Neural Network (RNN). The RNNs used for sequential data do the same computation at every step of the sequence. For instance, in language modeling the RNN would typically compute the probability of the next word given the previous words in the sequence. Thus, the parameters used in every step are the same as opposed to a standard neural network where they would be different. Using the same parameters could also be thought of as *unfolding* the cyclic connections in a single-step RNN. The key advantage of RNNs over simple neural networks is that the RNNs maintain a *memory* of what inputs and/or outputs they have already seen in the past, and use this information to compute the output at the current step. The feed forward parsing model could be thought of as a slightly complex version of a RNN where we have different kinds of connections to the previous time steps depending of the current stack and queue. On the other hand, in a typical RNN same kind of connections are used at every time step. However, similar to a RNN, the parsing model uses the same weight matrix for the same kind of connection.

Another difference with standard RNNs is that the RNNs are able to do the computation in both directions. For instance, in language modeling, one might have two RNNs, one from left to right, and the other from right to left. The probability of the current word could then use the context from both RNNs. Unfortunately, doing this bi-directional computation is not that straight-forward in the above feed forward parsing model because of an intractable number of possible derivations (and consequently, network structures) to the right of the current step. In fact, we need to dynamically construct the network, using pruning to keep only a certain number

of most probable configurations, to maintain tractability. The changing network structures with each derivation makes it difficult to use the bidirectional computation.

The second approximate inference procedure, called the mean field approximation, extended the feed-forward approximation by updating the current time step's hidden variables after each sub-decision. Although this approximation is more accurate than the feed-forward one, there is no analytical way to maximize likelihood with respect to the means of the hidden variables, which requires an iterative numerical method and thus makes inference very slow, restricting the model to only shorter sentences.

In this work, Model 1 is based on the feed forward approximation. The probability distribution of each decision in the ISBN model is given by:

$$p(\mathbf{v}^t|\mathbf{h}^t, history^t) = \frac{1}{Z} exp\left(\sum_{i \in \text{visible}} a_i v_i^t + \sum_{\substack{i \in \text{visible} \\ j \in \text{hidden}}} v_i^t h_j^t w_{ij}\right)$$

where v_i^t and h_j^t denote the ith visible and jth hidden variable respectively at time step t. a_i is the bias term for v_i^t. Z is the normalization term that sums over all possible values of the one-hot vector $\mathbf{v}^t$. The mean value of the hidden variables is calculated as:

$$h_j^t = \sigma\left(b_j + \sum_{\substack{c \in \text{connections} \\ l \in \text{hidden}}} w_{HH_{lj}}^{(c)} h_l^{(c)}\right)$$

where b_j denotes the bias term for the variable h_j^t, $h_l^{(c)}$ denotes a hidden variable in the past time step, and $w_{HH_{lj}}^{(c)}$ denotes the weight of the corresponding connection. σ is the logistic sigmoid function: $\sigma(x) = 1/(1 + e^{-x})$.

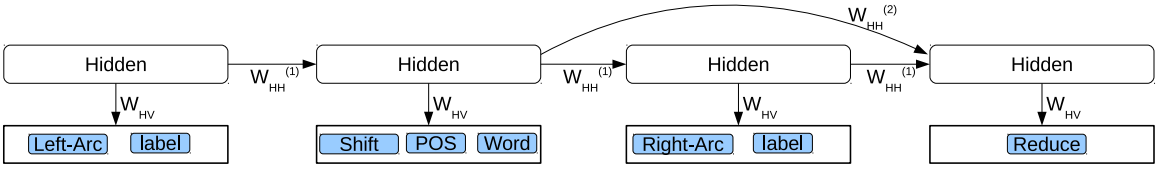


Figure 2.5: An ISBN network. Shaded nodes represent visible variables and 'Hidden' represents a vector of hidden variables. W_{HV} denotes the weight matrix for the connections between hidden and decision vectors, and $W_{HH}^{(c)}$ denotes the weight matrix for directed connection of type c between two hidden vectors, where $c \in [1, 7]$ corresponding to the 7 kinds of connections described in section 2.3.

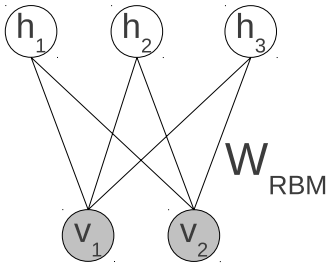


Figure 2.6: An RBM with a set of visible variables v_i and a set of hidden variables h_j. W_{RBM} is the symmetric weight matrix between hidden and visible variables.

2.5 Model 2: Temporal Restricted Boltzmann Machines (TRBM)

The proposed TRBM model is similar to the ISBN model, except that RBMs give TRBMs an analytical way to do exact inference within each time step. Although information passing between time steps is still approximated, TRBM inference is more accurate than previously proposed ISBN approximations. Before describing the proposed TRBM model, we give a brief overview of the theory behind RBMs in the next section.

2.5.1 Restricted Boltzmann Machines (RBM)

An RBM is an undirected graphical model with a set of binary visible units $\mathbf{v}$, a set of binary hidden units $\mathbf{h}$, and a weight matrix $\mathbf{W}$ for bipartite connections between $\mathbf{v}$ and $\mathbf{h}$. Figure 2.6 shows an RBM with 2 visible and 3 hidden variables. The

probability of an RBM configuration is given by:

$$p(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{v},\mathbf{h})}$$

where Z is the partition function and E is the energy function defined as:

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i \in \text{visible}} a_i v_i - \sum_{j \in \text{hidden}} b_j h_j - \sum_{\substack{i \in \text{visible} \\ j \in \text{hidden}}} v_i h_j w_{ij}$$

where a_i and b_j are biases for corresponding visible and hidden units respectively, and w_{ij} is the symmetric weight between v_i and h_j.

In RBMs, given the visible units, the hidden units are conditionally independent of each other, and vice versa. The conditional probability distributions of the variables are given by:

$$p(h_j = 1|\mathbf{v}) = \sigma\left(b_j + \sum_{i \in \text{visible}} v_i w_{ij}\right) \tag{2.1}$$

$$p(v_i = 1|\mathbf{h}) = \sigma\left(a_i + \sum_{j \in \text{hidden}} h_j w_{ij}\right) \tag{2.2}$$

where σ is the logistic sigmoid function: $\sigma(x) = 1/(1 + e^{-x})$.

This easy inference procedure is one of the main advantages of using RBMs over directed models such as ISBNs. Training the RBM weight parameters is, however, difficult as computing the exact gradient of log-likelihood is intractable. Specifically, the gradient is given by:

$$\frac{\partial \log p(\mathbf{v})}{\partial w_{ij}} = \langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{model}} \tag{2.3}$$

where $\langle\rangle_d$ denotes the expectation under distribution d. $\langle v_i h_j \rangle_{\text{data}}$ can be obtained by clamping $\mathbf{v}$ to the observation and then sampling $\mathbf{h}$ using equation 2.1. Unfortunately, computing $\langle v_i h_j \rangle_{\text{model}}$ is not easy. An unbiased sample may be obtained by initializing $\mathbf{v}$ randomly and sampling $\mathbf{h}$ and $\mathbf{v}$ alternatingly using equations 2.1 and 2.2, a process

that can take a long time to converge. (Hinton, 2002) proposed a faster learning procedure called Contrastive Divergence (CD) that does only *one step reconstruction* instead of continuing till convergence. In this procedure, given an observation $\mathbf{v}$, the hidden state $\mathbf{h}$ is sampled using equation 2.1. This $\mathbf{h}$ is in turn used to sample a "reconstructed" visible vector which is further used to sample a reconstructed hidden vector. The weight updates are now calculated as:

$$\Delta w_{ij} = \eta(\langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{reconstructed}})$$

where η is the learning rate.

2.5.2 Proposed TRBM Model Structure

Temporal RBMs (TRBM) (Taylor et al., 2007; Sutskever and Hinton, 2007) can be used to model sequences where the decision at each step requires some context information from the previous time steps. This information is provided by having directed connections from the hidden/visible vectors of a selected set of past steps to the current step. Figure 2.7 shows our proposed TRBM model with hidden to hidden connections between time steps. Each time step has an RBM with a visible and hidden layer. Figure 2.7 only shows one derivation path in the complete search tree. Figure 2.8 illustrates the search space of our TRBM model. In the first step, an RBM computes the probability of each decision starting from an initial configuration. In the second step, the two new configurations replicate the history and add new RBMs of their own. Thus, we now have two different derivations in the search tree, each with its own TRBM. The history at Step 2 consists only of the RBM at the first step. The new RBMs have directed connections coming from the RBMs at previous steps. The probability distribution of a TRBM is:

$$p(\mathbf{v}_1^T, \mathbf{h}_1^T) = \prod_{t=1}^{T} p(\mathbf{v}^t, \mathbf{h}^t | \mathbf{h}^{(1)}, ..., \mathbf{h}^{(C)})$$

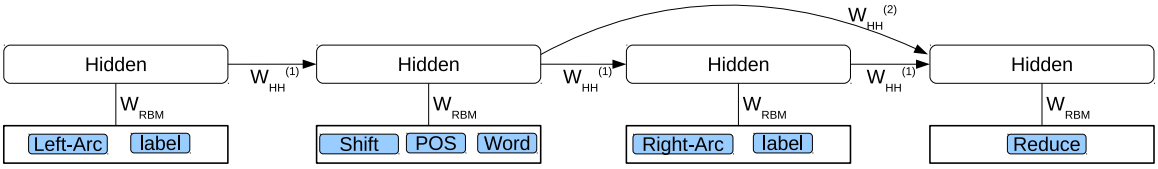


Figure 2.7: Proposed TRBM Model. Shaded nodes represent the visible vectors and 'Hidden' represents a vector of hidden variables. Edges with no arrows represent undirected RBM connections with weight matrix W_{RBM}. $W_{HH}^{(c)}$ denotes the weight matrix for temporal connection of type c and all the connections of type c share this weight matrix, where $c \in [1, 7]$ corresponding to the 7 kinds of connections described in section 2.3.

where $\mathbf{v}_1^T$ denotes the set of visible vectors from time steps 1 to T i.e. $\mathbf{v}^1$ to $\mathbf{v}^T$. The notation for hidden vectors $\mathbf{h}$ is similar. $\mathbf{h}^{(c)}$ denotes the hidden vector in the past time step that is connected to the current hidden vector through a connection of type c. To simplify notation, we will denote the past connections $\{\mathbf{h}^{(1)}, ..., \mathbf{h}^{(C)}\}$ by $history^t$. The conditional distribution of the RBM at each time step is given by:

$$p(\mathbf{v}^t, \mathbf{h}^t | history^t)$$
$$= \frac{1}{Z} exp\left(\sum_{i \in \text{visible}} a_i v_i^t + \sum_{\substack{i \in \text{visible} \\ j \in \text{hidden}}} v_i^t h_j^t w_{ij} + \sum_{j \in \text{hidden}} \left(b_j + \sum_{\substack{c \in \text{connections} \\ l \in \text{hidden}}} w_{HH_{lj}}^{(c)} h_l^{(c)} \right) h_j^t \right)$$

where v_i^t and h_j^t denote the ith visible and jth hidden variable respectively at time step t. $h_l^{(c)}$ denotes a hidden variable in the past time step, and $w_{HH_{lj}}^{(c)}$ denotes the weight of the corresponding connection.

The TRBM model is similar to the ISBN model (Model 1) with two key differences: (i) the directed connections between hidden and visible vectors in ISBNs become undirected RBM connections in TRBMs, (ii) at any given time step, ISBNs require a mean-field approximation of their hidden variables before the action probabilities can be calculated, whereas TRBMs only require it afterwards, as explained in the next section.

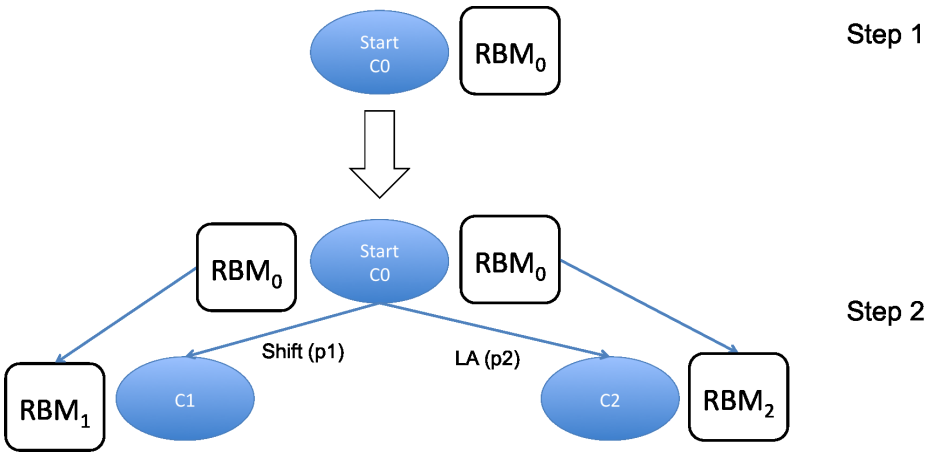


Figure 2.8: An illustration of the search space of TRBM based parsing.

2.5.3 TRBM Likelihood and Inference

Section 2.5.1 describes an RBM where visible variables can take binary values. In our model, similar to (Salakhutdinov et al., 2007), we have multi-valued visible variables which we represent as one-hot binary vectors and model via a softmax distribution:

$$p(v_k = 1|\mathbf{h}) = \frac{exp(a_k + \sum_j h_j w_{kj})}{\sum_{i \in \text{visible}} exp(a_i + \sum_j h_j w_{ij})} \tag{2.4}$$

As shown in figure 2.7, the hidden layer at every time step is connected to hidden layers of a selected set of previous time steps via directed connections. In the TRBM

model, these temporal connections simply contribute a bias during the inference procedure. Concretely, equation 2.1 now becomes:

$$\begin{aligned}\mu_j &= p(h_j = 1|\mathbf{v}, history^t) \\ &= \left\langle \sigma\left(b_j + \sum_{\substack{c\in\text{connections}\\ l\in\text{hidden}}} w^{(c)}_{HH_{lj}} h^{(c)}_l + \sum_{i\in\text{visible}} v_i w_{ij}\right)\right\rangle \\ &\approx \sigma\left(b'_j + \sum_{i\in\text{visible}} v_i w_{ij}\right), \end{aligned} \tag{2.5}$$

where

$$b'_j = b_j + \sum_{\substack{c\in\text{connections}\\ l\in\text{hidden}}} w^{(c)}_{HH_{lj}} \mu^{(c)}_l.$$

Here, $h^{(c)}$ denotes the hidden vector in the past time step that is connected to the current hidden variable through a connection of type c, $\mu^{(c)}$ denotes its means, and $w^{(c)}_{HH}$ denotes the weight matrix of the corresponding connection. To keep inference tractable, we do not do any backward reasoning across directed connections to update $\mu^{(c)}$. Thus, the inference procedure for hidden variables takes into account both the parser history and the current observation, but no future observations.

The limited set of possible values for the visible layer allows us to marginalize out hidden variables in linear time to compute the exact likelihood. Let $\mathbf{v}^k$ denote a vector with $v_k = 1$ and $v_{i(i\neq k)} = 0$. The conditional probability of a sub-decision is:

$$\begin{aligned} p(\mathbf{v}^k|history^t) &= \frac{1}{Z}\sum_{\mathbf{h}} e^{-E(\mathbf{v}^k,\mathbf{h})} \\ &= \frac{1}{Z} e^{a_k} \prod_{j\in\text{hidden}} \left(1 + e^{b'_j + w_{kj}}\right), \end{aligned} \tag{2.6}$$

where

$$Z = \sum_{i\in\text{visible}} e^{a_i} \prod_{j\in\text{hidden}} \left(1 + e^{b'_j + w_{ij}}\right).$$

Note that the sum over visible values in Z is normally not tractable but we only have a limited number of terms in this sum.

We actually perform this probability calculation once for each sub-decision, ignoring the future sub-decisions in that time step. This is a slight approximation, but avoids having to compute the partition function over all possible combinations of values for all sub-decisions. In cases where computing the partition function is still not feasible (for instance, because of a large vocabulary), sampling methods could be used. However, we did not find this to be necessary.

The complete probability of a derivation is:

$$p(\mathbf{v}_1^T) = p(\mathbf{v}^1).p(\mathbf{v}^2|history^2)...p(\mathbf{v}^T|history^T)$$

2.5.4 TRBM Training

The training procedure of a TRBM is similar to that of an RBM with the addition of learning rules for temporal connections. Equation 2.6 gives the likelihood of training data by marginalizing out hidden variables. Now, instead of learning by Contrastive Divergence (CD), we can perform maximum likelihood training by doing gradient ascent on the log likelihood of training data.

$$\frac{\partial \log p(\mathbf{v}^k|history^t)}{\partial w_{ij}} = (\delta_{ki} - p(\mathbf{v}^i|history))\sigma(b'_j + w_{ij}) \tag{2.7}$$

Equations 2.3 and 2.7 are equivalent, but the former is used with an approximation such as CD. In our experiments, we found training using equation 2.7 gave much better results than CD. It could be due to the fact that CD is only approximately following the gradient of the log likelihood (Hinton, 2002) whereas equation 2.7 is following the exact gradient.

The weights on the temporal connections are learned by back-propagating the likelihood gradients through the directed links between steps. The back-proped gradient from future time steps is also used to train the current RBM weights.

We also added a gaussian regularization term for the model parameters to the likelihood function, although the regularization weight was gradually reduced through

the course of training.

2.5.5 Decoding

As there are exponentially many parse trees possible for a given sentence, it is intractable for the parser to keep track of all possible configurations and then select the global optimum. Therefore, the parser uses pruning strategies after each action as follows. At any given stage, given a partially built parse tree and the corresponding TRBM, Stack and Queue structures, the parser first calculates the probabilities for possible next actions using equation 2.6. Note that this probability might be composed of probabilities of smaller decisions as illustrated in figure 2.7. Each action gives rise to a new configuration, i.e. an updated parse tree, Stack, Queue, and a new RBM time step. For each new configuration, inference of hidden variable means is performed using equation 2.5. All new configurations are then inserted into the list of candidate configurations. Whenever a configuration created after a "Shift" operation is inserted, we prune the candidate list taking only top α configurations. For every "Left-Arc" or "Right-Arc" operations, only the top β dependency labels are considered to create new configurations. α and β define the *beam* of the parser and are optimized using the development set. In our experiments, we found that increasing α beyond 20 and β beyond 3 did not result in much improvement. Using this beam search procedure, the parser collects all final configurations i.e. configurations with empty Stack and Queue, and selects the one with the highest probability to output its parse tree.

2.6 Experiments

2.6.1 Data

We used syntactic dependencies from the English section of the CoNLL 2009 shared task dataset (Hajič et al., 2009). The English section was taken from the Penn Treebank (Marcus et al., 1993) which consists of sentences from the Wall Street Journal and their manually labeled parse trees. The test set in the Penn Treebank

additionally contains some sentences from the Brown Corpus (Francis and Kucera, 1979). Standard splits of training (sections 02-21), development (section 24) and test (section 23) sets were used which have respectively 39279, 1334 and 2399 sentences, and 958167, 33368 and 57676 dependencies. In the training set, there were 46 unique predicted POS tags, 69 unique dependency labels, 39782 unique word forms, and 28258 unique predicted lemmas. To handle word sparsity, we replaced all the *(POS, word)* pairs with frequency less than 20 in the training set with *(POS, UNKNOWN)*, giving us only 4530 tag-word pairs.

Since our model can work only with projective trees and the dataset has about 7.6% non-projective trees, we mapped all the non-projective trees in the training set to the corresponding projective trees using the projectivize method of MaltParser (Nivre et al., 2006a). We trained our model on these *pseudo-projective trees*. After running the decoder on test set, we used the deprojectivize method of MaltParser to map the parser output to the corresponding non-projective trees.

The hyper-parameters of the model are the beam search parameters α and β mentioned in section 2.5.5, the number of hidden variables in every hidden layer, regularization weight, and the learning rate. Although we did not do a complete grid search for the best hyper-parameter configuration, we carried out limited optimization on the development set.

2.6.2 Evaluation

We use the commonly used "Attachment Score" metric to evaluate the parser. There are two versions of this metric:

Unlabeled Attachment Score (UAS) UAS computes the proportion of words for which the predicted head is correct (Eisner, 1996).

Labeled Attachment Score (LAS) LAS computes the proportion of words for which the predicted head as well as the dependency label is correct (Nivre et al., 2004).

We used the evaluation script provided with the CoNLL 2009 shared task to compute

	Model	LAS	UAS
a.	Model 0	60.46	70.25
b.	ISBN w/o features	38.38	54.52
c.	ISBN w/ features	88.79	91.65
d.	TRBM w/o features	86.36	90.08
e.	TRBM w/ features	89.08	91.87
f.	MST (McDonald et al., 2005)	87.07	89.95
g.	$\text{Malt}^{\rightarrow}_{\text{AE}}$ (Hall et al., 2007)	85.96	88.64
h.	MST_{Malt} (Nivre and McDonald, 2008)	87.45	90.22
i.	CoNLL 2008 #1 (Johansson and Nugues, 2008)	90.13	92.45
j.	$\text{ensemble}^{3}_{100\%}$ (Surdeanu and Manning, 2010)	88.83	91.47
k.	CoNLL 2009 #1 (Bohnet, 2009)	89.88	UA*

Table 2.2: LAS and UAS for different models. *UAS for (Bohnet, 2009) was unavailable.

the above scores. As is standard, we used the script with -p option that ignores all the punctuations.

2.6.3 Results

Table 2.2 lists the labeled (LAS) and unlabeled (UAS) attachment scores for different models. The low numbers for Model 0 (row *a*) indicate that without hidden variables, it gets very difficult to capture parser history. The 7 types of past connections that we use in this work (Section 2.3) do not provide enough features/information to correctly predict the next parser action. Note that not all 7 types of connections might be available at any given time step which causes further shortage of features.

Row *b* shows that a simple ISBN model without features, using feed forward inference procedure, does not quite work. As explained in section 2.4, this is expected since in the absence of explicit features, the hidden variables in a given layer do not take into account the observations in the previous layers. The huge improvement in performance on adding the features (row *c*) shows that the feed forward inference procedure for ISBNs relies heavily on these feature connections to compensate for the lack of backward inference.

The TRBM model avoids this problem as the inference procedure takes into account the current observation. Thus, the hidden variables in a given step carry much more information that can be passed onto future steps through temporal connections. However, as row d shows, the TRBM model without features falls a bit short of the ISBN performance which leads us to believe that features are indeed, a powerful substitute for backward inference in sequential hidden variable models. TRBM models would still be preferred in cases where such feature engineering is difficult or expensive, or where the objective is to compute the hidden features themselves. For a fair comparison, we add the same set of features to the TRBM model (row e) and the performance improves by about 1.8% to reach the same level as ISBN with features (the improvement from row c to e was non-significant[1]). Note that even without features, some domain knowledge is still essential to determine the set of temporal connections that provide context information to a given time step.

The improved inference in TRBM does however come at the cost of increased training and testing time. Keeping the same likelihood convergence criteria, we could train the ISBN in about 2 days and TRBM in about 5 days on a 3.3 GHz Xeon processor. With the same beam search parameters, the test time was about 1.5 hours for ISBN and about 4.5 hours for TRBM. Although more code optimization is possible, this trend is likely to remain. We also tried a Contrastive Divergence based training procedure for TRBM instead of equation 2.7, but that resulted in about an absolute 10% lower LAS.

For comparison, we also include the performance numbers for some state-of-the-art dependency parsing systems. (Surdeanu and Manning, 2010) compare different parsing models using CoNLL 2008 shared task dataset (Surdeanu et al., 2008), which is the same as our dataset. Rows $f - j$ show the performance numbers of some systems as mentioned in their paper. Row k shows the best syntactic model in CoNLL 2009 shared task. The TRBM model has about 1% lower LAS and 0.6% lower UAS compared to the best performing model.

[1]We use Dan Bikel's *Randomized Parsing Evaluation Comparator* for significance testing, which is commonly used in the NLP literature to compare parsing results. We take an improvement as non-significant if p-value ≥ 0.05

2.6.4 Hidden Layer Analysis

We analyzed the hidden layers in our models to see if they represented some semantic patterns. A hidden layer is a vector of 100 hidden variables. As shown in Figure 2.7, every *Shift* operations creates a hidden layer, which means we can extract a hidden representation for all the words in our dataset. We took all the verbs in the development set (i.e. words corresponding to POS tags: VB, VBD, VBG, VBN, VBP, VBZ) [2] and partitioned their hidden representations into 50 clusters using the k-means algorithm. Table 2.3 shows some partitions for the TRBM model. The partitions look semantically meaningful but to get some quantitative analysis, we computed pairwise similarity between all word pairs in a given cluster and aggregated this number over all the clusters. The word similarity was calculated using two different similarity measures on the Wordnet corpus (Miller et al., 1990): *path* and *lin*. *path* similarity is a score between 0 and 1, equal to the inverse of the shortest path length between the two word senses. *lin* similarity (Lin, 1998) is a score between 0 and 1 based on the *Information Content* of the two word senses and of the Least Common Subsumer. To account for randomness in k-means clustering, the clustering was performed 10 times with random initializations and similarity scores were computed for each run. Table 2.4 shows the mean similarity scores for different models. We observe that TRBM hidden representations give a slightly better clustering than ISBN models. Again, this is because of the fact that the inference procedure in TRBMs takes into account the current observation. However, at the same time, the similarity numbers for ISBN with features are not very low which shows that features are a powerful way to compensate for the lack of backward inference. This is in agreement with their good performance on the parsing task.

2.7 Discussion

We have presented a Temporal Restricted Boltzmann Machines based model for dependency parsing. The model shows how undirected graphical models can be used to

[2] We selected verbs as they have good coverage in Wordnet.

Cluster 1	Cluster 2	Cluster 3	Cluster 4
says	needed	pressing	renewing
contends	expected	bridging	cause
adds	encouraged	curing	repeat
insists	allowed	skirting	broken
remarked	thought	tightening	extended

Table 2.3: K-means clustering of words according to their TRBM hidden representations. Duplicate words in the same cluster are not shown.

Model	path	lin
ISBN w/o features	0.228	0.381
ISBN w/features	0.366	0.466
TRBM w/o features	0.386	0.487
TRBM w/ features	0.390	0.489

Table 2.4: Wordnet similarity scores for clusters given by different models.

generate hidden representations of local parsing actions, which can then be used as features for later decisions. The model performance compares well with state-of-the-art and similar models used in the past.

The TRBM model for dependency parsing can be extended to a Deep Belief Network (DBN) model by adding one more hidden layer on top of the existing one, with similarly defined temporal connections (Hinton et al., 2006). RBMs/DBNs have also been used to construct higher order features from unsupervised data such as learning features from unlabeled images (Hinton et al., 2006). In a similar fashion, one could construct n-gram features from a large collection of unlabeled text, which might then be used in the parser. (Collobert and Weston, 2008) showed that n-gram features could improve performance on other tasks such as Semantic Role Labeling (SRL). Further, as shown in section 2.6.4, the TRBM hidden representations contain good syntactic and semantic information that might also be useful for other NLP tasks, as was done with ISBN hidden representations for SRL by (Henderson et al., 2008).

A free distribution of our implementation is available at `https://github.com/nikgarg/TRBMParsing`.

Chapter 3

Unsupervised Semantic Role Induction

Semantic Role Labeling (SRL) has received significant interest in recent years due to the increasing need to identify semantic structures in dialogue systems, question answering, information extraction, summarization, and various other text analyses tasks. SRL is the problem of finding semantic relations between different constituents of a sentence. From an information extraction point of view, we would like to find out *who* did *what* to *whom*, *when*, and *where*. Formally, to have domain general definitions, semantic relations are usually defined as predicate-argument structures. For example, the FrameNet project (Baker et al., 1998) annotates a semantic frame as following

$[_{Cook}$ The boys] GRILL $[_{Food}$ their catches $][_{Heating\_instrument}$ on an open fire].

In this sentence, the predicate GRILL has three semantic arguments: *Cook*, *Food*, and *Heating_instrument*.

Statistical systems for SRL have taken advantage of the extensive research in the field of syntactic parsing, and developed SRL models on top of syntactic parse trees. In fact, the PropBank corpus (Kingsbury and Palmer, 2002) was constructed by annotating the constituents of the syntactic parse trees with semantic roles. As a

result, a lot of semantic information is already encoded in the syntactic parse trees. However, the mapping from syntax to semantics varies significantly often to make SRL a hard problem. The following example illustrates the ambiguity in SRL using the PropBank notations:

(a) [$_{Arg0}$ The fire] FILLED [$_{Arg2}$ smoke] in [$_{Arg1}$ the room].

(b) [$_{Arg0}$ The fire] FILLED [$_{Arg1}$ the room] with [$_{Arg2}$ smoke].

In sentence (a), 'smoke' is the direct object of FILLED and gets a semantic label of Arg2 (the substance being filled). However, 'the room' is the direct object of FILLED in the sentence (b), and gets a semantic label of Arg1 (the container).

The availability of annotated resources such as FrameNet (Baker et al., 1998), PropBank (Kingsbury and Palmer, 2002), and NomBank (Meyers et al., 2004) has accelerated the research work in SRL. FrameNet defines a set of semantic frames, where each frame consists of a set of semantically similar *lexical units* or predicates, and a set of *frame elements* or semantic roles. For example, 'Apply_heat' is a frame which can be evoked by lexical units such as *fry*, *bake*, *boil*, and *broil*. Some of the frame elements for this frame are *cook* (the person doing the cooking), *food* (the food being cooked), and *container* (the thing holding the food while it is being cooked). Although FrameNet is an excellent resource that spurred much of the initial work in SRL, the semantic roles defined for each frame are highly specific, and the annotated sentences were carefully chosen to illustrate these roles. As a result, the sentences in the corpus are not a representative sample of the natural language. PropBank alleviates this problem by instead choosing a corpus first (Wall Street Journal (WSJ) news articles from the Penn Treebank (Marcus et al., 1993)), and then annotating all the sentences in the corpus with semantic frames. The semantic roles are divided into two categories, core roles and modifier roles. The core roles: *Arg0*, *Arg1*, *Arg2*, etc. are defined for each predicate separately, although *Arg0* and *Arg1* generally correspond to Agent and Patient theme respectively for all the verbs. The modifier roles: *Arg-TMP*, *Arg-LOC* have common definitions across all the verbs. NomBank is a similar resource as PropBank but for nouns. PropBank also has the advantage

that a lot of research in syntactic parsing has been done on Penn Treebank WSJ corpus, and thus highly accurate parsers are available for this domain. Due to its advantages, PropBank has received much of the research focus in SRL, including the CoNLL SRL Shared tasks of 2004 (Carreras and Màrquez, 2004), 2005 (Carreras and Màrquez, 2005), 2008 (Surdeanu et al., 2008), and 2009 (Hajič et al., 2009). We also use the PropBank corpus as a reference for our SRL experiments.

Much of the earlier work in SRL focused on supervised systems where features extracted from the syntactic parse tree of a sentence were used to build a classifier that predicts semantic roles for the constituents. However, as the need for SRL arises in different domains and languages, the existing manually annotated corpora become insufficient to build such classifiers. Unsupervised learning presents an alternative to the expensive and time consuming task of corpus annotation. Although the research in unsupervised SRL has not been as extensive as its supervised counterpart, recent work has explored various methods for unsupervised learning including Bayesian models (Titov and Klementiev, 2012b), split-merge clustering (Lang and Lapata, 2011a), and graph partitioning (Lang and Lapata, 2011b). So far, the Bayesian models have shown the most promise both in terms of performance as well as a principled way to model a semantic frame. Continuing this line of research, we propose a Bayesian model for unsupervised semantic role induction that takes into account correlations among different roles in a sequence. Existing work in unsupervised SRL has modeled role correlations such as repetition constraints and argument position (left/right) relative to the predicate (Lang and Lapata, 2011a; Lang and Lapata, 2011b; Titov and Klementiev, 2012b). Moreover, supervised systems have shown that modeling additional correlations such as the role sequences can significantly improve the SRL performance (Pradhan et al., 2005; Toutanova et al., 2008). Taking a cue from the supervised systems, we propose a principled way to incorporate role sequence information in a Bayesian model. Our experiments show that modeling the ordering information of some of the roles helps improve the performance as opposed to modeling the complete sequence. This result aligns with the observation in supervised models that using the ordering information of only core-roles helped the model (Pradhan et

al., 2005; Toutanova et al., 2008). We further explore if limited amounts of annotated data could serve as an alternative to complex unsupervised models, and find that having about 10% of the data labeled (about 3.6k sentences in our dataset) is enough to outperform our completely unsupervised model.

We begin by describing the related work in SRL in section 3.1. Section 3.2 describes a typical unsupervised SRL framework, and section 3.3 gives the details of our proposed model. We describe the experiments in section 3.4, and present a closing discussion in section 3.5.

3.1 Related Work

Supervised SRL Systems Supervised SRL has been explored quite extensively following the availability of manually annotated corpora such as FrameNet (Baker et al., 1998), PropBank (Kingsbury and Palmer, 2002), and NomBank (Meyers et al., 2004). PropBank was used in the CoNLL SRL Shared tasks of 2004 (Carreras and Màrquez, 2004) and 2005 (Carreras and Màrquez, 2005), and the joint syntactic and semantic shared tasks of 2008 (Surdeanu et al., 2008) and 2009 (Hajič et al., 2009).

A single article would perhaps not do justice to the vast research in SRL but (Màrquez et al., 2008) provide a good overview of the supervised approaches to this task. Most SRL systems are designed to annotate the nodes of a syntactic tree with semantic roles. (Màrquez et al., 2008) note that although there have been variations, the typical architecture involves three subtasks: (i) filtering candidate arguments, (ii) calculating roles for candidates using local features, and (iii) global optimization to choose the most likely roles compatible with each other.

For the first step, (Xue and Palmer, 2004) proposed simple heuristic rules for filtering the set of candidate arguments for a given predicate, and these rules have been used quite often in the subsequent research. They propose a recursive process that starts at the predicate node of the constituent parse tree and

identifies its siblings as candidate arguments. It then moves to the parent of the predicate and adds its siblings to the list. The process goes on till it reaches the root. In addition, if a constituent is a PP, its children are also collected. (Abend et al., 2009) propose an unsupervised method for argument identification for a given predicate. Their algorithm first detects a set of *minimal clauses* in the constituent parse tree that contain the predicate and restricts the semantic arguments to those contained in the minimal clauses. It then further prunes out argument candidates by using pointwise mutual information as a measure of collocation strength between the predicate and the argument. (Lang and Lapata, 2011a) proposed heuristic rules for identifying arguments bearing nodes in a dependency parse tree. These rules make use of the syntactic features of the candidate argument and the path from the argument to the predicate. We use these rules in our model.

In the second step, probabilities for semantic roles are calculated for each of the candidate arguments using local features. With some variations, most of the research has used the features proposed by (Gildea and Jurafsky, 2002) as a core set of features. These features are calculated from the local context around the predicate and the candidate argument e.g. constituent label, headword of the argument, voice of the verb, etc. In addition, there are features that take into account the path from the predicate to the argument. Subsequent work such as (Pradhan et al., 2005; Xue and Palmer, 2004) has demonstrated an increase in performance by adding additional local features to this core set. Further, (Pradhan et al., 2005) propose to divide this task into sub-tasks of (i) argument identification which classifies each candidate as "argument" or "no-argument", and (ii) argument classification which calculates the role probabilities for each of the identified arguments. The authors found that the features required for these two sub-tasks might be quite different and treating them differently improves the performance.

After the second step, most systems do some post-processing to rerank the candidates based on global features or to filter out SRL labels that are over-lapping,

repeated, etc. This step is discussed further below in the "Role Correlations" part of related work.

Although most systems assume a syntactic parse tree as an input to the system, there has been some work in joint modeling of syntax and semantics e.g. (Henderson et al., 2008). (Henderson et al., 2008) extend the transition-based shift-reduce style syntactic parser to include the semantic derivation as well. The parser switches between syntactic and semantic derivations and includes synchronisation steps in between.

Unsupervised SRL Systems The work on unsupervised SRL has been relatively less well-organized as compared to supervised SRL due to the absence of shared tasks. (Swier and Stevenson, 2004) presented the first work on unsupervised SRL in a domain-general corpus, British National Corpus (Burnard, 2000). They use VerbNet (Schuler, 2005) to identify possible argument structures for each verb and find instances of that verb in the unannotated corpus that have similar syntactic structure. Initially roles are assigned only to the unambiguous cases according to the verb lexicon. An iterative procedure is followed afterwards which creates a probability model based on the current role annotations and uses that to predict roles for the yet unannotated sentences. High confidence annotations are added to the labeled set at each iteration. Although the results were impressive, the model was only built for 54 verbs from VerbNet. Further, VerbNet does not have an associated role-annotated corpus which makes the model difficult to evaluate on a large scale.

(Thompson et al., 2003) proposed a generative model for SRL using FrameNet. Although this model could have been used in an unsupervised or a semi-supervised manner, the model parameters were trained using maximum likelihood on labeled training data. The proposed generative model first selects a predicate, and then a frame conditioned on the predicate. The frame in turn generates an HMM sequence for semantic roles where the roles are hidden and parse tree constituents are visible variables. The HMM model however only includes local probabilities for the allowed sequence of roles. (Grenager and Manning, 2006)

propose a generative model that first generates the verb, then for a selected set of constituents (chosen based on some rules), it generates a linking which is a sequence of pairs of semantic roles and corresponding syntactic relations. This linking is then reordered according to the ordering observed in the sentence. However, due to large space of possible linkings which made unsupervised learning difficult, a separate language-specific rule based method had to be used to constrain this space.

(Lang and Lapata, 2010) formulate the unsupervised SRL problem as one of detecting alternations and finding a canonical syntactic form for each predicate. Their model exploits the fact that most observed syntactic functions will correspond to canonical syntactic functions which correspond well to their semantic roles. For each predicate, their approach aims to cluster the argument instances into groups such that each group corresponds to one semantic role. (Lang and Lapata, 2011a) use an iterative split-merge algorithm which starts with a single cluster containing all the argument instances of a predicate. The split phase sub-divides the clusters based on syntactic features and the subsequent merge phase merges some of them based on lexical features. The split-merge is done iteratively till a desired clustering quality is achieved. (Lang and Lapata, 2011b) propose a graph-partitioning based approach where graph vertices represent argument instances and edge weights encode semantic similarity. The algorithm aims to assign roles to vertices such that similar vertices get the same role. Initially each vertex is assigned to a separate cluster (or role), and a graph partitioning algorithm is used to iteratively update the cluster label of a vertex based on the labels of its neighbors.

(Titov and Klementiev, 2012b) propose a Bayesian model for unsupervised SRL which is closely related to ours. They define *argument-keys* as a tuple of four syntactic features, and a Bayesian model is used to cluster the keys into groups such that keys associated with the same or *similar* head words get assigned to the same cluster. Our model has two main advantages over their model. First, our model incorporates a global role ordering probability that is missing in

theirs. Secondly, in their model, all the arguments having the same argument-keys are assigned the same role. This kind of hard clustering is avoided in our model where two constituents having the same set of features might get assigned different roles if they appear in different contexts.

In a semi-supervised setting, (Fürstenau and Lapata, 2009) propose a method for inferring the semantic role annotations for unseen verbs by using the semantic and syntactic similarity between dependency tree arguments of known and unknown verbs. An integer linear program is used to score graph alignment between two dependency trees. New labeled training instances for an unknown verb are created by projecting role labels from the most similar sentence in the existing training corpus.

Role Correlations in SRL Most of the SRL work has focused on improving local classifiers to predict roles of syntactic constituents, and relatively few methods for incorporating global information have been explored. In the supervised domain, (Gildea and Jurafsky, 2002) use a prior over sets of roles observed in a frame, which is estimated from the training data. This prior, however, does not include any information about the order of roles observed in a frame. (Punyakanok et al., 2004) use an Integer Linear Programming (ILP) framework to enforce some global constraints such as the core arguments cannot repeat. However, this model also ignores the global order of roles. (Pradhan et al., 2005) retain some ordering information by using a trigram language model on the role sequences but this Markov approach ignores the longer range dependencies among roles. (Thompson et al., 2003) use a bigram role model to capture local ordering information. (Toutanova et al., 2008) address this issue by using the complete role sequence as a feature in a log-linear re-ranking model to select among candidate role assignments (for the whole frame) computed using local features. (Henderson et al., 2008) use latent variables to implicitly capture the information about the partial role sequence generated in the past, while generating the current role.

In the unsupervised domain, (Grenager and Manning, 2006) use an ordering over

set of possible linkings of syntactic constituents and semantic roles. (Lang and Lapata, 2011a; Lang and Lapata, 2011b) use the relative position (left/right) of the argument with respect to the predicate as one of the features.

3.2 Unsupervised SRL Pipeline

A typical unsupervised SRL setup, including ours, has the following steps:

1. **Syntactic Parsing** Thanks to the extensive research in syntactic parsing, highly accurate parsers are available that can be used to parse a given sentence. We use a dependency parse because of its simplicity and easier comparison with the previous work in unsupervised SRL, which has mostly used dependency parse trees.

2. **Predicate Identification** We select all the non-auxiliary verbs in a sentence as predicates.

3. **Argument Identification** For each predicate selected in step 2, this step classifies each constituent of the syntactic parse tree as a semantic argument or a non-argument. Following previous work (Titov and Klementiev, 2012a), we use the heuristics for argument identification proposed by (Lang and Lapata, 2011a). These heuristics consist of a set of rules to be applied sequentially to classify a constituent. The rules depends on syntactic features such as the dependency relation of a constituent to its head, path between the constituent and the predicate, etc. For German, the arguments are identified using the LTH system (Johansson and Nugues, 2008). Using the LTH classifier means that the model is based on some supervised data which is not desirable but enables comparison with earlier work.

4. **Argument Classification** Since unsupervised systems do not have access to semantic role labels, they cast the problem as a clustering problem. The aim is to divide the arguments of a predicate in all the sentences into clusters such that each cluster corresponds to a single semantic role. The better this clustering

is, the easier it becomes for a human to give it an actual semantic role label like *Arg0*, *Arg1*, etc. Our model assigns a role variable to every identified argument. This variable can take any value from 1 to N, where N is the number of semantic roles or clusters that we want the model to induce.

The above setup closely follows a typical supervised SRL pipeline (Gildea and Jurafsky, 2002; Pradhan et al., 2005).

3.3 Proposed Model

The proposed model handles Step 4 of the pipeline described in section 3.2. Our model is a generative model, and it is natural to describe it as a top-down generative process. We start from the a predicate and its voice as given visible variables, and generate the semantic role variables in two parts (*Primary* and *Secondary* as explained further below). Each role in turn generates some features of the corresponding constituent. The generative process is designed to incorporate the role ordering and repetition constraints. Before going into the details, let us begin by defining the following terms:

Primary Role (PR) For every predicate, we assume the existence of K primary roles (PRs) denoted by $P_1, P_2, ..., P_K$. These roles are not allowed to repeat in a frame and serve as "anchor points" in the global role ordering. Intuitively, the model attempts to choose PRs such that they occur with high frequency, do not repeat, and their ordering influences the positioning of other roles. For instance, if $K = 2$, the PRs could be *Arg0* and *Arg1*. However, the PRs are not constrained to be only core roles but can also correspond to modifier roles. For ease of explication, we create 3 additional PRs: $START$ denoting the start of the role sequence, END denoting its end, and $PRED$ denoting the predicate.

Secondary Role (SR) The roles that are not PRs are called secondary roles (SRs). Given N roles in total, there are $(N - K)$ SRs, denoted by $S_1, S_2, ..., S_{N-K}$. Unlike PRs, SRs are not constrained to occur only once in a frame and do not participate in the global role ordering.

Ordering An ordering is the sequence of PRs observed in a frame. For example, if the complete role sequence is ($START$, P_2, S_1, S_1, $PRED$, S_3, END), the ordering is defined as ($START$, P_2, $PRED$, END).

Interval An interval is a tuple of adjacent PRs in an ordering. For example, in the role sequence shown above, (P_2, $PRED$) is an interval which contains two SRs, S_1 and S_1. An interval could also be empty, for instance ($START, P_2$) contains no SRs. We say that an interval stretches from position i to position j if the left and right PRs of the interval are at positions i and j respectively in the role sequence. For example, the interval (P_2, $PRED$) stretches from position 1 to 4 in a 0-based indexing.

Since the model is unsupervised, there is no apriori correspondence between the model roles and the actual gold role labels such as *Arg0*, *Arg-TMP*, etc. It is the job of the final evaluation method to map each of the model roles to the corresponding gold roles. For instance, the PR P_2 could get mapped to a core role like *Arg0*, *Arg1*, etc. or to a modifier role like *Arg-TMP*, *Arg-MOD*, etc.

Given these definitions, we can now define the generative process of our Bayesian model.

3.3.1 Generative Process

Figure 3.1 gives a graphical description of the model. The generative process can be described stepwise as follows:

1. **Predicate, Voice** The predicate p and its voice vc are treated as top-level visible variables. All the semantic roles in our model are predicate-specific.

2. **Ordering (Generate PRs)** Select an ordered set of PRs from a multinomial distribution.
   $o \sim Multinomial(\theta^{order}_{p,vc})$
   In figure 3.1, this ordering or sequence of PRs is shown as $(PR_1, PR_2, ..., PR_N)$. Each pair of consecutive PRs, (PR_i, PR_{i+1}), in an ordering corresponds to an

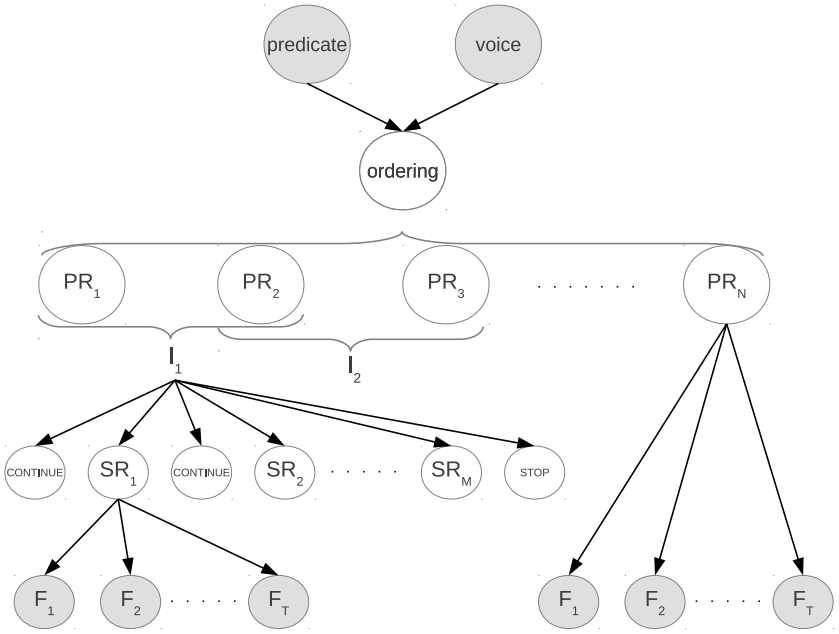


Figure 3.1: Monolingual model. Shaded nodes are visible and unshaded are hidden.

interval I_i. Note that at this stage, we only have the bounding primary roles of I_i and not the internal secondary roles.

3. **Generate SRs** For each interval I in the ordering o, a sequence of SRs, $(SR_{i1}, SR_{i2}, ...SR_{iM})$, is generated using a $STOP/CONTINUE$ process:

   for each interval $I \in o$:
   draw an indicator $s \sim Binomial(\theta^{STOP}_{p,I,0})$
   while $s \neq$ *STOP*:
   choose a SR $r \sim Multinomial(\theta^{SR}_{p,I})$
   draw an indicator $s \sim Binomial(\theta^{STOP}_{p,I,1})$

   Note that in addition to the interval, the indicator variable also depends on whether we are generating the first SR ($adj = 0$) or a subsequent one ($adj = 1$) in that interval.

4. **Generate Features** For each PR and SR, the features for that constituent are generated independently. To keep the model simple and comparable to previous unsupervised work, we only use three features: (i) dependency relation of the

argument to its head, (ii) head word of the argument, and (iii) part-of-speech (POS) tag of the head word. These features are available from the dependency parse tree of the sentence. The dependency relation and the POS tag provide the syntactic information while the head word provides the semantic or lexical information.

for each generated role r:
 for each feature type f:
  choose a value $v_f \sim Multinomial(\theta^F_{p,r,f})$

The above formulation models the global role ordering and repetition preferences using PRs, and models limited context information for SRs using intervals. Using a generative model instead of a discriminative one allows us to easily model unlabeled data. Our experiments show that using large amounts of unlabeled data for training gives significant gains in the model performance.

This particular choice of model is inspired from different sources. Firstly, making the role ordering dependent only on PRs aligns with the observation by (Pradhan et al., 2005) and (Toutanova et al., 2008) that including the ordering information of only core roles helped improve the SRL performance as opposed to the complete role sequence. Although our assumption here is softer in that we assume the existence of some roles (called primary roles) which define the ordering which may or may not correspond to core roles. Using this "partial" ordering information can be viewed as a compromise between (i) the full ordering information which would lead to sparsity and poor generalization, and (ii) no ordering information which would discard important information. This also models the intuition that certain roles such as the modifier roles are relatively free to move around in a semantic role sequence.

Secondly, the SRs are generated independently of each other given the interval (or more precisely, the bounding PRs of the interval), which is based on the intuition that knowing the core roles informs us about the expected non-core roles that occur between them. This intuition is supported by the statistics in the annotated data, where we found that if we consider the core roles as PRs, then most of the intervals tend to have only a few types of SRs (Figure 3.2) and a given SR tends to occur only in a few types of intervals (Figure 3.3). The exponentially decaying distributions in

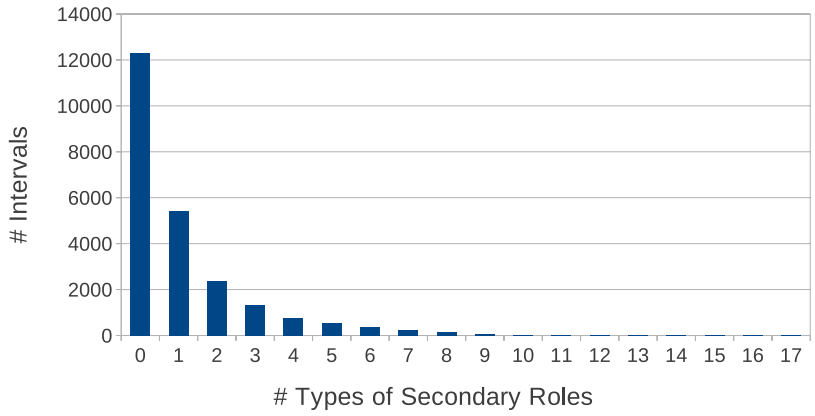


Figure 3.2: Histogram of number of intervals and number of types of SRs occurring in an interval.

both these graphs suggest that the type of non-core role occurring in a sequence has a good correlation with the core roles at the corresponding interval boundary. The simplifying assumption that given the PRs at the interval boundary, the SRs in that interval are independent of the other roles in the sequence, keeps the parameter space limited, which helps unsupervised learning. The concept of intervals also related to the linguistic theory of topological fields (Diderichsen, 1966; Drach, 1937; Höhle, 1983). Topological field parsing divides the sentence into fields (or sections) bounded by the clausal main verb and the subordinating heads. These fields often indicate the number and the type of words that can occur within them. Topological fields have been shown particularly useful for German as the order of fields in a sentence has been found to follow a particular pattern, inspite of the word order being relatively free with respect to the grammatical functions (Becker and Frank, 2002; Cheung and Penn, 2009).

Thirdly, not allowing some or all roles to repeat has been employed as a useful constraint in previous work (Punyakanok et al., 2004; Lang and Lapata, 2011b), which we use here for PRs. (Punyakanok et al., 2004) model several argument labeling constraints via an Integer Linear Program (ILP) such as the core arguments $Arg0$ - $Arg5$ cannot repeat. (Lang and Lapata, 2011b) include a constraint in their split-merge

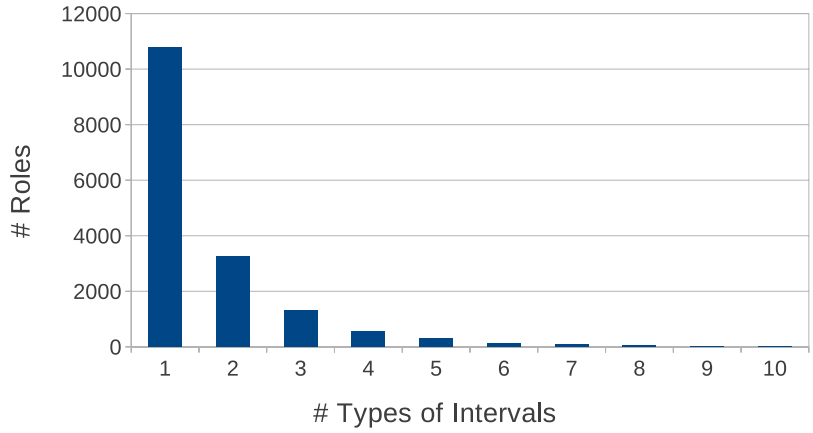


Figure 3.3: Histogram of number of SRs and number of types of intervals a SR occurs in.

algorithm that the argument clusters should not merge if the merged cluster would have "many" arguments in the same semantic frame, which ensures that arguments would not repeat in a frame. The first approach models repetition constraints for core roles, the second approach models it for all the roles. In contrast, our approach models it for a limited number of roles (PRs) which may or may not correspond to core roles. It is left to the learning algorithm to determine the most likely roles for this constraint.

Lastly, conditioning the ($STOP/CONTINUE$) indicator variable on the adjacency value (adj) is inspired from the DMV model (Klein and Manning, 2004) for unsupervised dependency parsing. We found in the annotated corpus that if we map core roles to PRs, then most of the time the intervals do not generate any SRs at all. So, the probability to $STOP$ should be very high when generating the first SR. Figure 3.4 shows the statistics collected from an annotated corpus.

3.3.2 Probability Equations

We now give the joint and marginal probability equations for the model. We use boldfaced variables to denote a sequence of values. Let $\mathbf{r}$ denote the complete sequence of

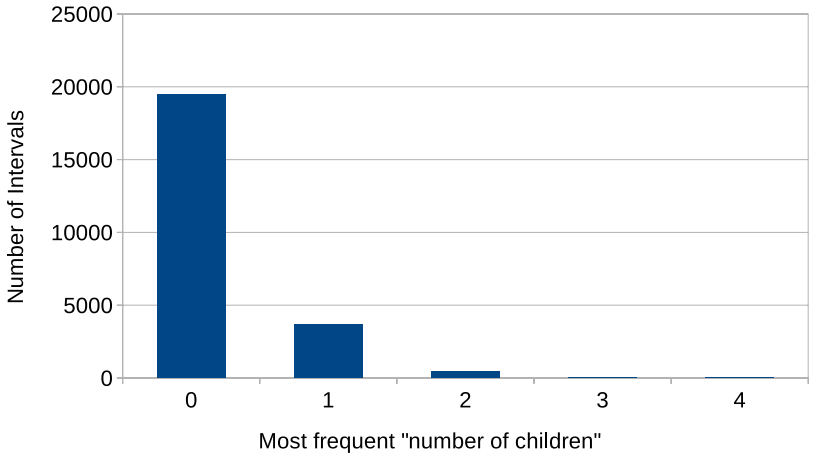


Figure 3.4: Histogram of number of intervals and number of children occurring in an interval.

roles, and $\mathbf{f}$ denote the complete sequence of features. p and vc denote the predicate and its voice respectively. o denotes the ordering of PRs in the sequence $\mathbf{r}$ and $ordering(\mathbf{r})$ is a function for computing this ordering. r_i and f_i denote the role and features at position i respectively, and $\mathbf{r}(I)$ and $\mathbf{f}(I)$ respectively denote the SR sequence and feature sequence inside interval I ($\mathbf{r}(I)$ and $\mathbf{f}(I)$ do not contain the PRs at the interval boundary.). $f_{i,t}$ denotes the value of feature t at position i. p, vc, and f_i are the observed variables in the model. The inference problem is to determine the values of r_i for each position i in the sequence. We can write the joint probability of the model as:

$$P(\mathbf{r}, \mathbf{f}|p, vc) = \underbrace{P(o|p, vc)}_{o=ordering(\mathbf{r})} \underbrace{\prod_{r_i \in \mathbf{r} \cap PR} P(f_i|r_i, p)}_{\text{Primary Roles}} \underbrace{\prod_{I \in o} P(\mathbf{r}(I), \mathbf{f}(I)|I, p)}_{\text{Intervals}} \quad (3.1)$$

where

$$P(\mathbf{r}(I), \mathbf{f}(I)|I, p) = \prod_{r_i \in \mathbf{r}(I)} \underbrace{P(continue|I, p, adj)}_{\text{generate indicator}} \underbrace{P(r_i|I, p)}_{\text{generate SR}} \underbrace{P(f_i|r_i, p)}_{\text{generate features}} \underbrace{P(stop|I, p, adj)}_{\text{end of the interval}} \quad (3.2)$$

and

$$P(f_i|r_i, p) = \prod_{t=1}^{T} P(f_{i,t}|r_i, p, t) \tag{3.3}$$

Thus, every role in $\mathbf{r}$ is either generated as a part of the ordering o (as a PR), or as a part of $\mathbf{r}(I)$ (as a SR) for some interval $I \in o$. The marginal probability of the observed features can be obtained by summing over all possible role sequences:

$$P(\mathbf{f}|p, vc) = \sum_{\mathbf{r}} P(\mathbf{r}, \mathbf{f}|p, vc) \tag{3.4}$$

The likelihood of the whole corpus is a product of marginal probabilities of individual semantic frames. Let the superscript (d) denote the model variables for the training instance $\#d$.

$$L = \prod_{d=1}^{D} P(\mathbf{f}^{(d)}|p^{(d)}, vc^{(d)}) \quad \text{where } D = \text{ number of training instances.} \tag{3.5}$$

Table 3.1 shows how the model can be described in terms of a Probabilistic Context Free Grammar (PCFG). In the PCFG, we use PR to denote any role label from the inventory of primary roles, and use SR to denote any role label from the inventory of secondary roles. The top-level symbol $S_{p,vc}$ first generates an ordering of PRs. I_0 is used as a top-level symbol for roles inside an interval. The subscripts in symbols I_0 and I_1 are used to distinguish between the generation of first and subsequent SRs in an interval. The symbol F_i denotes the feature of type i.

3.3.3 Parameters

The parameters of the model are:

PR Parameters The parameters for selecting an ordered set of PRs are represented as a multinomial distribution with a symmetric dirichlet prior. The multinomial has N_o dimensions where N_o is the number of valid permutations of PRs.
$P(o|p, vc)$ or $\theta^{order}_{p,vc} \sim Dirichlet(\alpha^{order}_{p})$

Rule	Probability
$S_{p,vc} \rightarrow PR\ I_0\ PR\ I_0\ ...\ PR$	$P(o\|p, vc)$
$I_0 \rightarrow \epsilon$	$P(stop\|I, p, 0)$
$I_0 \rightarrow SR\ I_1$	$P(continue\|I, p, 0)\ P(SR\|I, p)$
$I_1 \rightarrow \epsilon$	$P(stop\|I, p, 1)$
$I_1 \rightarrow SR\ I_1$	$P(continue\|I, p, 1)\ P(SR\|I, p)$
$PR \rightarrow F_1 F_2 \ldots F_T$	$\prod_{t=1}^{T} P(F_t\|PR, p, t)$
$SR \rightarrow F_1 F_2 \ldots F_T$	$\prod_{t=1}^{T} P(F_t\|SR, p, t)$

Table 3.1: PCFG of the proposed model.

SR Parameters Each interval defines a multinomial distribution from which the SRs in that interval are drawn.
$P(SR|I, p)$ or $\theta^{SR}_{p,I} \sim Dirichlet(\alpha^{SR}_p)$

STOP/CONTINUE Parameters Each interval defines two binomial distributions for selecting the indicator variable that says whether to STOP or CONTINUE generating more SRs. One distribution is for $adj = 0$ which is used when generating the first SR, and the other is for $adj = 1$ which is used for all subsequent SRs.
$P(stop|I, p, adj)$ or $\theta^{STOP}_{p,I,adj} \sim Beta(\alpha^{STOP}_p)$

Feature Parameters Each role defines a set of multinomial distributions, one for each feature type, from which the feature values are drawn.
$P(f_t|r, p, t)$ or $\theta^{F}_{p,r,t} \sim Dirichlet(\alpha^{F}_{p,t})$

All the multinomial and binomial distributions[1] have symmetric dirichlet and beta priors respectively.

3.3.4 Training & Inference

As shown in figure 3.1 and explained via the PCFG in table 3.1, our inference procedure builds a tree structure on top of a given constituent sequence. This structure

[1]In our model, the Multinomial and Binomial distributions should actually be called Categorical and Bernoulli distributions respectively since the number of trials is 1, but it is more common to use the former terms in NLP.

assigns some constituents to PRs and other to SRs. The position of PRs is what determines the stretch of the intervals, which then generate the required number of SRs in between. The inference procedure is expected to determine the structure, i.e. the locations of PRs, as well as the labels of all the roles. As the space of possible structures is very large, we use the chart parsing method commonly used in syntactic parsing, to find the highest probability structure. Chart parsing is a bottom-up dynamic programming algorithm that efficiently computes the best (highest probability) possible structure for every subsequence of the given example. The structures over adjacent subsequences are then combined to produce higher level symbols. The procedure continues till the top-level PCFG symbol is reached which stretches over the entire sequence.

The training procedure must efficiently search the space of all possible structures to maximize the likelihood of training examples. We experimented with two training methods, (a) an Expectation-Maximization (EM) based approach (Dempster et al., 1977), and (b) a collapsed Gibbs sampling based approach.

In the EM training, in each iteration, we use the *inside-outside algorithm* (Baker, 1979) to calculate the expected counts of the rules in table 3.1. These counts and the priors are then used to calculate the MAP estimate of the parameters. This estimate serves as an input to the next iteration. The procedure is described in more detail in appendix A.

In the sampling based training, we collect the samples for role labels for every constituent. These sample counts along with the priors are then used to calculate the MAP estimate of the parameters. Appendix B.1 describes this procedure in detail.

The advantage of the EM approach is that in the case of context-free grammars like ours, it is easy to implement and is computationally tractable. It also increases the marginal likehood of the data in every iteration and hence converges to a local optimum, but finding the global optimum requires additional search such as trying out different initial starting parameters. On the other hand, our sampling procedure is based on collapsed gibbs sampling which marginalizes out the model parameters and only samples the role variables in each iteration. So, instead of having to set an initial value for the parameters as in the EM procedure, we now have to set an initial value

for the role variables. As is standard, further care has to be taken in the sampling procedure to ensure an appropriate burn-in period (discarding initial samples which are likely to be far off from the optimal solution), and to ensure sufficient iterations for convergence. In our experiments, we found that the EM procedure gave good parameters in some cases and worse in others (as determined by comparing results with a baseline). In contrast, the sampling procedure was able to find good parameter estimates in all cases, which could be due to the reason that we marginalize over the parameters during sampling and only estimate them in the end, thus avoiding a single point estimate in every iteration as in the case of EM. We note here that it is possible to use other methods such as variational bayes (Attias, 1999), that also avoid computing a point estimate of the model parameters, but we did not explore those alternatives.

Once we have the trained parameters, given a new test instance, we use a bottom-up chart parsing approach to infer the role variables.

3.4 Experiments

3.4.1 Data

The absence of a shared task on unsupervised SRL has resulted in researchers using different datasets and preprocessing, which has made it harder to compare different methods. To compare our results with the most recent work, we follow the experimental setting of (Titov and Klementiev, 2012a). We run our experiments on two languages, English and German, using the CoNLL 2009 shared task corpus (Hajič et al., 2009). The English section of this corpus comes from the Penn Treebank (Marcus et al., 1993) consisting of sentences from the Wall Street Journal. The sentences were first labeled manually with syntactic parse trees, and then the PropBank corpus (Palmer et al., 2005) added semantic roles for all the sentences. The German section comes from the TIGER Treebank (Brants et al., 2004) consisting of sentences from the German newspaper, Frankfurter Rundschau. The sentences were manually labeled with syntactic parse trees. The SALSA project (Burchardt et al., 2006) then

added manual semantic roles annotations for the TIGER project sentences using a German extension of the FrameNet semantic roles (Baker et al., 1998).

The standard training section of the CoNLL dataset has about 40k English sentences and 36k German sentences. In contrast to the standard supervised SRL setup, unsupervised systems typically train and test their models on the same dataset. There are two main reasons for this setup. First, as the model is unsupervised, no labels are used in training which means that evaluating on the labels produced by the model is safe even on the training set. Second, the dev and test sets in the standard train-dev-test split for CoNLL data are very small, and computing/evaluating the clustering on very small number of examples would be unreliable. For appropriate comparison, we keep the same setting as in (Titov and Klementiev, 2012a) for automatic parses and argument identification, which we briefly describe here. The English sentences are parsed syntactically using MaltParser (Nivre et al., 2007) and German using LTH parser (Johansson and Nugues, 2008). We could have also used our TRBM parser described in Chapter 2. However, we chose to use the parses from MaltParser and LTH parser for a head-to-head comparison with (Titov and Klementiev, 2012a). As these parsers are also close to state-of-the-art like ours, the performance numbers will remain close irrespective of the chosen parser. All the non-auxiliary verbs are selected as predicates. This gives us about 3k predicates in English and 0.5k predicates in German. Since a sentence can have multiple predicates, each predicate having its own arguments, we get about 89k predicate instances in English and 17k in German. The arguments for English are identified using the heuristics proposed by (Lang and Lapata, 2011a). However, we get an F1 score of 85.1% (precision=84.7%, recall=85.6%) for argument identification on the English data as opposed to 80.7% reported by (Titov and Klementiev, 2012a). This could be due to implementation differences, which unfortunately makes our English results incomparable. For German, the arguments are identified using the LTH system (Johansson and Nugues, 2008), which gives an F1 score of 86.5%. In total, we get about 240k identified arguments for English, and 32k for German.

As we are doing unsupervised learning, it is possible to use large amounts of unlabeled sentences to train our models better. In addition to our experiments on

the CoNLL dataset, we also perform experiments using the unannotated Europarl corpus (Koehn, 2005). We use the English-German section of the Europarl corpus, which has about 1.5M parallel English-German sentences. Note that for the monolingual models, we do not use any parallel information, and simply treat the sentences as additional unlabeled training data. The same automatic parsing, and predicate, argument identification methods are used as for the CoNLL dataset. Further, we discard any predicates not seen in the CoNLL data since we are evaluating only on the CoNLL data. The Europarl corpus adds about 3.3M predicate instances for English, and 2.6M for German. The number of identified arguments is about 9.1M for English, and 4.4M for German.

Since we use the lexical head of the argument as a feature, it could lead to sparsity in the feature space. We take a simple approach by mapping any lexical item occurring less than 20 times to a special item '$UNKNOWN$'.

3.4.2 Evaluation Measures

We use the same evaluation measures as used by (Lang and Lapata, 2011b), which we briefly describe here. For each predicate, let N denote the total number of argument instances, C_i the instances in the induced cluster i, and G_j the instances having label j in gold annotations.

Purity (PU) measures how well an induced cluster corresponds to a single gold role.

$$PU = \frac{1}{N} \sum_i max_j |C_i \cap G_j|$$

Collocation (CO) measures how well a gold role corresponds to a single induced cluster.

$$CO = \frac{1}{N} \sum_j max_i |C_i \cap G_j|$$

F1 is the harmonic mean of Purity and Collocation.

$$F1 = \frac{2 \cdot PU \cdot CO}{PU + CO}$$

The score for each predicate is weighted by the number of its argument instances, and a weighted average is computed over all the predicates.

3.4.3 Baseline

We use the same baseline used by (Lang and Lapata, 2011b) which has been shown to be difficult to outperform. This baseline assigns a semantic role to a constituent based on its syntactic function, i.e. the dependency relation to its head. If there is a total of N clusters, $(N-1)$ most frequent syntactic functions get a cluster each, and the rest are assigned to the Nth cluster. This baseline performs very well and is hard to beat because of the high correlation between the dependency relations and semantic roles. This correlation is very high especially in English because the PropBank corpus was constructed by first running an automatic rule based semantic role tagger (Palmer et al., 2001) that maps syntactic functions to semantic roles. (Palmer et al., 2005) report that the rule based tagging had an accuracy of 83%, measured on pilot data. The mistakes in the annotations were then corrected manually. Hence, the semantic role annotation process itself ensures a highly accurate mapping from syntax to semantics.

3.4.4 Main Results

Since the dataset has 21 semantic roles in total, we fix the total number of roles in our model to be 21. In section 3.4.5, we see that the model does not actually use all the 21 clusters it is allowed to use. So, setting the number of clusters higher is not likely to improve the results. Further, we set the number of PRs to 2 (excluding $START$, END and $PRED$), and SRs to 21-2=19. Later in section 3.4.8, we evaluate the effect of changing the number of PRs, and we find that setting the number of PRs to 3 for English and 2 for German yields the best results. To keep the setting uniform for both languages, we choose #PR=2 for both.

Method	English			German		
	PU	**CO**	**F1**	**PU**	**CO**	**F1**
Random	46.40	16.20	24.01	62.18	13.68	22.42
Baseline	78.23	79.46	78.84	83.09	79.32	81.16
Sampling	76.29	83.13	**79.56***	82.54	81.94	82.24*
EM	75.65	78.71	77.15	82.21	84.15	**83.17***

Table 3.2: Evaluation using automatic syntax, predicate identification, argument identification. A * denotes a significant improvement in the F1 score over the baseline. Significance is computed using stratified shuffling. We take an improvement as significant if the p-value < 0.05.

Table 3.2 gives the results using the automatic syntax trees, and automatic predicate and argument identification. This corresponds to a 'real world' scenario where the system would start from sentences without any human annotation. We evaluate only on arguments that were correctly identified using the gold semantic roles as a reference. The incorrectly identified arguments do not participate in the evaluation because (i) they do not have any gold semantic labels, and (ii) as the contribution of this work is only in argument classification, similar to previous work, we do not penalize argument identification mistakes. Note that in a full SRL pipeline evaluation which includes argument identification as well as classification, the incorrectly identified arguments would be counted as a mistake. To account for randomness in the initialization and the sampling procedure, each experiment was repeated 10 times and an average of the scores was taken to be the final score.

The first line shows the performance numbers when the roles are assigned randomly to the constituents. Both Purity and Collocation scores are low. The Collocation score is extremely low because the random process uses all the available 21 roles which makes it very unlikely that two constituents with the same gold role label would get assigned the same label by the model as well. The baseline method is a huge improvement over the random method. As discussed in section 3.4.3, this baseline performs extremely well due to high correlation between the dependency labels and semantic roles ensured by the annotation process (during corpus creation).

Next two rows show the results of our model with different training procedures. Overall, in terms of the F1 score, we get an improvement in both English and German

using the sampling training method, but an improvement only in German using the EM procedure. One reason for this could be that as an optimization procedure, the EM method is more prone to get stuck in a local maxima.

Using the CoNLL 2009 dataset (Titov and Klementiev, 2012a), report an F1 score of 80.9% (purity=86.8%, collocation=75.7%) for German, and 83.6% (purity=87.5%, collocation=80.1%) for English. Thus, our method outperforms their model in German, but lags behind in English. However, note that our English results are not directly comparable to theirs due to differences argument identification as discussed in section 3.4.1. As their argument identification score is lower, perhaps their system is discarding "difficult" arguments which leads to a higher clustering score.

For a qualitative evaluation, we analyzed the model output of 5 predicates taken from the most frequent predicates: *say*, *have*, *make*, *sell*, and *take*. Analyzing high frequency predicates gives us a good analyses of the errors because (i) our performance metrics give more weight to more frequent predicates, and (ii) clustering analyses is more reliable when the number of data points (argument instances in our case) are large. We found that our model makes very few mistakes for core roles $A0$, $A1$, and modifier roles $AM-TMP$, $AM-LOC$, and $AM-MOD$. On the other hand, across all the 5 predicates, the model has difficulties distinguishing between modifier roles $AM-ADV$, $AM-DIS$, and $AM-MNR$. For instance, for the predicate *say*, the secondary role S_0 had 263 instances of the gold role $AM-ADV$, and 255 instances of the gold role $AM-DIS$. Similarly, for the predicate *make*, S_0 had 101, 52, and 45 instances of the gold roles $AM-ADV$, $AM-MNR$, and $AM-DIS$ respectively. The PropBank annotations guidelines described in (Bonial et al., 2012) say the following about these 3 roles. Discourse roles ($AM-DIS$) connect two adjacent sentences, for instance *also*, *however*, *but*, etc. Crucially, the guidelines say that the elements such as *however*, *also*, *in addition*, etc. should be tagged as $AM-ADV$ when they relate arguments within the clause (e.g. *Mary reads novels in addition to writing poetry*), as opposed to relating arguments across clauses (e.g. *In addition, Mary reads novels*). Our model does not consider secondary roles in the global role ordering and unfortunately, fails to effectively capture the subtle difference between discourse and adverbial roles. The lexical features of these two

roles are very similar, and the heavy weights on these features forces our model to assign argument instances these two gold roles to the same model role regardless of the argument's position in the clause. Similar effect is observed with the Manner roles. Manner roles ($AM-MNR$) specify how an action is performed, and should be used when an adverb answers a *How?* question. For instance, *works well* is a manner. The guidelines say that the manner roles modify the verb, while the adverbial roles ($AM-ADV$) usually modify the whole sentence. The guidelines give an example of an ambiguous case: *happily* is manner in *She sang happily* and adverbial in *Happily, she sang* (paraphrase: *I am happy that she sang*). Again, our model fails to capture this subtle difference between manner and adverbial roles.

Compared to the baseline, most of the improvement in English comes from the highly frequent predicates. Recall that more frequent predicates have a higher weight in the performance metrics. For example, the English model improves over the baseline for about 57% of the predicates, and if we look only at the most frequent predicates, there is an improvement in 8 out of the 10 most frequent predicates, and 16 out of the top 20. Among the highly frequent predicates, large improvements (more than 2 point F1) were observed in *make*, *include*, *get*, *use*, and *increase*, while there was no highly frequent predicate with large degradations (more than 2 points F1).

Unlike English, we did not observe a strong confusion between roles in German for highly frequent predicates. Analyzing the output of the most frequent 5 predicates: *sagen*, *bleiben*, *kommen*, *heißen*, and *fordern*, we found that only *kommen* had significant confusion between the roles $A0$ and $A1$. Compared to the baseline, the model obtained an improvement in F1 score for about 66% of the predicates. It was hard to pin-point any particular pattern in the predicates that obtained large improvements in terms of their frequency or arguments. Among highly frequent predicates, large improvements (more than 2 points F1) were observed in *bleiben*, *heißen*, *sprechen*, *nennen*, *wissen*, etc., while large degradations (more than 2 points F1) were observed in *zeigen*, *brauchen*, and *drohen*.

To evaluate the errors caused due to automatic syntactic parsing and argument identification, we ran the same experiments in an "artificial setting", using the manually constructed (gold) parse trees and predicate argument identification in the

Method	English			German		
	PU	**CO**	**F1**	**PU**	**CO**	**F1**
Random	44.30	15.82	23.31	59.84	13.45	21.96
Baseline	78.42	78.45	78.43	84.48	78.70	81.49
Sampling	75.47	82.55	**78.85**	82.98	82.53	82.75*
EM	75.71	79.49	77.56	82.19	84.77	**83.46***

Table 3.3: Evaluation using gold syntax, predicate identification, argument identification. A * denotes a significant improvement in the F1 score over the baseline. Significance is computed using stratified shuffling. We take an improvement as significant if the p-value < 0.05.

CoNLL 2009 dataset. Table 3.3 gives these results. We observe that the relative comparison between different methods follows the same pattern.

As we get an improvement in both the languages using the sampling training method, all subsequent experiments are done using the sampling method. Further note that we use the automatic parse trees, and predicate, argument identification and the performance with gold is expected to follow the same pattern as demonstrated above.

3.4.5 Output Analyses

We did the following analyses on the output of our model:

Mapping between gold and model roles

For calculating purity, each model role (or cluster) is mapped to a particular gold role that has the maximum instances in the model cluster. Analyzing the output of our model, we found that in English, about 83% of the PRs got mapped to the gold roles *Arg0* and *Arg1*. In contrast, only about 30% of the SRs got mapped to *Arg0* and *Arg1*. Since PRs define the global role ordering and repetition preferences in our model, this confirms the intuition that (i) the ordering of core roles (in particular *Arg0* and *Arg1*) is an important information for SRL systems, and (ii) the intervals bounded by core roles provide good context information for the classification of other roles. One could also look at the mapping from the other direction, i.e. from gold

roles to model roles as computed during collocation calculations. The results are similar. About 84% of the gold roles *Arg0* and *Arg1* got mapped to PRs as opposed to about 21% of the rest of the gold roles that got mapped to PRs.

The mappings for German were not as peaked as English. Using the Purity mappings, about 84% of the PRs and 61% of the SRs got mapped to the gold roles *Arg0* and *Arg1*. Using the Collocation mappings, about 85% of the gold roles *Arg0* and *Arg1*, and about 53% of the rest of the gold roles got mapped to PRs. Part of the reason why so many SRs get mapped to *Arg0* and *Arg1*, and vice versa, is that there are fewer roles in German compared to English (as can be seen in figure 3.6). One might think that the number of PRs should be reduced to account for overall fewer roles, but as Table 3.4.8 shows, reducing the number of PRs to 1 or 0 gives much worse results in German. Hence, the ordering information provided by having 2 PRs in the model is very beneficial inspite of fewer number of roles.

Number of induced clusters

During inference, if the model assigns a role or a cluster to some constituent, that cluster is considered as used. The clusters never assigned by the model are called unused. The number of induced clusters is the number of clusters used by the model. Figures 3.5 (English) and 3.6 (German) show the average number of model clusters induced by the model with respect to the gold clusters present in the data. To compute this graph, for all predicates using a certain number of gold clusters g, we averaged the number of induced clusters used by those predicates. We found that (i) in general, the number of clusters induced increases with an increase in the actual gold clusters, and (ii) even though the upper limit for the number of clusters in our model is 21, the model prefers to allocate lower number of clusters on average. The reason for the second observation could be that even though the features of two constituents are different, the model might assign them the same role if they occur in the same interval, thus reducing the number of induced roles. This effect is amplified when a predicate has a strong preference for an ordering, which results in the model creating same intervals for different frames, and assigning the same PRs to constituents even in the presence of different features. This effect is not necessarily bad since we would

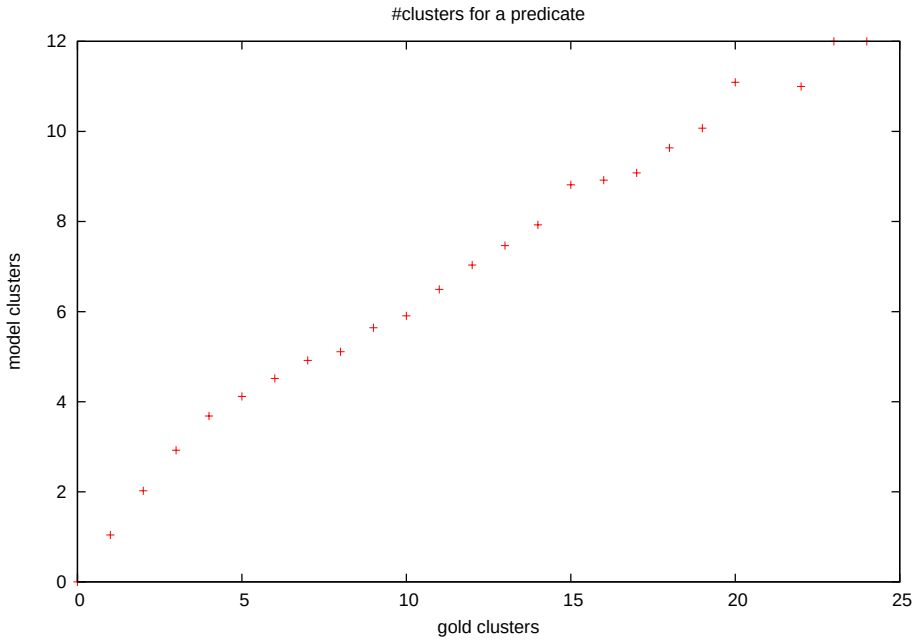


Figure 3.5: Number of model clusters in the output vs the number of actual gold clusters for English data. A point on the graph is calculated as follows: for all predicates using a certain number of gold clusters g, we averaged the number of model (or induced) clusters used by those predicates.

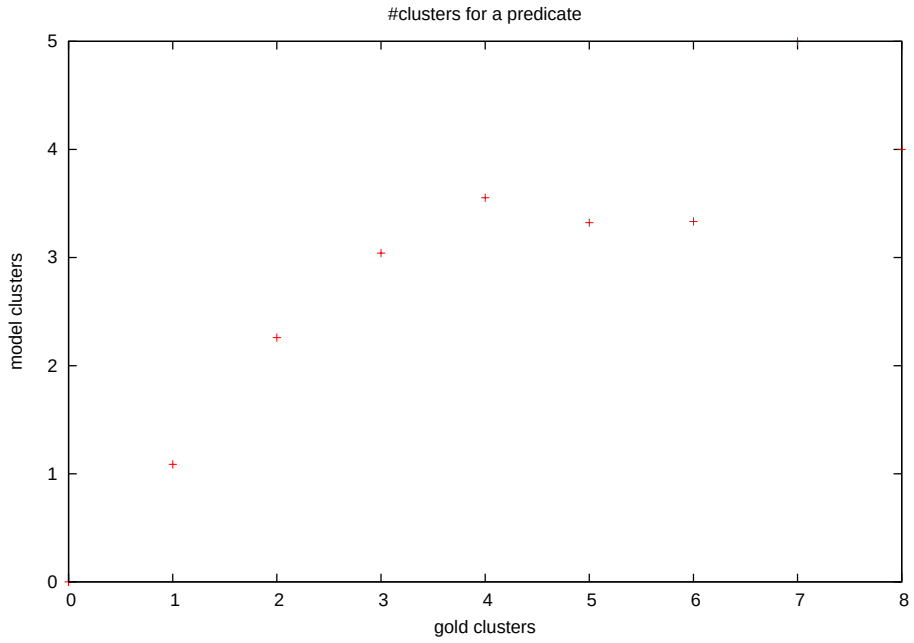


Figure 3.6: Number of model clusters in the output vs the number of actual gold clusters for German data. A point on the graph is calculated as follows: for all predicates using a certain number of gold clusters g, we averaged the number of model (or induced) clusters used by those predicates.

like the model to generalize such that it can put together constituents with different features into the same cluster.

Induced Orderings

We evaluate the orderings induced by our model with respect to the orderings found in the gold standard. To this end, we assume the gold roles *Arg0* and *Arg1* to be the PRs, and classify the orderings into three groups: (i) "verb middle" having PRs both to the left and the right of the verb, (ii) "verb begin" having PRs only to the right of the verb, and (iii) "verb end" having PRs only to the left of the verb. Similarly, the orderings generated by the model were also classified into these three groups. We started with the intuition that the ordering of core-roles is an important information for SRL. Here, we further assume that *Arg0*, *Arg1*, and the verb are the most important constituents for the semantic role order. As we have two PRs in our model, the above classification into 3 groups lets us compare the relative positions of the two most important roles w.r.t. the verb as learned by the model, to the relative positions of *Arg0* and *Arg1* w.r.t. the verb.

For English, we found that out of all the frames having verb middle configurations in the gold annotations, 76% had verb middle configurations in the model predictions. Likewise, 62% of the verb begin configurations in gold had verb begin configurations in the predictions, and 89% of the verb end configurations in gold had verb end configurations in the predictions.

For German, the same numbers were 29% for verb middle, 82% for verb begin, and 89% for verb end configurations. The majority of the gold configurations were verb end in German, which influenced the model to give more weight to verb end predictions. This resulted in a lot of verb middle configurations being incorrectly predicted as verb end configurations.

3.4.6 Effect of labeled data

We tested the effectiveness of having some portion of the data as labeled as opposed to a completely unlabeled dataset. To this end, we randomly selected $S\%$ of the

Supervised sentences	Supervised predicates	Supervised Baseline PU	CO	F1	Model Performance PU	CO	F1
0%	0%	83.09	79.32	81.16	82.58	81.92	**82.24**
1%	20%	80.27	80.16	80.22	80.72	82.59	**81.64**
5%	49%	82.02	83.47	82.74	81.85	84.72	**83.26**
10%	63%	83.86	85.04	**84.45**	82.85	85.70	84.25
20%	74%	86.03	86.87	**86.45**	84.26	86.82	85.52
30%	81%	87.43	87.98	**87.71**	85.34	87.70	86.51
40%	85%	88.16	88.60	**88.38**	86.42	88.41	87.40
50%	90%	88.84	89.17	**89.00**	87.43	88.94	88.18
60%	92%	89.43	89.65	**89.54**	88.12	89.30	88.71
70%	95%	89.84	89.93	**89.89**	88.95	89.71	89.33
80%	96%	90.27	90.26	**90.26**	89.65	90.07	89.87
90%	98%	90.71	90.65	**90.68**	90.32	90.48	90.40
100%	100%	90.97	90.83	**90.90**	90.97	90.83	90.90

Table 3.4: Semi supervised evaluation on German

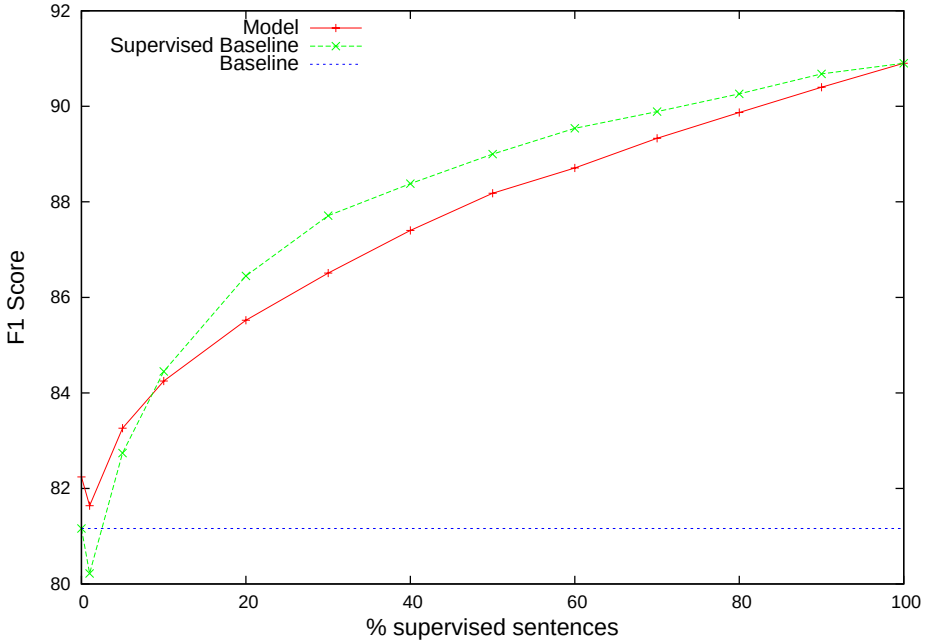


Figure 3.7: Monolingual model performance on German with a portion of data supervised.

Supervised sentences	Supervised predicates	Supervised Baseline			Model Performance		
		PU	CO	F1	PU	CO	F1
0%	0%	85.09	85.64	85.36	84.95	86.03	**85.49**
1%	20%	83.55	86.01	84.76	83.80	86.73	**85.24**
5%	49%	85.61	88.12	**86.85**	84.17	88.15	86.11
10%	63%	87.17	88.03	**87.60**	84.70	88.49	86.55
20%	74%	88.32	88.89	**88.60**	85.51	88.90	87.17
30%	81%	89.20	89.51	**89.35**	86.14	89.18	87.63
40%	85%	89.32	89.61	**89.46**	87.32	89.77	88.53
50%	90%	89.13	89.04	**89.08**	87.77	89.54	88.65
60%	92%	89.33	89.56	**89.44**	88.45	89.74	89.09
70%	95%	89.39	89.45	**89.42**	88.81	89.51	89.16
80%	96%	89.42	89.57	**89.50**	89.08	89.42	89.25
90%	98%	89.47	89.48	**89.47**	89.29	89.42	89.36
100%	100%	89.40	89.24	**89.32**	89.40	89.24	89.32

Table 3.5: Semi supervised evaluation on German unseen test data

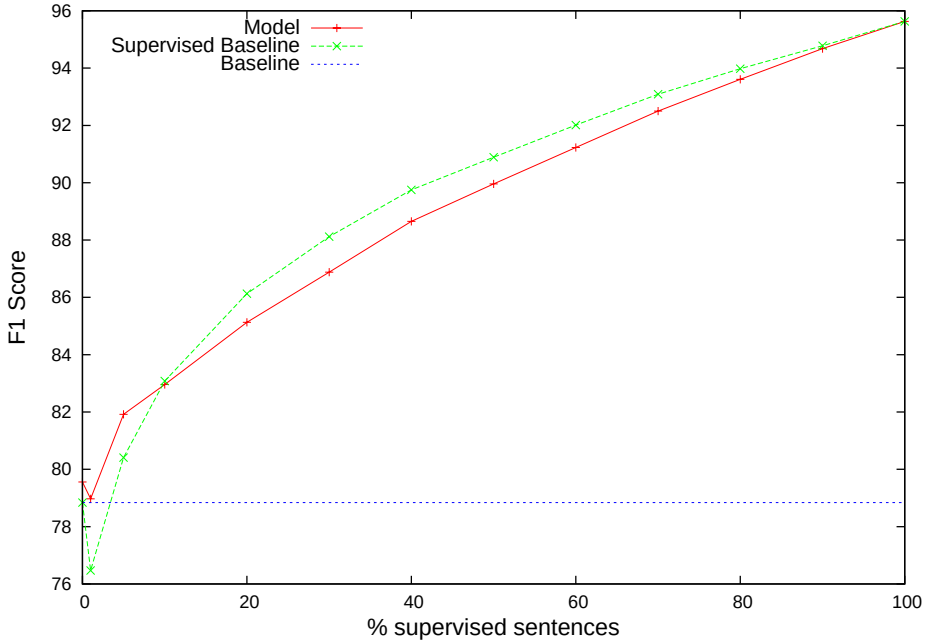


Figure 3.8: Monolingual model performance on English with a portion of data supervised.

Supervised sentences	Supervised predicates	Supervised Baseline			Model Performance		
		PU	CO	F1	PU	CO	F1
0%	0%	78.23	79.46	78.84	76.29	83.13	**79.56**
1%	12%	74.07	79.02	76.47	74.33	84.21	**78.97**
5%	33%	79.03	81.83	80.41	79.03	85.04	**81.92**
10%	44%	82.25	83.93	**83.08**	80.84	85.20	82.96
20%	58%	85.67	86.59	**86.13**	83.65	86.67	85.13
30%	69%	87.94	88.31	**88.12**	85.90	87.87	86.88
40%	76%	89.68	89.82	**89.75**	88.02	89.28	88.65
50%	81%	90.90	90.88	**90.89**	89.65	90.28	89.96
60%	86%	92.09	91.92	**92.01**	91.12	91.34	91.23
70%	91%	93.19	92.98	**93.09**	92.50	92.50	92.50
80%	94%	94.12	93.84	**93.98**	93.70	93.52	93.61
90%	97%	94.92	94.63	**94.78**	94.81	94.54	94.68
100%	100%	95.78	95.48	**95.63**	95.78	95.48	95.63

Table 3.6: Semi supervised evaluation on English

Supervised sentences	Supervised predicates	Supervised Baseline			Model Performance		
		PU	CO	F1	PU	CO	F1
0%	0%	81.62	83.22	**82.41**	79.32	85.22	82.16
1%	12%	76.81	82.69	79.64	77.46	86.49	**81.73**
5%	33%	82.60	85.16	**83.86**	81.31	86.43	83.79
10%	44%	84.91	86.14	**85.52**	83.25	86.49	84.84
20%	58%	87.61	87.70	**87.66**	85.44	87.25	86.34
30%	69%	88.71	88.44	**88.57**	86.73	87.91	87.32
40%	76%	89.11	88.62	**88.86**	87.65	88.24	87.94
50%	81%	89.85	89.03	**89.44**	88.58	88.77	88.67
60%	86%	90.20	89.38	**89.79**	89.11	88.87	88.99
70%	91%	90.47	89.42	**89.94**	89.56	89.11	89.33
80%	94%	90.71	89.45	**90.08**	90.23	89.21	89.72
90%	97%	90.94	89.62	**90.28**	90.80	89.59	90.19
100%	100%	91.10	89.85	**90.47**	91.10	89.85	90.47

Table 3.7: Semi supervised evaluation on English unseen test data

sentences in our dataset as supervised sentences and the rest $(100 - S)\%$ were kept unsupervised. Using the supervised sentences, a mapping was created from the PropBank roles to the model roles as follows. $Arg0$ was mapped to the primary role P_1, $Arg1$ to P_2, and the rest were mapped to the secondary roles $(S_1, ..., S_{19})$ in the order of their decreasing frequency. Next, for the supervised sentences, we assigned a model role to each constituent using this mapping. If $Arg0$ or $Arg1$ was repeated in a frame, the first one was assigned a PR according to the mapping, and the rest were assigned a randomly chosen SR. While training via sampling, the role assignments for the supervised sentences were not sampled. The trained parameters were then used to parse the whole training set, including the supervised as well as unsupervised sentences. The clustering evaluation was also done on the whole training set. The supervised sentences were also taken in the evaluation set because our objective in this section is to simulate a situation where every time we have a new corpus, we would pick $S\%$ of the sentences for labeling, and this labeling as well as the clustering would be different from corpus to corpus. Also, removing the supervised sentences from the evaluation set would have further reduced an already small evaluation set. Nevertheless, we also present evaluation on a completely unseen test set further below. To account for the randomness in selecting the supervised sentences, the experiment was repeated 10 times and average of the performance numbers was taken.

To access the contribution of partial supervision better, we constructed a "supervised baseline" as follows. For the supervised sentences, we assigned the model roles using the mapping scheme described above. These role assignments were then used to calculate a MAP estimate for the predicates seen in the supervised sentences. We parsed the semantic frames with seen predicates using these parameters. For the predicates unseen in the supervised sentences, the standard baseline, described in section 3.4.3, was used that maps dependency label to a semantic role. Thus, the supervised baseline uses the same set of supervised sentences as the model but does not use the sampling training procedure.

Table 3.4 shows the performance of the supervised baseline, and the model performance on the German dataset as we increase the number of supervised sentences. Figure 3.7 shows this variation in a graph. The table also shows the percentage of

"supervised predicates", which is the percentage of predicates seen in the supervised sentences. The F1 scores for both the supervised baseline, as well the model increase with an increase in supervision. This improvement is clearly expected because (i) more sentences in the evaluation set have been labeled using the supervised data, and (ii) more supervised data aids in better sampling of the unsupervised sentences. Note that with $S = 100\%$, the performance is still not 100% because of two reasons: (i) the role assignment for a frame is not perfect since $Arg0$ and $Arg1$ are repeated in some frames while our model does not allow repeated PRs, and (ii) we calculate the model parameters from the role assignments and use those to parse the training instances again which means some performance degradation is expected due to parsing errors. $S = 100\%$ may be treated as an upper bound for our model.

We observe that around 10% mark, the supervised baseline overtakes the model performance which suggests that manually labeling about 10% of the sentences is a good enough alternative to our training procedure. We further observe that the proportion of seen predicates increases dramatically as we increase the proportion of supervised sentences. In particular, at 10% supervised sentences, the model has already seen 63% predicates in German. Perhaps, this is the reason why only 10% of the supervised sentences are enough for the supervised baseline to outperform the model. Note that 10% of the sentences in our dataset amount to about 3.6k sentences.

To evaluate the effect of partial supervision on the parameter estimates better, we used the parameters to parse a completely unseen test set taken from the standard test section of the CoNLL 2009 dataset. Since the test set is very small compared to the training set, the clustering evaluation is not as good as the training set. Nonetheless, as shown in the table 3.5, we observe a similar performance gain with an increase in supervision.

The same evaluation for English is shown in tables 3.6 and 3.7, and figure 3.8. We observe a similar pattern that around 10% supervised sentences, the supervised baseline starts outperforming the model.

3.4.7 Effect of adding unlabeled data

As we are doing unsupervised training, the model could benefit from adding large amounts of unlabeled training data. To this end, we take English and German sentences from the Europarl corpus as described in section 3.4.1, and add them to our training set. Table 3.8 shows the results of adding the Europarl data. When training on CoNLL and Europarl (Line 3), we get almost the same results in English and a substantial improvement in German compared to our previous setting (Line 2). The German dataset in CoNLL is quite small and hence benefits from the additional EP training data. In contrast, the English dataset of CoNLL is perhaps quite sufficient so that adding additional training data does not help. The existing supervised parsers available for English are also able to parse Wall Street Journal data quite effectively, which makes the English CoNLL dataset already quite good for training. Moreover, we should note that the EP data is from European Parliament proceedings whereas CoNLL data is from Wall Street Journal, which are two different domains. Therefore, the improvement is expected to be small. Further, it is interesting to look at Figure 3.7 and note that for German, it takes about 3.5% or 1260 supervised sentences to have the same performance increase as 1.5M unlabeled sentences.

For our next experiment, we train only on Europarl data, and test on the CoNLL data. This setting corresponds to the standard setting of supervised systems where the evaluation is done on an unseen dataset. In a real life scenario, this means that the model parameters are calculated once on a big dataset, and not re-estimated when a new dataset comes in. As shown in Line 4, we get a lower F1 score in English and a higher score in German compared to Line 2. The substantially lower performance in English could be attributed to the lower quality parse trees as the syntactic parsers trained on WSJ are not able to parse Europarl with the same accuracy. The improvement in German is perhaps again due to the issue that the CoNLL data for German is quite small.

	Dataset		Method	English			German		
	Training	**Testing**		**PU**	**CO**	**F1**	**PU**	**CO**	**F1**
1.	CoNLL	CoNLL	Baseline	78.23	79.46	78.84	83.09	79.32	81.16
2.	CoNLL	CoNLL	Sampling	76.29	83.13	**79.56***	82.54	81.94	82.24*
3.	CoNLL+EP	CoNLL	Sampling	76.11	83.33	**79.56**	83.77	81.65	**82.70***
4.	EP	CoNLL	Sampling	73.26	80.60	76.76	83.72	81.28	82.48

Table 3.8: Training using Europarl data. A * denotes a significant improvement in the F1 score over the the previous line. Significance is computed using stratified shuffling. We take an improvement as significant if the p-value < 0.05.

3.4.8 Effect of number of PRs

Table 3.9 shows the variation in performance with respect to the number of PRs[2]. As a general trend, we note that for both languages, as we increase the number of PRs, there is an increase in purity and decline in collocation. This behavior could be explained by the fact that increasing the number of PRs also increases the number of intervals, which makes the probability distributions more sparse. In the extreme case, where all the roles are PRs and there are no SRs, the model would just learn the complete sequence of roles, which would make the parameter space too large to be tractable. The best tradeoff between purity and collocation, as measured by the F1 score, is achieved when using 3 PRs for English, and 2 PRs for German. We also note that our model performs worse than the baseline when using 0 PRs. In this setting, there are only two possible intervals: $(START, PRED)$ and $(PRED, END)$ which means that the only context information a SR has is whether it is to the left or right of the predicate. Increasing the number of PRs gives increasingly specific context information to a SR. Since a high number of PRs (greater than 3) also results in a worse performance (F1 score), it leads us to conclude that the intervals defined by 2 or 3 PRs serve as good context information for the SRs. In other words, PRs at the interval boundaries provide a good context and increasing this information might lead to sparsity and worse performance.

[2]Note that the system might not use all available PRs to label a given frame instance. #PRs refers to the maximum number of PRs.

# PRs	English			German		
	PU	**CO**	**F1**	**PU**	**CO**	**F1**
Baseline	78.42	78.45	78.43	84.48	78.70	81.49
0	70.37	88.02	78.21	72.29	87.73	79.27
1	75.37	83.88	79.40	80.68	81.12	80.90
2	76.29	83.13	79.56	82.58	81.92	**82.24**
3	77.61	81.90	**79.70**	82.54	81.11	81.82
4	77.53	80.89	79.18	82.67	80.58	81.62
5	77.75	80.80	79.25	82.75	80.48	81.60

Table 3.9: Performance variation with the number of PRs (excluding $START$, END and $PRED$)

3.4.9 Inducing distributed word representations

As observed in several recent studies (Collobert and Weston, 2008; Socher et al., 2011), using distributed word representations or word clusters as features instead of using the words directly can significantly increase the performance by alleviating the issues of lexical sparsity and unknown words. Note that we are using the head word of the constituent as one of the features which could lead to lexical sparsity. Quite often, semantically similar words take the same semantic roles for a given predicate but the proposed model described above does not capture this similarity. In this section, we address the lexical sparsity issue by giving each word a distributed hidden representation and using that as a feature in our model instead of the word directly. If similar words have similar representations, they would be more likely to get assigned the same semantic role. We induce the distributed representations with the help of a neural network integrated into the Bayesian model. Figure 3.9b shows the modification over the original model (Figure 3.9a). The modified part is a neural network with weights (instead of probabilities) on the edges going from role to hidden vector and from hidden vector to the head word. The weights from the hidden vector to the head word are shared among all the predicates whereas the weights from the role to the hidden vector are predicate-specific. Intuitively, the shared weights give the same distributed representations to words irrespective of the predicate, and capture that the words that often appear in the same semantic role, as observed

in different predicates, are similar. Neural network models have been explored in previous work for syntactic parsing and SRL to induce representations for words or parser actions (Henderson and Titov, 2010; Collobert and Weston, 2008; Garg and Henderson, 2011).

The joint and marginal probability Equations 3.1 and 3.4 remain the same, with the only difference being in how the probability of features is calculated, i.e. Equation 3.3. Assuming that the last feature $f_{i,T}$ is the lexical feature, the new equation is:

$$P(f_i|r_i,p) = \prod_{t=1}^{T-1} P(f_{i,t}|r_i,p,t) \quad \cdot \quad P(f_{i,T}|r_i,p,T)$$

$$P(f_{i,T}|r_i,p,T) = \frac{1}{Z} exp(a_{f_{i,T}} + \sum_v w_{f_{i,T}v} h_v)$$

where h_v denotes the vth variable in the hidden vector, and w_{uv} denotes the neural network weight from hidden variable h_v to feature value u. Z is the normalization term calculated as $Z = \sum_u exp(a_u + \sum_v w_{uv} h_v)$, where a_u denotes the bias term for the feature layer. We set the values of hidden variables as $h_v = \sigma(b_v + c_{vr_i})$ where c_{vr_i} denotes the neural network weight from role r_i to h_v, b_v denotes the bias term for h_v, and σ is the logistic sigmoid function: $\sigma(x) = 1/(1 + exp(-x))$.

The training procedure remains the same as before with new rules for training the neural network weights. The weight updates for the neural network are calculated by differentiating the log likelihood of the data (log of Equation 3.5) w.r.t. the corresponding weight parameter. The weights from the role variables to the hidden vector, c_{vr_i}, need to be trained for all possible values of the role labels, resulting in a large increase in the training time, and limiting the number of experiments we could do. Limited experiments indicated a 1% increase in F1 score by using distributed representations. However, detailed experiments need to be done for conclusive results. To increase regularization, the weights from the hidden vector to the feature variables, w_{uv}, can also be trained using external sources, for example, a language model over large unlabeled text.

(a) Original Feature Model

(b) Feature Model with distributed word representations

Figure 3.9: Graphical Illustration of feature models.

3.4.10 Inducing Predicate Classes

The previous section attempted to share information between predicates via shared weights from the hidden word representations to the actual word vectors. In this section, we explicitly share information between predicates by classifying them into classes, and using the same model weights for all the predicates in a given class. We make a slight modification to the generative model proposed in section 3.3 by using a top level random variable z denoting the predicate class, and generating both the verb as well as the semantic structure given this new variable. Figure 3.10 shows this modification. Note that the semantic role variables are now specific to a particular class instead of a predicate. This model assumes a soft clustering of predicates in that each predicate class z has a distribution of predicates $P(p|z)$. As indicated by figure 3.10, the new model makes three changes to the joint probability distribution in equation 3.1: (i) introduction of a class prior $P(z)$, (ii) introduction of probability distribution over predicates $P(p|z)$, and (iii) conditioning the parameters of the semantic structure on z instead of the predicate p, for instance, $P(o|p, vc)$ gets replaced by $P(o|z, vc)$.

Intuitively, the new model is trying to classify predicates into classes such that each class has a strong preference (or skewed distributions) for certain semantic structures, for instance, preference for certain semantic role orderings, preference for certain syntactic features such as head words, POS tags, etc. Shared parameters within a single class help alleviate data sparsity issues to some extent. However, training this model becomes much more costly since we now need to sample semantic role variables

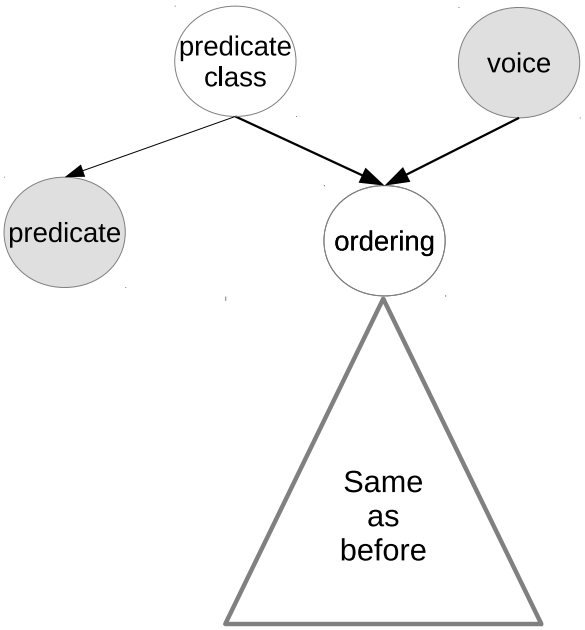


Figure 3.10: Monolingual model extension that induces predicate classes

for every class, which means that the training time gets multiplied by the number of predicate classes.

While decoding, we choose the class that maximizes the joint probability. Evaluation criteria remain the same as before. Unfortunately, this model failed to give any improvements over the model in section 3.3. We observed that the model was estimating the distributions $P(o|p, z)$ to be skewed as desired. However, the estimations for the distributions $P(f_{i,t}|r_i, z, t)$ were quite bad. Since we experimented with only upto 30 predicate classes, this effect could be due to the reduced semantic role inventory of the model. Increasing the number of classes to a larger number made the training procedure intractable.

One possible solution would be to pre-divide the predicates into classes using some external data sources, and then train the model given this clustering. Such a method would fix the predicate classes first and then estimate the parameters of the semantic structure, instead of estimating both at the same time. We tried this solution by using the word clusters computed on the English gigaword corpus (Graff et al., 2003), using two different toolkits distributed by the authors of (Clark, 2003; Mikolov et al.,

2013b). This experiment also failed to improve the results of our SRL model. More experiments using predicate classes from other source such as VerbNet (Schuler, 2005) might help the model.

3.5 Discussion

We propose a unified generative SRL model that incorporates global role ordering information as well as local feature information. For unsupervised semantic role induction, this model outperforms a strong baseline on English and German data. The results indicate that a partial global role ordering that only includes a small number of primary roles (PRs) is a good representation of global ordering constraints for SRL. They also indicate that the intervals between PRs can be well characterized using only the boundary PRs as context information. These conclusions allow global role information to be included in the model while keeping the parameter space small enough even for unsupervised semantic role induction. Additional experiments indicate that labeling about 10% of the sentences is a good alternative to our unsupervised model. Further improvements in German were obtained by adding large amounts of unlabeled training data.

The model can be improved in a number of ways. Firstly, the model assumes all roles to be predicate-specific in contrast to PropBank where modifier roles have the same meaning across predicates. The model could benefit more from shared parameters for modifier roles. Secondly, the proposed model does not use similarity across predicates and has no way of handling unseen predicates. Dividing predicates into classes and sharing model parameters within a single class could help alleviate this problem. Thirdly, external ways of computing word similarity, such as using a language model from a large corpus, could also help alleviate lexical sparsity issues. Lastly, the ordering of primary roles could be derived from a probabilistic model of permutations instead of a simple multinomial distribution. As we observed unimodal ordering distributions in our data and parameter estimates, a unimodal permutations model such as the Mallows model (Mallows, 1957) combined with an appropriate distance metric would be a good fit for our model. A good distance metric would be the

main advantage of this approach because we would be able to incorporate the notion of similarity between different orderings as opposed to treating them independently.

Chapter 4

Bilingual Unsupervised Semantic Role Induction

Multilingual learning has gained significant traction recently for various NLP tasks. In a typical setup, large amounts of parallel text in two or more languages is used to increase the accuracy of monolingual models. For instance, in part-of-speech (POS) tagging, the English word *water* can be either a noun or a verb. If the monolingual POS tagging model for English does not have enough data to accurately predict the tag, a parallel sentence in French could help do the disambiguation. For example, if the aligned word in French is *eau* (as in 'drinking water'), *water* should be a noun, and if it is *arroser* (as in 'water the plants'), *water* should be a verb. (Naseem et al., 2009) demonstrated the effectiveness of multilingual learning in POS tagging using multiple languages.

Similar reasoning could be applied to SRL as well. For instance, consider the following English-German sentence pair:

English: [$_{Arg0}$ Mike] is WRITING [$_{Arg1}$ a book].

German: [$_{Arg0}$ Mike] SCHREIBT [$_{Arg1}$ ein Buch].

If we have the word alignments: *Mike-Mike*, *writing-schreibt*, and *book-Buch*, the model can establish some correspondence between the roles of aligned constituents,

even if it is not a direct mapping i.e. *Arg0-Arg0*, *Arg1-Arg1*, etc. The parallel part provides extra information to the respective monolingual models which could help in role disambiguation, especially in cases where the monolingual data is sparse or not good enough.

In a semi-supervised setting, annotation transfer methods have been explored for SRL where role labels are transferred from the labeled source language to the unlabeled target language (Padó and Lapata, 2009). (Titov and Klementiev, 2012a) take this a step further and propose to exploit the parallel information in SRL by building separate unsupervised monolingual models and training them together with an extra penalty term that ties together roles in the two languages. Continuing this line of research, we propose an extension to our monolingual semantic role induction model which ties together the monolingual models of two languages with the help of bilingual latent variables. The joint bilingual model is a unified model that integrates the monolingual models and the bilingual variables in a single model. The bilingual latent variables couple the aligned constituents together creating a soft role correspondence between them.

The remaining part of this chapter is organized as follow. Section 4.1 describes the related work in the field of multilingual models. We explain our proposed model in section 4.2, list the experiments in section 4.3, and present a concluding discussion in section 4.4.

4.1 Related Work

Bayesian models have been been employed in a number of multilingual NLP problems. (Snyder et al., 2009b; Naseem et al., 2009) demonstrate an increase in unsupervised POS tagging by using a model trained on parallel multilingual text. Each language has its own HMM structure for POS tagging but while training, a set of *superlingual* tags is introduced that generates tags for aligned words in individual languages. A product of experts model is used while training to combine HMM transition probabilities and superlingual probabilities. (Snyder et al., 2009a) extend the Constituent-Context Model (CCM) (Klein and Manning, 2002) for bilingual constituency parsing. Their

probabilistic model first generates an unlabeled alignment tree (Jiang et al., 1995) from the set of all such trees. For each pair of aligned nodes, it then generates a pair of constituents and contexts from bilingual distributions. For unaligned nodes, single constituents and contexts are drawn from monolingual distributions. The model achieves significant gains over the monolingual CCM model.

(Burkett and Klein, 2008) propose constituency parsing in two languages by using a joint log-linear model defined over source trees, target trees, and node-to-node tree alignments. (Cohen and Smith, 2009) use DMV model (Klein and Manning, 2004) for unsupervised dependency parsing in two languages and demonstrate parameter tying in two languages even when the corpus is not parallel.

In another stream of work, annotations are projected from a resource-rich language to a resource-poor language. (Hwa et al., 2005) use a parallel corpus and word alignments in two languages to project dependency relations from the annotated language to the unannotated language. The projected dependencies are then used to train a parser for the resource-poor language. (Padó and Lapata, 2009) project role-semantic annotations from an annotated corpus in one language onto an unannotated corpus in another language. They encode the constituent parse tree nodes in two languages as nodes in a bipartite graph. Similarity between nodes of two languages is defined in terms of word alignment. The aim is to find a maximum matching in the bipartite graph such that it maximizes the product of link similarities. (Van der Plas et al., 2011) first transfer semantic role annotations from English to French using word alignments and then train a joint syntactic-semantic parser. The joint parser results in an improved semantic annotation by using correlations between syntax and semantics.

(Titov and Klementiev, 2012a) presented the first work on cross-lingual unsupervised semantic role induction. They have separate monolingual models for each of the two languages and add an external penalty term to the objective function that aims to maximize role correspondence for aligned constituents.

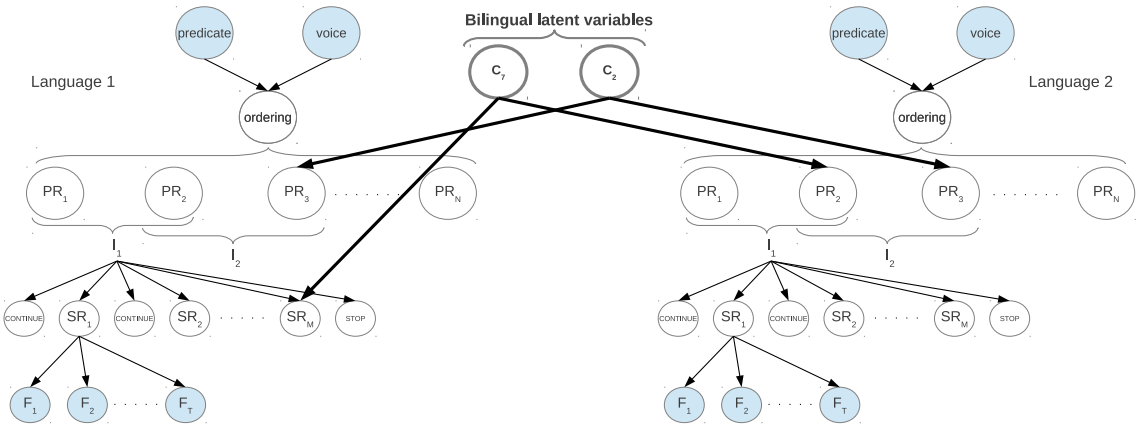


Figure 4.1: Bilingual model. The bilingual latent variables and their associated parameters are drawn in bold. PR_3 in language 1 is aligned to PR_3 in language 2 with the corresponding bilingual latent variable taking the value c_2, and SR_M is aligned to PR_2 with the bilingual latent variable taking the value c_7.

4.2 Proposed Model

The proposed bilingual extension utilizes the word alignments (automatically produced) between the two languages to encourage role correspondence. We keep the monolingual Bayesian model for each language and add additional *bilingual latent variables* to couple the two models, capturing cross-lingual semantic role patterns. Concretely, while training on parallel sentences, whenever the head words of the arguments are aligned, we add a bilingual latent variable as a parent of the two corresponding role variables. Figure 4.1 illustrates this model.

4.2.1 Generative Process

The generative process, as explained below, remains the same as the monolingual model for the most part with the exception of aligned roles which are now generated by both the monolingual process as well as the bilingual latent variables.

1. **Monolingual Data** Given a parallel frame with the predicate pair $p1, p2$, generate separate monolingual frames for each language as in section 3.3.1.

2. **Aligned Arguments** For each aligned argument, first generate a bilingual latent

variable from Chinese Restaurant Process (CRP). Then generate the two aligned roles.

for aligned arguments i, j:
  draw a bilingual latent variable:
    $z \sim CRP(\alpha^{CRP}_{p1,p2})$
  draw role for language $l1$:
    $r_i \sim Multinomial(\theta^{align}_{p1,p2,z,l1})$
  draw role for language $l2$:
    $r_j \sim Multinomial(\theta^{align}_{p1,p2,z,l2})$

Every predicate-pair in the two languages has its own inventory of bilingual latent variables specific to that pair. Each bilingual latent variable is a multi-valued variable where each value defines a distribution over role labels for each of the two languages (denoted by $\theta^{align}_{p1,p2,z,l}$ above). For instance, a value c for the latent variable z might give high probabilities to S_3 and S_8 in language 1, and to S_1 in language 2. Thus, the bilingual model reduces the ambiguity in the monolingual models by preferring to assign the label S_1 in language 2 whenever it encounters S_3 or S_8 in language 1, given that the corresponding bilingual latent variable z has the value c. One can view this as a soft-clustering over pairs of aligned semantic roles where each value of the bilingual latent variable corresponds to a cluster. Since we do not know the number of latent values or clusters beforehand, we generate the clusters via a Chinese Restaurant Process (Pitman, 2002), a non-parametric Bayesian model. Note that we continue to train on the non-parallel sentences using the respective monolingual models.

Another thing to note here is that the bilingual model is deficient as the aligned roles are being generated twice. Ideally, we would like to add the bilingual latent variables as additional conditioning variables in the monolingual models. The new joint probability can be written as:

$$P(\mathbf{r}^{(l1)}, \mathbf{f}^{(l1)}, \mathbf{r}^{(l2)}, \mathbf{f}^{(l2)}, \mathbf{z} | p^{(l1)}, vc^{(l1)}, p^{(l2)}, vc^{(l2)}) = P(\mathbf{z}) \prod_{l \in \{l1,l2\}} P(\mathbf{r}^{(l)}, \mathbf{f}^{(l)} | \mathbf{z}, p^{(l)}, vc^{(l)})$$

where the superscript (l) denotes the variable for language l. $\mathbf{z}$ denotes the common bilingual latent variables for both languages. The right hand side can be further decomposed following the decomposition of the monolingual model in equation 3.1. However, having this additional conditioning variable breaks down the dirichlet-multinomial conjugacy, which makes it intractable to marginalize out the parameters during inference. Hence, we use an approximation where we treat each of the aligned roles as being generated twice, once by the monolingual model and once by the corresponding bilingual latent variable. Thus, the above equation is approximated as:

$$P(\mathbf{r}^{(l1)}, \mathbf{f}^{(l1)}, \mathbf{r}^{(l2)}, \mathbf{f}^{(l2)}, \mathbf{z} | p^{(l1)}, vc^{(l1)}, p^{(l2)}, vc^{(l2)}) \approx P(\mathbf{z}) \prod_{l \in \{l1, l2\}} P(\mathbf{r}^{(l)}, \mathbf{f}^{(l)} | p^{(l)}, vc^{(l)}) \prod_{i,k: z_k \rightarrow r_i^{(l)}} P(r_i^{(l)} | z_k) \quad (4.1)$$

where $z_k \rightarrow r_i^{(l)}$ denotes that the argument at position i in language l is connected to the bilingual latent variable $\#k$. Generating the aligned roles twice gives them more weight compared to the other roles.

This is the first work to incorporate the coupling of aligned arguments directly in a Bayesian SRL model. This makes it easier to see how to extend this model in a principled way to incorporate additional sources of information. First, the model scales gracefully to more than two languages. With more than two languages, the bilingual latent variables simply become multilingual instead, with each one having role distributions for all the languages. If there are a total of n languages, and there is an aligned argument in m of them, the multilingual latent variable is connected to only those m aligned arguments. If an argument appears in more than one alignment, then that argument would be generated one plus the number of alignments it is in. This would be the simplest possible approach but would also increase the deficiency of the model.

Second, the latent variable framework for role alignment could also be very useful in the semantic role transfer methods used in cross-lingual annotation projection models. In the annotation projection setting, the source language is assumed to be labeled, and the semantic role labels are copied from the source constituents to the

corresponding aligned constituents in the target language (for example, (Padó and Lapata, 2009)). The latent variable framework allows for a soft mapping between roles of the two languages instead of a direct one-to-one mapping, which aids in cases where a semantic role in the source language can map to different roles in the target language depending on the context. It also provides a principled way to combine the evidence from the annotation transfer with the monolingual evidence in the target language.

4.2.2 Training & Inference

The generative grammar for our bilingual model is not context-free, which makes it harder to calculate the expected counts as in the EM training of the monolingual model. Hence, we use an extension of the sampling based training procedure of the monolingual model. Appendix B.2 gives the details of this procedure.

We use the parameters obtained from the training process to parse the monolingual data using the monolingual model. Note that the bilingual parameters are ignored even if they were used during training. Thus, the information coming from the bilingual latent variables acts as a regularizer for the monolingual models.

4.3 Experiments

4.3.1 Data

Following (Titov and Klementiev, 2012a), we run our experiments on the English and German sections of the CoNLL 2009 corpus (Hajič et al., 2009), and English-German section of the Europarl corpus (Koehn, 2005). This is the same data that we used for our monolingual model. The preprocessing of this data remains the same as for the monolingual experiments. We use the automatic parse trees, automatic predicate identification, and automatic argument identification, as done in the experiments for the monolingual model. The word alignments for the English-German parallel Europarl corpus are computed using GIZA++ (Och and Ney, 2003). For high-precision, alignments in both directions are computed and only the intersecting alignments are

	Dataset		Model	English			German		
	Training	**Testing**		**PU**	**CO**	**F1**	**PU**	**CO**	**F1**
0	CoNLL	CoNLL	Baseline	78.23	79.46	78.84	83.09	79.32	81.16
1	CoNLL	CoNLL	Monolingual	76.29	83.13	79.56*	82.54	81.94	82.24*
2	CoNLL+EP	CoNLL	Monolingual	76.11	83.33	79.56	83.77	81.65	82.70*
3	CoNLL+EP	CoNLL	Bilingual	76.23	83.25	**79.59**	83.81	81.79	**82.79**
4	CoNLL	CoNLL	T&K's MonoBayes	87.5	80.1	83.6	86.8	75.7	80.9
5	CoNLL+EP	CoNLL	T&K's MultiBayes	86.8	80.7	83.7	85.0	80.6	82.7

Table 4.1: Results. Lines 1, 2 and 3 give the results of our models. Lines 4 and 5 give the results reported by Titov and Klementiev (2012) for monolingual (MonoBayes) and bilingual (MultiBayes) models respectively. Note that their English results are not directly comparable to ours, as discussed in section 4.3.2. A * denotes a significant improvement in the F1 score over the the previous line. Significance is computed using stratified shuffling. We take an improvement as significant if the p-value < 0.05.

kept. We define two semantic arguments as aligned if their corresponding head-words are aligned. In the Europarl corpus, we get 9.1M arguments for English and 4.4M for German. Using the above mentioned procedure for alignment, we get 0.76M aligned arguments.

4.3.2 Main Results

Table 4.1 shows the results of our bilingual experiments. Lines 1 and 2 give the results of our monolingual models as a baseline. Line 3 shows the performance when we use the automatic word alignments in the Europarl data. Comparing with Line 2, we get non-significant improvements in both languages (p-value = 0.33 in English, 0.18 in German). Since there are only 0.76M aligned arguments out of 9.1M arguments in English and 4.4M in German, the small improvement is somewhat expected. As mentioned in Section 4.3.1, we use GIZA++ for computing alignments in both directions and keep only the intersecting alignments for high precision. If we instead use the union of alignments, we get 0.83M aligned arguments. Note that this amounts to increasing the recall of alignments but at the cost of precision. Nevertheless, the number of aligned arguments remains small compared to the overall number of arguments. If we use the union of alignments to run the experiment in Table 4.1 Line 3,

we get F1 scores of 79.56 in English and 82.70 in German, which are insignificantly lower than those reported in Line 3. Perhaps a better alignment scheme rather than simply an intersection or union would a significant improvement. We further note that the German result in Line 3 is almost the same as (Titov and Klementiev, 2012a) MultiBayes model (Line 5). Thus, the bilingual Bayesian model is able to capture the cross-lingual patterns at least as well as the external penalty term in (Titov and Klementiev, 2012a).

For a qualitative analyses of the model, we took the same set of high frequency predicates that we took for the analyses of the monolingual model in Section 3.4.4. Recall that analyzing high frequency predicates gives us a good analyses of the errors because (i) our performance metrics give more weight to more frequent predicates, and (ii) clustering analyses is more reliable when the number of data points (argument instances in our case) are large, and (iii) in the case of bilingual model, high frequency predicates also have a high number of aligned arguments. For English, the predicates were: *say*, *have*, *make*, *sell*, and *take*. All of these predicates had very small differences in the F1 scores between the experiments in Line 2 and 3. Concretely, the difference was +0.27 for *say*, -0.84 for *have*, -0.04 for *make*, -0.28 for *sell*, and -0.30 for *take*. Further, out of the top 10 most frequent predicates, 5 showed an improvement and 5 showed a degradation in F1 score. Overall, there was an improvement in 29% predicates, degradation in 26% predicates, and no change in 45% predicates. This roughly balanced improvement and degradation led to the overall improvement from Line 2 to 3 being insignificantly small. The predicates that showed the most improvement (excluding the very low frequency predicates) were: *base*, *allege*, *revise*, *include*, and *fuel*. Of these *include* and *base* had the most number of instances. Taking a deeper look at the predicate *include*, we found that the monolingual model was frequently assigning the instances of $A1$ and $A2$ to the same cluster, whereas the bilingual model was able to distinguish between them and put them into separate clusters, leading to an improvement in purity. Looking at the alignments, *include* was most frequently aligned to the German predicates *gehören* and *enthalten*. When aligned with *gehören*, the role $A1$ was 90% of the times (455 instances) aligned to the role $A0$ in German, and the role $A2$ was only 56% of the times (36 instances) aligned with the role $A0$ in

German. Similarly, when *include* was aligned with the German predicate *enthalten*, 80% of the times the role $A1$ was aligned to the German role $A1$, while the role $A2$ was only aligned 6% of the times to the German role $A1$. Thus, there were strong signals coming from the alignments that the instances of the roles $A1$ and $A2$ should belong to different clusters.

As another example, for the predicate *base*, the monolingual model was putting the instances of $A2$ and $AM-LOC$ into the same cluster, whereas the bilingual model was able to distinguish between them and put them into separate clusters, leading to an improvement in purity. In the alignments, the role $A2$ of the predicate *base* was aligned 97% (219 instances) to the role $A1$ of the predicate *gründen*, and very few times (only 6) with any other role of *gründen*. In contrast, only one instance of $AM-LOC$ was aligned with the German role $A1$. In fact only two $AM-LOC$ roles had any alignments for the predicate *base*, which means that the $AM-LOC$ instances received only a very weak signal from the alignments. The strong signal given by the alignments of the role $A2$ enabled the bilingual model to separate out the instances of $A2$ from $AM-LOC$.

For German, the 5 most frequent predicates were *sagen*, *bleiben*, *kommen*, *heißen*, and *fordern*. The change in F1 scores going from the monolingual to the bilingual model was +0.8 for *sagen*, +2.5 for *bleiben*, +0.4 for *kommen*, -0.2 for *heißen*, and +3.0 for *fordern*. Out of the top 10 most frequent predicates, 5 showed an improvement and 5 showed a degradation in F1 score. Overall, there was an improvement in 23% predicates, degradation in 27% predicates, and no change in 50% predicates. Similar to the English results, this roughly balanced improvement and degradation led to only a small change in the final performance metrics going from the monolingual to the bilingual model. The predicates that showed the most improvement (excluding the very low frequency predicates) were: *teilen*, *durchsetzen*, *diskutieren*, *stellen*, and *verurteilen*. Among these, *teilen* was the most frequent. Taking a closer look, we found that for the predicate *teilen*, the monolingual model was putting 69% of the instances of $A0$ in the cluster for $A1$ and thus mixing these two roles. In the alignments, the predicate *teilen* was most frequently aligned with the English predicate *share*, and the role $A0$ was aligned 92% of the times (2065 instances) with

the English role $A0$. The role $A1$ on the other hand was aligned 94% of the times (1528 instances) with the English role $A1$. Thus, the strong signal coming from the alignments helped the bilingual model correct the mistakes of the monolingual model. The 69% of the wrongly clustered $A0$ instances of *teilen* were correctly assigned by the bilingual model to the cluster of $A0$ increasing both the purity (due to correct separation of $A0$ and $A1$) as well as the collocation (due to merging of $A0$ instances).

To sum up the qualitative analyses of the bilingual model, we have observed that the predicates that are showing large improvements have strong signals coming from the bilingual alignments, in terms of a large number of instances of a particular role in one language being aligned to a large number of instances of a particular role in the second language. However, due to a low number of alignments in the dataset, such cases are rare and that is probably the reason why the quantitative results in Table 4.1 show that the bilingual model overall results in a very small improvement.

4.3.3 Separate Training & Testing sets

This experiment emulates the standard supervised setup of separate training and test data sets. We train only on the Europarl dataset and tune the parameters of the model. Next, we test on CoNLL dataset which was completely unseen during training. Table 4.2 gives the result of this experiment. Line 1 shows the results when only monolingual models for each language are used[1]. Line 2 shows the results of the bilingual model, potentially benefiting from the aligned arguments in the Europarl data whenever such arguments are present. The model achieves non-significant improvements over the Line 1 setting in English (p-value = 0.09) as well as German (p-value = 0.31).

[1]The performance numbers in Line 1 are expectedly lower than the setting when the model is trained on both Europarl as well as CoNLL data (Line 2 in Table 4.1)

	Dataset		Model	English			German		
	Training	**Testing**		**PU**	**CO**	**F1**	**PU**	**CO**	**F1**
1	EP	CoNLL	Monolingual	73.26	80.60	76.76	83.72	81.28	82.48
2	EP	CoNLL	Bilingual	73.07	81.24	**76.94**	83.59	81.50	**82.54**

Table 4.2: Results for separate training and test datasets. The improvements in the F1 score are non-significant.

	Source	Target	English			German		
			PU	**CO**	**F1**	**PU**	**CO**	**F1**
1	Monolingual Model		76.11	83.33	79.56	83.77	81.65	82.70
2	Bilingual Model		76.23	83.25	79.59	83.81	81.79	82.79
3	English	German	NA			83.83	81.83	**82.82**
4	German	English	76.26	83.37	**79.66**	NA		

Table 4.3: Results for the Bilingual Model with using labeled data for the source language. The improvements in the F1 score are non-significant.

4.3.4 Labeled data for one language

In this experiment, we simulate a situation where we have a small labeled dataset for one language (source), a large parallel unlabeled dataset for the source and another (target) language. We investigate whether this setting improves the parameter estimates for the target language. To this end, we fix the role annotations of the source language in the CoNLL dataset using the semi-supervised baseline described in section 3.4.6. This gives us better parameters for the source language, which helps to improve the role sampling of the source language in the unlabeled Europarl data. The bilingual latent variables aim to capture this improvement and result in better sampling and parameter estimates for the target language. Table 4.3 shows the results of this experiment. We include two previous settings in lines 1 and 2 for comparison. Line 1 shows the results of our monolingual models when trained on CoNLL and Europarl corpus. Line 2 shows the results when the bilingual model is trained on unlabeled sentences from CoNLL and Europarl corpus. This serves as a baseline for this experiment. Lines 3 and 4 show the results of this section's setting. We obtain non-significant improvements over the baseline bilingual model in English (p-value = 0.13) as well as German (p-value = 0.30).

4.4 Discussion

We propose a Bayesian model to capture cross-lingual semantic role correspondence in two languages using word-alignments in parallel text. The proposed bilingual model extends the bayesian monolingual models of individual languages by coupling the aligned constituents together with *bilingual* latent variables. The resulting model is a unified generative bayesian model in which unaligned arguments are generated only by the respective monolingual models, and the aligned arguments are generated by the monolingual models as well as the common bilingual latent variables, which act as a bridge between the semantic role assignments in the two languages. The reduced ambiguity in the role assignments during the training process leads to better parameter estimates for the monolingual models, although giving only non-significant improvements. The design of the model enables us to use the model easily in other settings as well, such as transferring semantic role annotations from a labeled source language to an unlabeled target language.

An improved scheme for calculating role alignments could give us much more aligned constituents and hence further improvement in parameter estimates. In this work, we only had 0.76M aligned arguments in contrast to 9.1M and 4.4M total arguments in English and German respectively. As the nature of the model allows easy extension to more than two languages, future work would include training on aligned corpora of more than two languages. Moreover, we focused only on English and German in this work, so the future work also includes training on different language pairs.

Chapter 5

Conclusions

We have proposed machine learning models for two of the basic problems in Natural Language Processing (NLP), dependency parsing and semantic role labeling. The two problems are part of a common NLP pipeline that takes in a sentence and performs a sequence of operations: parts-of-speech tagging, syntactic parsing, and semantic role labeling. The proposed models for both the problems are generative probabilistic models based on latent variables. The inter-dependence between latent variables captures the parsing history for dependency parsing, and role correlations for semantic role induction.

The first model is a Temporal Restricted Boltzmann Machines (TRBM) based model for transition-based dependency parsing. The model uses Restricted Boltzmann Machines (RBMs) to infer probabilities for the next parser action at each step. The hidden variables of the RBM capture high-dimensional history based features with the help of directed connections from the previous RBMs. The model induces meaningful high-dimensional features and gives good parsing performance comparable to the state-of-the-art models. Neural network based models such as ours have been gaining popularity in NLP recently because of much reduced feature engineering requirements. Our experiments show that the TRBM based model performs close to the best model even in the absence of explicit hand crafted features used in a comparable previous model based on Incremental Sigmoid Belief Networks (ISBNs). Further analyses shows that the abstract features of words automatically induced by

the model correlate better with the Wordnet ontology compared to the ISBN model.

The second model is a Bayesian model for unsupervised semantic role induction. The model is designed to incorporate global role correlations as well as local constituent features in a unified framework. Such correlations have proved useful in supervised models but not used in previous unsupervised model in a principled way due to tractability difficulties. In the proposed model, the partitioning of the role inventory into *Primary* and *Secondary* roles allows the model to choose a limited set of roles (Primary) for the global role correlations, that maximizes the data likelihood. The results on English and German datasets outperform a strong baseline. The model also highlights the importance of role ordering information. In particular, we observe the best performance when using the global role ordering information of only some roles (Primary) as opposed to either no or complete ordering information. Not using the complete ordering information also keeps the parameters space limited and the model tractable for unsupervised learning. Further analyses shows high correlation between the Primary roles chosen by the model and core-roles in the PropBank annotations. We show that the model performance can be improved significantly by either labeling a small portion of the data (about 10% in our dataset), or by adding large amounts of unlabeled data (especially for German which has a small corpus in CoNLL). Other experiments that tried to induce word representations and predicate classes within the bayesian model resulted in little or no improvements.

We extend the bayesian model for a single language to take advantage of the parallel text available for a language pair. The monolingual models for each language are connected together via a set of *bilingual* latent variables that promote semantic role correspondence in aligned constituents. The roles for the aligned constituents are generated by the individual monolingual models as well as their parent bilingual latent variables. This design allows for a soft-role agreement, that is a role in one language can be aligned with different roles in the second language depending on the context. This is in contrast with earlier models where a one-to-one correspondence is usually assumed. Also in contrast to previous models, we incorporate monolingual evidence as well as bilingual alignments in a unified bayesian model instead of optimizing the two separately. The joint bilingual model achieves a non-significant improvements

over the base monolingual models. Part of the reason is very few aligned arguments which could a source of improvement in future. Further experiments show how the model can be used to take advantage of labeled data in a resource-rich language and unlabeled parallel text with a resource poor language, although our experiments obtained only non-significant improvements.

Overall, this thesis explored generative models for two well established problems in NLP. Using latent variables to induce distributed feature representations proved extremely useful and resulted in much reduced feature engineering requirements for the parsing task. The results were mixed in the unsupervised semantic role induction task. The latent variables for roles and their associated encodings of role correlations proved very useful in the monolingual models. However, the bilingual latent variables used to couple aligned arguments together resulted in only non-significant improvements.

Appendix A

Monolingual SRL model: EM based Training

Expectation-Maximization (EM) (Dempster et al., 1977) is an iterative algorithm to estimate the Maximum Likelihood (ML) parameters of a model in the presence of hidden variables. It can also be used to estimate the Maximum a Posteriori (MAP) estimate of the parameters with a slight modification. Let X denote the observed training data, and θ denote all the model parameters. We would like to find the MAP estimate θ^* as:

$$\begin{aligned}\theta^* &= \arg\max_{\theta} \log p(\theta|X) \\ &= \arg\max_{\theta} \log p(X|\theta) + \log p(\theta) \qquad \text{(A.1)} \\ &= \arg\max_{\theta} \log \sum_{Z} p(X, Z|\theta) + \log p(\theta) \qquad \text{(A.2)}\end{aligned}$$

where Z denotes the hidden variables in the model. $p(\theta)$ is the prior of the model parameters, $p(X|\theta)$ is the marginal likelihood, and $p(X, Z|\theta)$ is the joint likelihood. The above optimization is intractable to do directly in most models including ours. The EM algorithm initializes the parameters θ randomly and alternates between the following two steps:

Expectation step (E step) Compute the expectation of $\log p(X, Z|\theta)$.

$$Q(\theta|\theta^t) = E_{Z|X,\theta^t} \log p(X, Z|\theta)$$

where θ^t denotes the parameter estimates at iteration t.

Maximization step (M step) Find the θ that maximizes Q.

$$\theta^{t+1} = \arg\max_{\theta} Q(\theta|\theta^t) + \log p(\theta)$$

For Probabilistic Context Free Grammars (PCFGs) such as ours, these steps can be carried out via the *inside-outside algorithm* (Baker, 1979). In each iteration, this algorithm first computes the inside probabilities P_{IN} and the outside probabilities P_{OUT} for each training example. These probabilities are then used to determine the expected number of times each PCFG rule is used in a training example. The expected counts of each rule are used to reestimate the parameters. Each of these steps is described below for our model:

A.1 Inside-Outside Probabilities

A.1.1 Inside Probabilities

First, a bottom up chart parsing algorithm is used to compute the inside probabilities as follows:

- Calculate the inside probability of constituent at position i having role r:

  $$P_{IN}(i, r) = P(f_i|r, p)$$

  where $P(f_i|r, p) = \prod_{t=1}^{T} P(f_{i,t}|r, p, t)$

- Calculate the inside probability of constituent at position i lying in an interval

I:

$$P_{IN}(i, I) = \sum_{r} P_{IN}(i, r)\ P(r|I, p)$$

- Calculate the inside probability of an interval I spanning the constituents i to j. Let r_{Ibegin} and r_{Iend} denote the PRs at the beginning and end of the interval I respectively.

  Case (a) $j = i + 1$

  $$P_{IN}(i, j, I) = P(f_i|r_{Ibegin}, p)\ \cdot\ P(stop|I, p, 0)\ \cdot\ P(f_j|r_{Iend}, p)$$

  Case (b) $j > i + 1$

  $$\begin{aligned} P_{IN}(i, j, I) =& P(f_i|r_{Ibegin}, p)\ \cdot\ P(continue|I, p, 0)\ \cdot\ P_{IN}(i+1, I) \\ & \cdot \prod_{k=i+2}^{j-1} P(continue|I, p, 1)\ P_{IN}(k, I) \\ & \cdot\ P(stop|I, p, 1)\ \cdot\ P(f_j|r_{Iend}, p) \end{aligned}$$

- Calculate the probability of a sequence of intervals spanning the constituents i to j. Let the sequence of intervals be denoted by $I_1 I_2 ... I_n$.

  $$P_{IN}(i, j, I_1 I_2 ... I_n) = \sum_{k=i+1}^{j-1} P_{IN}(i, k, I_1)\ P_{IN}(k, j, I_2 ... I_n)\ /\ P_{IN}(k, r_{I_1 end})$$

- Calculate the inside probability of an ordering for the whole frame. Let the sequence of intervals in an ordering ord be denoted by $I_1 I_2 ... I_n$.

  $$P_{IN}(0, L-1, ord) = P(ord|p, vc)\ \cdot\ P_{IN}(0, L-1, I_1 I_2 ... I_n)$$

  where L denotes the number of constituents in the frame.

- Calculate the marginal probability of the whole frame.

$$P(\text{frame}) = \sum_{ord} P_{IN}(0, L-1, ord)$$

A.1.2 Outside Probabilities

Next, we calculate the outside probability of an interval I spanning the constituents i to j. We consider all orderings ord of a frame that have interval I. Let the sequence of intervals in an ordering ord be denoted by $I_1...I_t I I_{t+2}...I_n$.

$$P_{OUT}(i, j, I) = \sum_{ord} \frac{P(ord|p, vc)\ P_{IN}(0, i, I_1...I_t)\ P_{IN}(j, L-1, I_{t+2}...I_n)}{P_{IN}(i, r_{Ibegin})\ P_{IN}(j, r_{Iend})}$$

Note that boundary conditions where $I = I_1$ or I_n have to handled separately, but in a similar fashion.

A.2 Expected Counts

We calculate the expected counts of each rule listed in the table 3.1 using the inside-outside probabilities. Since the parameters are all conditioned on the predicate, the following formulae for calculating expected counts are done separately for each predicate. For readability, we omit the conditioning on the predicate. The counts are added for all frames $d = 1$ to D.

- PR or ordering counts:

$$count(S_{p,vc} \rightarrow O) = \sum_{d=1}^{D} count(S_{p,vc} \rightarrow O, W_d) = \sum_{d=1}^{D} \frac{P_{IN}(0, L-1, O)}{P(\text{frame})}$$

  for all training examples with voice vc

- SR counts:

$$count(I_0 \rightarrow r\ I_1) + count(I_1 \rightarrow r\ I_1) = \sum_{d=1}^{D} \frac{\sum_{i=0}^{L-3} \sum_{j=i+2}^{L-1} \sum_{k=i+1}^{j-1} P_{IN}(i,j,I(r_k = r))\ P_{OUT}(i,j,I)}{P(\text{frame})}$$

  where $P_{IN}(i,j,I(r_k = r))$ denotes the inside probability of an interval I spanning constituents i to j with role assigned to constituent k, $r_k = r$. It is calculated similar to $P_{IN}(i,j,I)$ with keeping $r_k = r$ fixed.

- STOP/CONTINUE counts:

$$count(I_0 \rightarrow \epsilon) = \sum_{d=1}^{D} \frac{\sum_{i=0}^{L-2} P_{IN}(i,i+1,I)\ P_{OUT}(i,i+1,I)}{P(\text{frame})}$$

$$count(I_0) = \sum_{d=1}^{D} \frac{\sum_{i=0}^{L-2} \sum_{j=i+1}^{L-1} P_{IN}(i,j,I)\ P_{OUT}(i,j,I)}{P(\text{frame})}$$

$$count(I_1 \rightarrow \epsilon) = \sum_{d=1}^{D} \frac{\sum_{i=0}^{L-3} \sum_{j=i+2}^{L-1} P_{IN}(i,j,I)\ P_{OUT}(i,j,I)}{P(\text{frame})}$$

$$count(I_1) = \sum_{d=1}^{D} \frac{\sum_{i=0}^{L-3} \sum_{j=i+2}^{L-1} (j-i-1)\ P_{IN}(i,j,I)\ P_{OUT}(i,j,I)}{P(\text{frame})}$$

- Feature counts are calculated as shown below. Note that they are done for each feature type separately, but feature type is not mentioned in the formulae for readability.

$$count(r \to f) = \sum_{d=1}^{D} \frac{\sum_{i=0}^{L-3} \sum_{j=i+2}^{L-1} \sum_{k=i+1:f_k=f}^{j-1} \sum_I P_{IN}(i,j,I(r_k=r))\ P_{OUT}(i,j,I)}{P(\text{frame})} \quad \text{for } r \in SR$$

$$count(r) = \sum_{d=1}^{D} \frac{\sum_{i=0}^{L-3} \sum_{j=i+2}^{L-1} \sum_{k=i+1}^{j-1} \sum_I P_{IN}(i,j,I(r_k=r))\ P_{OUT}(i,j,I)}{P(\text{frame})} \quad \text{for } r \in SR$$

$$count(r \to f) = \sum_{d=1}^{D} \frac{\sum_{i=1:f_i=f}^{L-2} \sum_{j=i+1}^{L-1} \sum_{I:r_{Ifirst}=r} P_{IN}(i,j,I)\ P_{OUT}(i,j,I)}{P(\text{frame})} \quad \text{for } r \in PR$$

$$count(r) = \sum_{d=1}^{D} \frac{\sum_{i=1}^{L-2} \sum_{j=i+1}^{L-1} \sum_{I:r_{Ifirst}=r} P_{IN}(i,j,I)\ P_{OUT}(i,j,I)}{P(\text{frame})} \quad \text{for } r \in PR$$

A.3 Parameter Re-estimation

Finally, the parameters are updated using the counts.

- PR or ordering parameters

$$P(o|p,vc) = \frac{count(S_{p,vc} \to O) + \alpha_p^{order}}{count(S_{p,vc}) + \alpha_p^{order}(\#\text{orderings})}$$

- SR parameters

$$P(r|I,p) = \frac{count(I_0 \to r\ I_1) + count(I_1 \to r\ I_1) + \alpha_p^{SR}}{\sum_r count(I_0 \to r\ I_1) + count(I_1 \to r\ I_1) + \alpha_p^{SR}(\#\text{SRs})}$$

- STOP/CONTINUE parameters

$$P(stop|I,p,0) = \frac{count(I_0 \rightarrow \epsilon) + \alpha_p^{STOP}}{count(I_0) + 2\alpha_p^{STOP}}$$

$$P(continue|I,p,0) = 1 - P(stop|I,p,0)$$

$$P(stop|I,p,1) = \frac{count(I_1 \rightarrow \epsilon) + \alpha_p^{STOP}}{count(I_1) + 2\alpha_p^{STOP}}$$

$$P(continue|I,p,1) = 1 - P(stop|I,p,1)$$

- Feature parameters

$$P(f|r,p) = \frac{count(r \rightarrow f) + \alpha_p^F}{count(r) + \alpha_p^F(\#\text{features})} \quad \text{, done for every feature type separately}$$

Appendix B

SRL model: Sampling based Training

B.1 Sampling in Monolingual SRL Model

Let us begin with some notation. We use the superscript (t) for a variable of the training instance t, and the superscript $(-t)$ for the variables of all the training instances except t. The visible variables in our model are the predicate p, voice vc, and features $\mathbf{f}$. The hidden variables are the role variables $\mathbf{r}$, specifically r_i for position i. We use V to denote the visible variables, H to denote the hidden variables, and D to denote all the variables. Without any superscripts, V, H and D denote the variables for all the training instances. For readability, we use θ to denote all the model parameters instead of listing out all the parameters individually. The subscript $-i$ refers to all the positions in a sequence except i.

Our training method aims to choose the best possible role assignments for every frame according to the marginal distribution $P(\mathbf{r}^{(t)}|V)$, i.e. we would like to select

the best role assignment $\mathbf{r}^{*(t)}$ as:

$$\begin{aligned}\mathbf{r}^{*(t)} &= \arg\max_{\mathbf{r}^{(t)}} P(\mathbf{r}^{(t)}|V) \\ &= \arg\max_{\mathbf{r}^{(t)}} \sum_{H^{(-t)}} P(\mathbf{r}^{(t)}, H^{(-t)}|V)\end{aligned}$$

Since it would be intractable to sum over all possible values of $H^{(-t)}$, we use a collapsed Gibbs sampling based approach to sample over all the individual role labels. Each Gibbs sampling step samples from the following distribution:

$$\begin{aligned}P(r_i^{(t)}|\mathbf{r}_{-i}^{(t)}, \mathbf{f}^{(t)}, p^{(t)}, vc^{(t)}, D^{(-t)}) &\propto P(r_i^{(t)}, \mathbf{r}_{-i}^{(t)}, \mathbf{f}^{(t)}|p^{(t)}, vc^{(t)}, D^{(-t)}) \\ &= \int_{\theta} P(r_i^{(t)}, \mathbf{r}_{-i}^{(t)}, \mathbf{f}^{(t)}|\theta, p^{(t)}, vc^{(t)}) P(\theta|D^{(-t)}) \mathrm{d}\theta\end{aligned} \tag{B.1}$$

Following the joint probability equation 3.1, $P(r_i^{(t)}, \mathbf{r}_{-i}^{(t)}, \mathbf{f}^{(t)}|\theta, p^{(t)}, vc^{(t)})$ can be factorized into a product of parameters from different multinomial distributions of the model. The second term $P(\theta|D^{(-t)})$ is a short-hand for the product of posterior probabilities of multinomial distributions, one for each distribution in θ, each of which is a dirichlet distribution due to dirichlet-multinomial conjugacy. Thus, equation B.1 is the expected value of the product of multinomial parameters under dirichlet distributions, which can be computed using a variation of the standard closed form solution:

$$E_{dirichlet(\alpha)}[\theta_j] = \frac{\alpha_j}{\sum_i \alpha_i}$$

We sample the roles at all positions in a randomized order. Note that the role ordering changes implicitly when we sample the PRs. We perform 100 sampling iterations and choose the most frequently sampled role assignment for each training

instance. In other words, we estimate $P(\mathbf{r}^{(t)}|V)$ as:

$$P(\mathbf{r}^{(t)}|V) \approx \frac{\text{freq}(\mathbf{r}^{(t)})}{\#\text{samples}}$$

Additional sampling iterations beyond 100 did not give significantly different results. The rule counts from the chosen role assignments along with the priors are then used to calculate a MAP estimate of the model parameters. These parameters are then used to parse the test examples.

For large datasets, for example the Europarl dataset, the above sampling procedure can take a very long time. We implemented an approximate distributed training procedure for such datasets using the map-reduce framework. We begin with an initial set of samples. In the map phase, the dataset is divided into M parts and sampling is done on each part independently. The updated sample counts from each part are combined together in the reduce phase which serves as input to the next iteration. This method is an approximation because the sampling updates in a map process do not affect the sampling in the other maps processes. This approximation resulted in only an insignificant reduction in performance.

Each multinomial distribution has a symmetric dirichlet prior with a single concentration hyper-parameter. We place a gaussian hyper-prior with mean 1 and standard deviation 0.3 on all the hyper-parameters. The hyper-parameters are given an initial value of 1, and they are sampled according to their posterior distribution after each sampling iteration of the roles. The last sample values of the hyper-parameters are taken as the final values.

B.2 Sampling in Bilingual SRL Model

For sampling roles in the bilingual model, we need to consider the additional probabilities of roles being generated by the bilingual latent variables, as in equation 4.1. Let the superscript $(l : t)$ denotes the training example $\#t$ of language l. Equation B.2 shows the sampling of a role.

$$
\begin{aligned}
P(r_i^{(l:t)}|\mathbf{r}_{-i}^{(l:t)}, \mathbf{f}^{(l:t)}, p^{(l:t)}, vc^{(l:t)}, \mathbf{z}^{(t)}, D^{(-t)}) &\propto P(r_i^{(l:t)}, \mathbf{r}_{-i}^{(l:t)}, \mathbf{f}^{(l:t)}|\mathbf{z}^{(t)}, p^{(l:t)}, vc^{(l:t)}, D^{(-t)}) \\
&= \int_{\theta} P(r_i^{(l:t)}, \mathbf{r}_{-i}^{(l:t)}, \mathbf{f}^{(l:t)}|\theta, \mathbf{z}^{(t)}, p^{(l:t)}, vc^{(l:t)}) P(\theta|D^{(-t)}) \mathrm{d}\theta \\
&= \int_{\theta} P(r_i^{(l:t)}, \mathbf{r}_{-i}^{(l:t)}, \mathbf{f}^{(l:t)}|\theta, p^{(l:t)}, vc^{(l:t)}) \prod_{j,k:z_k^{(t)} \rightarrow r_j^{(l:t)}} P(r_j^{(l:t)}|\theta, z_k^{(t)}) P(\theta|D^{(-t)}) \mathrm{d}\theta
\end{aligned}
\tag{B.2}
$$

Similar to the monolingual sampling, the above equation is the expected value of the product of multinomial parameters under dirichlet distributions, which can be evaluated using a closed form solution.

For sampling bilingual latent variables, we need to consider three factors in equation 4.1: two factors corresponding to probabilities of generating the aligned roles in two languages, and the third one corresponding to selecting the bilingual latent variable according to CRP. Equation B.3 illustrates this sampling.

$$
\begin{aligned}
P(z_k^{(t)}|\mathbf{z}_{-k}^{(t)}, \mathbf{r}^{(l1:t)}, \mathbf{f}^{(l1:t)}, p^{(l1:t)}, vc^{(l1:t)}, \mathbf{r}^{(l2:t)}, \mathbf{f}^{(l2:t)}, p^{(l2:t)}, vc^{(l2:t)}, D^{(-t)}) \\
\propto P(r_i^{(l1:t)}|z_k^{(t)}, D^{(-t,k)}) P(r_j^{(l2:t)}|z_k^{(t)}, D^{(-t,k)}) P(z_k^{(t)}|D^{(-t,k)})
\end{aligned}
\tag{B.3}
$$

where $r_i^{(l1:t)}$ and $r_j^{(l2:t)}$ are the aligned roles connected to the bilingual latent variable $z_k^{(t)}$, and $D^{(-t,k)}$ denotes all the variables except $z_k^{(t)}$, $r_i^{(l1:t)}$ and $r_j^{(l2:t)}$.

Bibliography

O. Abend, R. Reichart, and A. Rappoport. 2009. Unsupervised argument identification for semantic role labeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 28–36. Association for Computational Linguistics.

Hagai Attias. 1999. Inferring parameters and structure of latent variable models by variational bayes. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 21–30. Morgan Kaufmann Publishers Inc.

C.F. Baker, C.J. Fillmore, and J.B. Lowe. 1998. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics.

J.K. Baker. 1979. Trainable grammars for speech recognition. *The Journal of the Acoustical Society of America*, 65:S132.

Markus Becker and Anette Frank. 2002. A stochastic topological parser for german. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

B. Bohnet. 2009. Efficient parsing of syntactic and semantic dependency structures. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, CoNLL '09, pages 67–72, Morristown, NJ, USA. Association for Computational Linguistics.

Claire Bonial, Jena Hwang, Julia Bonn, Kathryn Conger, Olga Babko-Malaya, and Martha Palmer. 2012. English propbank annotation guidelines. *Center for Computational Language and Education Research Institute of Cognitive Science University of Colorado at Boulder*.

Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen-Schirra, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans Uszkoreit. 2004. Tiger: Linguistic interpretation of a german corpus. *Research on Language and Computation*, 2(4):597–620.

Aljoscha Burchardt, Katrin Erk, Anette Frank, Andrea Kowalski, Sebastian Padó, and Manfred Pinkal. 2006. The salsa corpus: a german corpus resource for lexical semantics. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC-2006)*.

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

Lou Burnard. 2000. Reference guide for the british national corpus (world edition).

X. Carreras and L. Màrquez. 2004. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proceedings of CoNLL-2004*, pages 89–97.

X. Carreras and L. Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 152–164. Association for Computational Linguistics.

X. Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL*, pages 957–961.

Jackie Chi Kit Cheung and Gerald Penn. 2009. Topological field parsing of german. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 64–72. Association for Computational Linguistics.

Alexander Clark. 2003. Combining distributional and morphological information for part of speech induction. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1*, pages 59–66. Association for Computational Linguistics.

S.B. Cohen and N.A. Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 74–82. Association for Computational Linguistics.

R. Collobert and J. Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38.

P. Diderichsen. 1966. *Elementary Danish Grammar*. Gyldendal, Copenhagen.

E. Drach. 1937. *Grundstellung der Deutschen Satzlehre*. Diesterweg, Frankfurt.

J.M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th conference on Computational linguistics-Volume 1*, pages 340–345. Association for Computational Linguistics.

Jason Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. In *Advances in Probabilistic and Other Parsing Technologies*, pages 29–61. Springer.

W Nelson Francis and Henry Kucera. 1979. Brown corpus manual. *Brown University Department of Linguistics*.

H. Fürstenau and M. Lapata. 2009. Graph alignment for semi-supervised semantic role labeling. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 11–20. Association for Computational Linguistics.

N. Garg and J. Henderson. 2011. Temporal restricted boltzmann machines for dependency parsing. In *Proc. 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 11–17.

D. Gildea and D. Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia*.

T. Grenager and C.D. Manning. 2006. Unsupervised discovery of a statistical verb lexicon. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 1–8. Association for Computational Linguistics.

J. Hajič, M. Ciaramita, R. Johansson, D. Kawahara, M.A. Martí, L. Màrquez, A. Meyers, J. Nivre, S. Padó, J. Štěpánek, et al. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–18. Association for Computational Linguistics.

J. Hall, J. Nilsson, J. Nivre, G. Eryigit, B. Megyesi, M. Nilsson, and M. Saers. 2007. Single malt or blended? A study in multilingual parser optimization. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 933–939. Association for Computational Linguistics.

J. Henderson and I. Titov. 2010. Incremental sigmoid belief networks for grammar learning. *The Journal of Machine Learning Research*, 9999:3541–3570.

J. Henderson, P. Merlo, G. Musillo, and I. Titov. 2008. A latent variable model of synchronous parsing for syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 178–182. Association for Computational Linguistics.

James Henderson. 2003. Inducing history representations for broad coverage statistical parsing. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 24–31. Association for Computational Linguistics.

G.E. Hinton, S. Osindero, and Y.W. Teh. 2006. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

G.E. Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800.

Tilman N Höhle. 1983. Topologische felder. *Unpublished manuscript, University of Cologne.*

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics.

R. Hwa, P. Resnik, A. Weinberg, C. Cabezas, and O. Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering*, 11(03):311–325.

T. Jiang, L. Wang, and K. Zhang. 1995. Alignment of trees–an alternative to tree edit. *Theoretical Computer Science*, 143(1):137–148.

R. Johansson and P. Nugues. 2008. Dependency-based semantic role labeling of propbank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 69–78. Association for Computational Linguistics.

P. Kingsbury and M. Palmer. 2002. From treebank to propbank. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC-2002)*, pages 1989–1993. Citeseer.

D. Klein and C.D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 128–135. Association for Computational Linguistics.

D. Klein and C.D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 478. Association for Computational Linguistics.

P. Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5.

T. Koo and M. Collins. 2010. Efficient Third-order Dependency Parsers. *Proceedings of the 48th ACL, Association for Computational Linguistics*.

Peter CR Lane and James B Henderson. 2001. Incremental syntactic parsing of natural language corpora with simple synchrony networks. *Knowledge and Data Engineering, IEEE Transactions on*, 13(2):219–231.

J. Lang and M. Lapata. 2010. Unsupervised induction of semantic roles. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 939–947. Association for Computational Linguistics.

J. Lang and M. Lapata. 2011a. Unsupervised semantic role induction via split-merge clustering. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, Portland, Oregon.*

J. Lang and M. Lapata. 2011b. Unsupervised semantic role induction with graph partitioning. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1320–1331, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

D. Lin. 1998. An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning*, volume 1, pages 296–304.

Colin L Mallows. 1957. Non-null ranking models. i. *Biometrika*, pages 114–130.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

L. Màrquez, X. Carreras, K.C. Litkowski, and S. Stevenson. 2008. Semantic role labeling: an introduction to the special issue. *Computational linguistics*, 34(2):145–159.

R. McDonald and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*, volume 6.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530. Association for Computational Linguistics.

A. Meyers, R. Reeves, C. Macleod, R. Szekely, V. Zielinska, B. Young, and R. Grishman. 2004. The NomBank project: An interim report. In *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*, pages 24–31.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

G.A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K.J. Miller. 1990. Introduction to wordnet: An on-line lexical database. *International Journal of lexicography*, 3(4):235.

A. Mnih and G. Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning*, pages 641–648. ACM.

T. Naseem, B. Snyder, J. Eisenstein, and R. Barzilay. 2009. Multilingual part-of-speech tagging: Two unsupervised approaches. *Journal of Artificial Intelligence Research*, 36(1):341–385.

J. Nivre and R. McDonald. 2008. Integrating graph-based and transition-based dependency parsers. *Proceedings of ACL-08: HLT*, pages 950–958.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of english text. In *Proceedings of the 20th international conference on Computational Linguistics*, page 64. Association for Computational Linguistics.

J. Nivre, J. Hall, and J. Nilsson. 2004. Memory-based dependency parsing. In *Proceedings of CoNLL*, pages 49–56.

J. Nivre, J. Hall, and J. Nilsson. 2006a. MaltParser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC*, volume 6.

J. Nivre, J. Hall, J. Nilsson, G. EryiÇ§it, and S. Marinov. 2006b. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 221–225. Association for Computational Linguistics.

J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kubler, S. Marinov, and E. Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95.

F.J. Och and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51.

S. Padó and M. Lapata. 2009. Cross-lingual annotation projection for semantic roles. *Journal of Artificial Intelligence Research*, 36(1):307–340.

Martha Palmer, Joseph Rosenzweig, and Scott Cotton. 2001. Automatic predicate argument analysis of the penn treebank. In *Proceedings of the first international conference on Human language technology research*, pages 1–5. Association for Computational Linguistics.

M. Palmer, D. Gildea, and P. Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

J. Pitman. 2002. Combinatorial stochastic processes. Technical report, Technical Report 621, Dept. Statistics, UC Berkeley, 2002. Lecture notes for St. Flour course.

S. Pradhan, K. Hacioglu, V. Krugler, W. Ward, J.H. Martin, and D. Jurafsky. 2005. Support vector learning for semantic argument classification. *Machine Learning*, 60(1):11–39.

V. Punyakanok, D. Roth, W. Yih, and D. Zimak. 2004. Semantic role labeling via integer linear programming inference. In *Proceedings of the 20th international conference on Computational Linguistics*, page 1346. Association for Computational Linguistics.

R. Salakhutdinov and G. Hinton. 2009. Replicated softmax: an undirected topic model. *Advances in Neural Information Processing Systems*, 22.

R. Salakhutdinov, A. Mnih, and G. Hinton. 2007. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, page 798. ACM.

Karin Kipper Schuler. 2005. Verbnet: A broad-coverage, comprehensive verb lexicon.

B. Snyder, T. Naseem, and R. Barzilay. 2009a. Unsupervised multilingual grammar induction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 73–81. Association for Computational Linguistics.

B. Snyder, T. Naseem, J. Eisenstein, and R. Barzilay. 2009b. Adding more languages improves unsupervised multilingual part-of-speech tagging: A Bayesian non-parametric approach. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 83–91. Association for Computational Linguistics.

Richard Socher, Cliff C Lin, Andrew Y Ng, and Christopher D Manning. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, volume 2.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013. Parsing with compositional vector grammars. In *In Proceedings of the ACL conference*. Association for Computational Linguistics.

M. Surdeanu and C.D. Manning. 2010. Ensemble models for dependency parsing: cheap and good? In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 649–652. Association for Computational Linguistics.

M. Surdeanu, R. Johansson, A. Meyers, L. Màrquez, and J. Nivre. 2008. The conll-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177. Association for Computational Linguistics.

I. Sutskever and G.E. Hinton. 2007. Learning multilevel distributed representations for high-dimensional sequences. In *Proceeding of the Eleventh International Conference on Artificial Intelligence and Statistics*, pages 544–551.

R. Swier and S. Stevenson. 2004. Unsupervised semantic role labelling. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 95–102.

Akihiro Tamura, Taro Watanabe, and Eiichiro Sumita. 2014. Recurrent neural networks for word alignment model. In *In Proceedings of the ACL conference*. Association for Computational Linguistics.

G.W. Taylor, G.E. Hinton, and S.T. Roweis. 2007. Modeling human motion using binary latent variables. *Advances in neural information processing systems*, 19:1345.

C. Thompson, R. Levy, and C. Manning. 2003. A generative model for semantic role labeling. *Machine Learning: ECML 2003*, pages 397–408.

I. Titov and J. Henderson. 2007a. Constituent parsing with incremental sigmoid belief networks. In *Proceedings of the 45th Annual Meeting on Association for Computational Linguistics*, volume 45, page 632.

I. Titov and J. Henderson. 2007b. Fast and robust multilingual dependency parsing with a generative latent variable model. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL*, pages 947–951.

Ivan Titov and James Henderson. 2007c. A latent variable model for generative dependency parsing. *IWPT*, page 144.

I. Titov and A. Klementiev. 2012a. Crosslingual induction of semantic roles. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Ivan Titov and Alexandre Klementiev. 2012b. A bayesian approach to unsupervised semantic role induction. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 12–22. Association for Computational Linguistics.

K. Toutanova, A. Haghighi, and C.D. Manning. 2008. A global joint model for semantic role labeling. *Computational Linguistics*, 34(2):161–191.

Lonneke Van der Plas, Paola Merlo, and James Henderson. 2011. Scaling up automatic cross-lingual semantic role annotation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 299–304. Association for Computational Linguistics.

N. Xue and M. Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of EMNLP*, volume 2004, pages 88–94.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*, volume 3.

Will Y Zou, Richard Socher, Daniel M Cer, and Christopher D Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *EMNLP*, pages 1393–1398.