© 2014 Gourab Kundu

DOMAIN ADAPTATION WITH MINIMAL TRAINING

BY

GOURAB KUNDU

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2014

Urbana, Illinois

Doctoral Committee:

Professor Dan Roth, Chair
Professor Chengxiang Zhai
Associate Professor Julia Hockenmaier
Associate Professor Hal Daumé III, University of Maryland

Abstract

The performance of a machine learning model trained on labeled data of a (source) domain degrades severely when they are tested on a different (target) domain. Traditional approaches deal with this problem by training a new model for every target domain. In natural language processing, top performing systems often use multiple interconnected models; therefore training all of them for every target domain is computationally expensive. Moreover, retraining the model for the target domain requires access to the labeled data from the source domain which may not be available to end users due to copyright issues. This thesis is a study on how to adapt to a target domain, using the system trained on source domain and avoiding the cost of retraining and the need for access to the source labeled data.

This thesis identifies two key ingredients for adaptation without training: broad coverage resources and constraints.

We show how resources like Wikipedia, VerbNet and WordNet that contain comprehensive coverage of entities, semantic roles and words in English can help a model adapt to the target domain. For the task of semantic role labeling, we show that in the decision phase, we can replace a linguistic unit (e.g. verb, word) with another equivalent linguistic unit residing in the same cluster defined in these resources (e.g. VerbNet, WordNet) such that after replacement, text becomes more like text on which the model was trained. We show that the model's output is more accurate on the transformed text than on original text. In another instance, we show how to use a system for linking mentions to Wikipedia concepts for adaptation of a named entity recognition system. Since Wikipedia has a broad domain coverage, the linking system is robust across domain variations. Therefore, jointly performing entity recognition and linking improves the accuracy of entity recognition on the target domain without requiring training of a new system for the new domain. In all cases, we show how to use intuitive constraints to guide the model into making coherent predictions.

We show how incorporating prior knowledge about a new domain as declarative constraints into the decision phase can improve performance of a model on the new domain. When such prior knowledge is unavailable, we show how to acquire knowledge automatically from unlabeled text from the new domain and domains similar to both source and target domains.

To my family.

Acknowledgments

First and foremost, I would like to thank my parents, my wife and my sister for encouraging me to embark on the grand mission of obtaining a PhD degree and then providing me all possible forms of support in every step of this mission. Specially I would like to thank my mother whose encouragement and support are behind every success of my life. My new born son also helped me in his own way by making me smile at difficult times.

I always consider myself fortunate that Prof. Dan Roth was my research advisor. He is an outstanding advisor, researcher and teacher. I truly learned a lot from him , even outside Natural Language Processing and Machine Learning. I learned from him how to think critically and how to present my ideas orally and in writing. Dan was always patient and at the same time, he always encouraged me to strive for better results. These lessons will be extremely valuable for me in my future life.

I want to thank my committee members Prof. Zhai, Prof. Daumé and Prof. Hockenmaier. I have got valuable feedback from them regarding my research and presentations.

I would like to thank the past and present members of the Cognitive Computation Group. Mark Sammons always provided the support in terms of infrastructure that was essential for all my projects. Ming-wei Chang and Vivek Srikumar were my collaborators in several of my early projects and I learned from them the methodology of research. I would also like to thank Kai-Wei Chang, James Clarke, Michael Connor, Christos Christodoupoulos, Quang Do, Prateek Jindal, Parisa Kordjamshidi, Daniel Khasabi, Wei Lu, Stephen Mayhew, Jeff Pasternack, Haoruo Peng, Lev Ratinov, Nick Rizzolo, Alla Rozovskaya, Subhro Roy, Rajhans Samdhani, Yangqiu Song, Chen-Tse Tai, Ivan

Titov, Yuancheng Tu, Shyam Upadhyay, Vinod Vydiswaran, John Weiting and Ruichen Wang. I would like to give special thanks to Eric Horn for his corrections of the grammatical mistakes of my thesis. I would also like to thank Dawn Cheek for her help in room reservations.

Finally, I would like to thank all my friends from the Bangladeshi community in Champaign-Urbana. The time I spent with them in laughters and joys helped me to overcome the stress of work for this dissertation.

Table of Contents

List of Tables . . . . . **x**

List of Figures . . . . . **xii**

List of Abbreviations . . . . . **xiii**

Chapter 1 Introduction **1**
1.1 Statistical Learning in NLP 1
1.2 The Need for Domain Adaptation 3
1.3 Contributions of this Thesis 5
1.4 Thesis Outline 6

Chapter 2 Basic Terminology **9**
2.1 Introduction 9
2.2 Classification 9
2.3 Learning Protocols 9
2.3.1 Supervised Learning 10
2.3.2 Unsupervised Learning 10
2.3.3 Semi-supervised Learning 11
2.4 Linear Models 11
2.5 Binary and Multiclass Classification 12
2.6 Structured Prediction 13
2.6.1 Constrained Conditional Models 14
2.7 Summary 16

Chapter 3 Related Work **17**
3.1 Introduction 17
3.2 Adaptation w/ Labeled Data in Target 17
3.2.1 Source-as-Prior Based Methods 18
3.2.2 General Distribution Based Methods 19
3.2.3 Instance Weighting Based Methods 20
3.2.4 Support Vector Based Methods 23
3.2.5 Relation to Current Thesis 25
3.3 Adaptation without Labeled Data in Target 25
3.3.1 Theoretical Studies 26
3.3.2 Bootstrapping Based Methods 27
3.3.3 Representation Learning Based Methods 29

3.3.4 Relation to Current Thesis . . . 34
3.4 Summary . . . 35

Chapter 4 Adaptation Using Transformation . . . 36
4.1 Introduction . . . 36
4.2 Problem Formulation . . . 38
4.3 Motivating Examples . . . 38
4.4 Transformation Functions . . . 40
4.4.1 Transformation From List . . . 41
4.4.2 Learned Transformations . . . 45
4.5 Joint Inference . . . 50
4.6 Experimental Results . . . 51
4.7 Summary . . . 54

Chapter 5 Prior Knowledge Driven Adaptation . . . 55
5.1 Introduction . . . 55
5.2 Tasks and Datasets . . . 56
5.3 Motivating Examples . . . 58
5.4 Inference w/ Prior Knowledge . . . 60
5.5 Self-training w/ Prior Knowledge . . . 60
5.6 Experimental Results . . . 62
5.7 Summary . . . 63

Chapter 6 Adaptation Using Learned Constraints . . . 64
6.1 Introduction . . . 64
6.2 Motivating Examples . . . 66
6.3 Algorithm for Learning Constraints . . . 68
6.3.1 Intermediate Domain Constraints . . . 68
6.3.2 Target Domain Constraints . . . 69
6.3.3 Prediction . . . 74
6.4 Experimental Results . . . 76
6.4.1 Gene Recognition in the Biomedical Domain . . . 76
6.4.2 Entity Recognition in News Domain . . . 78
6.5 Summary . . . 81

Chapter 7 Adaptation Using Wikipedia . . . 82
7.1 Introduction . . . 82
7.2 Motivating Examples . . . 85
7.3 Formulation of Joint Inference of NER and Wikifier . . . 86
7.4 Experimental Results . . . 88
7.4.1 In-domain Experiments . . . 89
7.4.2 Domain Adaptation Experiments . . . 92
7.5 Summary . . . 95

Chapter 8 Adaptation using Small Training on Target 96
8.1 Introduction 96
8.2 Motivating Example 98
8.3 Algorithm 99
8.4 Experimental Results 101
8.5 Summary 104

Chapter 9 Conclusion 105
9.1 Future Research Directions 106
9.1.1 Extension of the Proposed Methods to Labeled Adaptation Scenario 106
9.1.2 Joint Inference of NLP Tasks 106
9.1.3 Domain Robust Models 107

References 108

List of Tables

4.1 Examples of Simplifications (Predicate is run) . . . 44
4.2 Comparing single parse system on Brown. . . . 52
4.3 Comparison of the multi parse system on Brown. . . . 52
4.4 Ablation Study for ADUT-Charniak . . . 53
4.5 Performance on Infrequent Verbs for the Transformation of *Predicate Replacement* . 53

5.1 Results (F1) of Baseline model of POS Tagging . . . 58
5.2 Results (F1) of Baseline model of SRL . . . 58
5.3 Comparsion of results(F1) for the Baseline Model versus PDA-KW . . . 60
5.4 Comparsion of results(F1) of different ways of incorporating knowledge . . . 62
5.5 Comparsion of results(F1) for POS Tagging with Jiang and Zhai (2007a) . . . 63

6.1 The upper part of the table shows target domain text with gold labels. The lower part of the table shows the prediction of NER trained from news text Only entities of type person are shown in bold. . . . 68
6.2 Adaptation results on Gene Recognition Task, FD05= (Finkel et al., 2005a) and HY09= Huang and Yates (2009) . . . 77
6.3 Experimental results on ACE 2005 data set, Wu09= Wu et al. (2009), RR09= Ratinov and Roth (2009) . . . 79
6.4 Experimental results on CoNLL to Enron Adaptation, RR09= Ratinov and Roth (2009), HY09= Huang and Yates (2009), M05= Minkov et al. (2005),CCAM^{-w}=CCAM without using the word level constraints from Section 6.3.1 . . . 79

7.1 Definition of Tasks for NER, Wikifier, (Sil and Yates, 2013) and Our system . . . 84
7.2 Confusion set of Wikifier and NER for the mention *England*. The gold label for Wikification & NER for England are respectively `http://en.wikipedia.org/wiki/England_cricket_team` and ORG . . . 86
7.3 Disjoint NER, Joint NER, Disjoint Wikifier and Joint Wikifier F1 on CoNLL test set using α=0.9 . . . 91
7.4 Text snippet from source and target domains . . . 93
7.5 Named entity annotation (gold & predicted by the NER trained on source labeled data) for target domain text . . . 93
7.6 Wikipedia URLs predicted for mentions in the target domain text by the Wikifier and the types listed in freebase for those URLs . . . 93
7.7 In-domain$^{\circ}$ & Out-domain* experimental results for NER . . . 94

8.1 Text snippets from the target (email) domain, predictions of an NER (trained on source (news)) on this text and corresponding gold NER annotations. Each underlined text segment represents a mention either predicted by the NER model or from gold annotations. The corresponding predicted entity type and the gold entity type are shown as subscript. The portion of text where the NER predictions did not match the gold NER annotations has been shown in bold. 98
8.2 F1 score on two email data sets (emails related to meetings and randomly selected emails) and training time for one email data set (meetings). The source domain is news wire. In terms of accuracy, FAJIST outperforms both source only and target only models. It also outperforms a state-of-the-art adaptation method Huang and Yates (2009) (referred to as HY09 in table). In terms of training time, FAJIST has a speedup of 16x compared to Huang and Yates (2009). 101
8.3 Comparison of F1 scores and time required by the FAJIST with supervised adaptation methods. concat refers to retraining the model of Ratinov and Roth (2009) on the concatenation of the source and *selected* target labeled instances. Daumé III (2007) refers to the method of Daumé III (2007) trained over both the source and *selected* target labeled instances. Firstly, FAJIST outperforms both these approaches on the meetings data set and performs comparable on the random data set. FAJIST does not perform any training over the source labeled data and so it is 8x times faster than other approaches. Secondly, the supervised adaptation methods (concat & Daumé III (2007)) both outperform the state-of-the-art unsupervised adaptation method of Huang and Yates (2009) (referred to as HY09) 103

List of Figures

4.1 Original Parse tree 39
4.2 Corrected Parse tree after replacement of unknown word *checkup* by *examination* . . 39
4.3 Parse Tree for Sentence *Mr. X was entitled to a discount* 46

7.1 Variation of F1 scores of Joint NER, Joint Wikifier, Disjoint NER and Disjoint Wikifier (y axis) on the CoNLL Dev Set as α is varied (x axis) 90

List of Abbreviations

ILP	Integer Linear Programming
NLP	Natural Language Processing
POS	Part-of-Speech Tagging
SRL	Semantic Role Labeling
NER	Named Entity Recognition
SVM	Support Vector Machine
CCM	Constrained Conditional Model
WSJ	Wall Street Journal

Chapter 1

Introduction

This chapter presents an overview of the thesis. It presents a broad picture of the state-of-the-art in statistical learning for natural language processing, describes current challenges and gives an overview of how this thesis proposes to overcome these challenges. It also outlines the scope of the thesis and the potential impact of the methods developed in the context of this thesis.

1.1 Statistical Learning in NLP

Comprehension of natural language is a trivial task for humans, but automatic processing of language by computers has been a stiff challenge, because natural language is inherently ambiguous. For example, in some contexts, *Chicago* can refer to the city, whereas in other contexts, *Chicago* can refer to a band or a font. Understanding natural language for computers means resolving all such ambiguities in text, extracting the key facts stated in the text, organizing the collected facts and linking them to previously accumulated facts and reasoning about the facts to discover missing facts. Successful natural language processing can lead to applications like automated natural language based customer support systems, information extraction from the web, question answering and others.

Machine learning based methods are now *de-facto* standard for natural language processing. Machine learning builds models for a natural language task based on some form of supervision in terms of labeling of instances, and uses it on unseen text to annotate text according to the specifications of the task. Two important tasks addressed in this thesis are named entity recognition and semantic role labeling. These two tasks are critical for natural language understanding. We will use these two tasks and in this thesis, we have proposed methods for improving the coverage of machine learning based systems for these tasks across a wide range of text genres.

Named Entity Recognition In language, we often express facts about entities of interest. For example, consider the following fragment of text

> President Obama addressed the United Nations Security Council yesterday in New York.

In this sentence, the key entities of interest are *Obama*, *United Nations Security Council* and *New York*. A computer program needs to recognize the text segments that represent these entities and classify them into types like Person (Obama), Organization (United Nations Security Council) and Location (New York). Once this set of concepts has been recognized, it can ground them to the list of entities in its data base. So recognizing text segments representing entities and their types is one of the fundamental tasks for natural language understanding.

To build a system for predicting named entities from unseen text, we use machine learning to build models from some form of supervision (e.g. annotated data, declarative knowledge). Many variations of models are possible, but the simplest model will do a prediction based on every word. For each word, the system will extract features like *previous word*, *next word* etc and learn weights for these features by taking statistics over some annotated data. Given a new sentence, for every new word, the system will predict the entity label that achieves the highest score where score is the sum of weights of features extracted for that word. For example, the features *previous word = President* and *current word = capitalized* are strong indicators for the next word being a person. So the model will learn high weights for these features. Given a new sentence

> President Nixon was involved in the Watergate Scandal.

the model can predict the label or type of person for *Nixon*, even without seeing it before in annotated data, because this word will generate high scoring features for person, i.e., *previous word = President*, *current word = capitalized*.

Semantic Role Labeling In natural language, a key part in a sentence is actually the *verb*. The verb or predicate refers to / describes an action performed by the agents in the sentence. Therefore, recognizing the arguments of a verb and categorizing them can give us a form of meaning representation of the sentence. Again consider the text presented above

President Obama addressed the United Nations Security Council yesterday in New York.

In this sentence, the action described is *address* or give a speech, the agent of this action is *Obama*, and the patient is *United Nations Security Council*. The sentence also expresses the fact that the time of this action was *yesterday* and the place of this action was *New York*. If we can build an automatic computer program to detect the boundaries of arguments of verbs, we can automatically extract the facts expressed in the text and store them in an ontology after linking to existing facts and later retrieve these facts in response to a query.

As before, we can train a machine learning based model from text annotated with the semantic roles of verbs. Many variations of the model are possible but a simple model will consider every possible segment of the sentence and classify it as *agent* or *patient* or *time* or *place* argument with respect to each verb in the sentence. Useful features here can be *location of the argument with respect to the verb* because for the verb *address*, often arguments of type *agent* will precede the verb and arguments of type *patient* will follow the verb. Richer features based on named entity annotation of the sentence e.g. will also be useful since the model can learn that features like *current word is a named entity of type person*, *current word is a named entity of type organization* are strong indicators of current word being labeled as respectively, the agent and patient of the verb *address*.

1.2 The Need for Domain Adaptation

Statistical learning based methods have been very successful in natural language processing. For example, machine learning based named entity recognition systems achieved 94% F1 on MUC-7 data sets. The performance achieved by a human on this task is about 97%. Although there still exists some gap between the performance of the machine and human performance in several NLP tasks specially parsing, semantic role labeling etc, the developed models are already practically useful. Unfortunately, we are not done yet. A fundamental assumption in machine learning is that to build a model from some annotated data (training data) such that the model can make accurate predictions on unseen text, unseen text has to come from the same distribution as the training

data. The previously reported performance of named entity recognition systems was obtained by training the model on annotated text which is similar to the text on which the model would be evaluated on. Instead, if we train a model on training data from Reuters and test it on text from the Wall Street Journal, the performance drops from 90% to 65% (Ciaramita and Altun, 2005). If we train a semantic role labeling system on annotated text from the Wall Street Journal and test it on text from the literature domain, the performance drops from 76% to 65% (Kundu and Roth, 2011). We will discuss the reasons for this very significant drop below.

Named Entity Recognition Adaptation Entities can vary considerably from domain to domain, both in terms of surface forms and in terms of contextual evidence. For example, consider the following text from the email domain

> Subject: Meeting
>
> meeting today w/ Dan & Mckeough & Stephen

In this email, there are three named entities of type *Person* (Dan, Mckeough, Stephen). Note that this email text does not follow the formal written English convention and contains terms like *Subject:* and *Mckeough* which are not very frequent in the news domain. If we train the named entity recognition system from annotated newswire data, the system does not encounter such phenomenon in training data and therefore, cannot make accurate predictions on the new text. For example, a state-of-the-art named entity recognition system might predict *Subject:* as one named entity because this word is capitalized and was not seen as a non-named entity in training data. It might also fail to predict *Mckeough* as a person name because this name was unseen in the training data and also the context of this name was not encountered in training data before. The trained NER on newswire has learned that contexts like "President" and "said" are important indicators of named entities but the context of *Mckeough* (previous and next word are '&') never appeared in training data. So the trained NER does not know if this context implies any named entity or not. In the absence of any strong evidence, it might fail to predict *Mckeough* as a named entity.

Semantic Role Labeling Adaptation Semantic role labeling systems trained on a domain will perform very poorly on other domains. Consider the following text

Scotty gazed at ugly gray slumps.

The gold labeling with respect to the predicate *gaze* is that *Scotty* is the agent and *at ugly gray slums* is the patient. However, a SRL system trained on newswire predicts *Scotty* as the agent and *at ugly gray slums* as the location type argument for predicate *gaze*. The reason for this is that the predicate *gaze* is very infrequent in the training data of newswire and therefore the SRL system makes mistakes on predicting the types of the arguments of this verb.

We have shown examples of problems that difference of domains causes for two tasks in natural language processing, namely named entity recognition and semantic role labeling. However, this problem of domain adaptation is present for almost all tasks in natural language processing. Since natural language varies considerably across different genres, and annotating data from each genre and training a separate system for each genre is not feasible, how to adapt a model to a domain different than the training domain is one of the critical questions facing the natural language processing research community.

1.3 Contributions of this Thesis

In this thesis, we investigate if it is possible to adapt to a new domain without retraining the model trained on annotated text of a different domain. In particular, we make the following claims in this thesis

> *Adaptation of a model to a new domain is possible, without training a new model for the new domain. Many existing resources (Wikipedia, WordNet) provide coverage for a broad range of domains. We can extract knowledge from these resources and incorporate it in the decision phase to reduce the domain gap. We can also automatically extract knowledge from target domain unlabeled data and target domain annotation manuals and incorporate them in the decision phase with the same old model. This paradigm of adaptation without training will make application of natural language processing tools consisting of multiple interconnected models across different domains efficient and flexible.*

Below, I outline the primary contributions of this thesis

- We propose *ADUT (ADaptation Using label-preserving Transformation)*, an approach that avoids the need for retraining and does not require knowledge of the new domain, or any data from it. Our approach applies simple label-preserving transformations to the target text so that the transformed text is more similar to the training domain; it then applies the existing model on the transformed sentences and combines the predictions to produce the desired prediction on the target text.

- We propose a framework called Prior knowledge Driven Adaptation (PDA), which takes advantage of knowledge of the target domain to adapt the model. We empirically study the effects of prior knowledge and different ways of incorporating it.

- We propose a principled and effective method for adaptation to a new domain via an inference process that makes use of the source trained model along with constraints mined in an unsupervised way from large amounts of text, including the test domain, to bias the model decision. We show that this process significantly increases the accuracy of the original source model on new domain, without modifying it at all.

- We propose a method for domain adaptation of a named entity recognition system, without training a new model for the new domain. Again, this is accomplished via a joint inference process with an entity linking system to Wikipedia, which is robust across domain variations due to the broad domain nature of Wikipedia.

1.4 Thesis Outline

The organization of the rest of the thesis is described below

Background Starting from the next chapter, we review some key background concepts needed to understand the methods developed in this thesis and to put the methods in context with contemporary approaches in literature.

- Chapter 2 provides the necessary machine learning background and introduces the key machine learning concepts used throughout this thesis.

- Chapter 3 gives an overview of prior research in domain adaptation. This chapter categorizes the different scenarios under which the problem of domain adaptation appears and groups the proposed methods in terms of methodology and framework and compares and contrasts them.

Adaptation with Minimal training This section presents in-depth analysis of the methods proposed in this thesis. A reader familiar with the basic machine learning concepts and domain adaptation research literature can simply start reading from this section.

- Chapter 4 proposes the first method developed in the paradigm of domain adaptation without retraining. This method is based on transformation of text from the target domain to look similar to text from the source domain.

- Chapter 5 presents an approach which uses prior knowledge about the target domain for adaptation. We show that prior knowledge is often cheap to obtain and using it can provide significant improvements.

- Chapter 6 presents an approach which learns constraints automatically from intermediary corpora and unlabeled corpora from the target domain and then use these constraints for effective adaptation to the target domain.

- Chapter 7 presents an approach using joint inference across tasks to facilitate the adaptation of the models for those tasks. We show that if we have two models, one relatively weak across domain variations and another robust across domain variations, joint inference can help the adaptation of the weaker model.

- Chapter 8 presents an approach for adaptation by training a simple model for target domain using only target domain labeled data (automatically generated by source model) and do joint inference with the model from the source domain. We show that this gives significant performance improvement in accuracy while training time is kept to a minimum.

Conclusion This is the concluding part of the thesis.

- Chapter 9 concludes the thesis by outlining the limitations of the proposed approaches and the future directions of research.

Chapter 2

Basic Terminology

2.1 Introduction

Statistical machine learning based methods are now very popular in natural language processing. This thesis proposes methods for making these statistical models generalizable beyond the domain of the training data used for training these models. This chapter provides the necessary machine learning backgrounds for this thesis.

2.2 Classification

The kind of machine learning models we are interested in this thesis is classification models. Given an input space $\mathcal{X}$ and discrete output space $\mathcal{Y}$, the goal is to learn a mapping / function $f : \mathcal{X} \rightarrow \mathcal{Y}$. An example of this scenario is the task of document classification.

Assume we have documents from two categories: sports and politics. Each document can be represented as a bag of words appearing in that document. This means that each document will be represented as a vector $x \in \{0, 1\}^V$, where V is the size of the vocabulary. $x_i = 1$ if word i appears in the document, 0 otherwise. So the input space $\mathcal{X}$ is $\{0, 1\}^V$ and $\mathcal{Y}$ is {politics, sports}. The goal is to learn a function $f : \mathcal{X} \rightarrow \mathcal{Y}$ that can output a category {politics, sports} given an input document.

Machine learning is concerned with how to train this function f. In the next section, we will discuss several scenarios for training this function f.

2.3 Learning Protocols

There are three dominant learning protocols in machine learning. They are discussed below.

2.3.1 Supervised Learning

We can randomly collect some instances / examples (e.g. documents) and manually label them with categories (e.g. politics / sports). The examples correspond to samples drawn from the distribution over the example space D. The goal is to learn a function f from this set of labeled examples so that it performs well on unseen examples.

$$f^* = \arg\min_f E_{x \backsim D} L(f(x), y) \tag{2.1}$$

Here $f(x)$ is the prediction made by the function f on example x and y is the gold label for example x. L denotes a loss function that is non-zero when $f(x)$ does not equal y. Since we do not typically know the distribution D, we cannot directly minimize according to Equation 2.1. Instead, we measure the loss on the manually prepared set of (example, label) pairs $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$.

$$f^* = \arg\min_f \sum_{n=1}^{N} L(f(x_n), y_n) \tag{2.2}$$

Minimizing loss according to Equation 2.2 is known as *Empirical Risk Minimization*. However, ERM often leads to overfitting of the training data. Therefore, it is essential to include a *regularization term* that penalizes extremely large values in f and therefore tries to achieve an f that can perform well on training examples but is still reasonably simple.

$$f^* = \arg\min_f \sum_{n=1}^{N} L(f(x_n), y_n) + \lambda \|f\|^2 \tag{2.3}$$

Minimizing loss according to Equation 2.3 is known as *Structural Risk Minimization*.

2.3.2 Unsupervised Learning

In this protocol of learning, no manually annotated example is available. What is available is simply the examples drawn from the example space $\mathcal{X}$. The goal here is to find some hidden structure in the data that can help end users understand the intrinsic properties of the data. In many cases, this structure can also correlate to the label of a classification task that we are trying to solve. Often the intended structure in the data is simply a clustering of the data. A very popular

clustering algorithm is K-means clustering. In this algorithm, K is the number of clusters and is provided by the user. The algorithm starts by randomly selecting K data points as cluster centers. It then assigns each point to the nearest cluster center. Based on this assignment, K new cluster centers are chosen and this process iterates repeatedly until convergence. Other popular forms of clusterings include hidden markov models, mixture models etc.

2.3.3 Semi-supervised Learning

In semi-supervised learning, there is a small set of labeled data available but we also want to leverage a large amount of unlabeled data. Similar to unsupervised learning, the hope is that by using large amounts of unlabeled data, we can discover meaningful structures from data that can help us in learning the classification task at hand. One explanation for the success of semi-supervised learning is that if there is any underlying clustering in the data, we need to learn a classifier that maximizes separation across these clusters. This might result in a classifier closer to the true target function.

Many different frameworks for semi-supervised learning have been proposed in the literature. They include mixture model based approaches, self-training, co-training or multi-view training based methods and graph based methods. For a detailed introduction to this area, please refer to (Chapelle et al., 2006).

2.4 Linear Models

To find a function f for accurately predicting classes on the training data, we first need to define a hypothesis class within which we search for the best hypothesis that can explain the data. This hypothesis class can be class of decision trees, linear functions, quadratic functions or other non-linear functions for example neural networks. The most popular choice in NLP these days is the hypothesis class of linear functions. Given an input (word / sentence) x, we first create a feature vector $\phi(x)$ and learn a weight vector w_y for every label y. The score for each label is then simply calculated as the dot product.

$$y^* = \arg\max_{y \in \mathcal{Y}} w_y^T \cdot \phi(x) \tag{2.4}$$

An alternate way to express the previous equation is to consider a feature vector over the example and the label $\phi(x, y)$ and have a single weight vector. The optimal prediction is then calculated as

$$y^* = \arg\max_{y \in \mathcal{Y}} w^T \cdot \phi(x, y) \tag{2.5}$$

Here, $\phi(x, y)$ is the concatenation of $|\mathcal{Y}|$ vectors, where only the vector at index corresponding to label y is $\phi(x)$, and all others are zero vectors. Similarly, w is the concatenation of weight vectors for each label, where the concatenation order is the same as in $\phi(x, y)$.

2.5 Binary and Multiclass Classification

In binary prediction, the output space $\mathcal{Y}$ consists of only two labels. For binary classification, we need to learn a weight vector w such that for example x, the decision rule is given by

$$y = \begin{cases} 1 & \text{if } w^T \cdot \phi(x) > 0 \\ 0 & \text{if } w^T \cdot \phi(x) < 0 \end{cases}$$

Binary classification is the simplest case of classification. An example of a binary classification task is categorization of documents into two categories: politics or others.

In natural language processing, most problems are actually multi-class problems. An example of a multi-class problem is: predicting the named entity tag for each word where there are four possible entity tags (Person, Location, Organization or Miscellaneous Named Entity). There are two common strategies for reducing multi-class problems into binary problems. The first strategy, *One vs All*, learns a classifier for every class by treating the examples of that class as positive examples and examples of all other classes as negative examples. After the classifiers $\{w_y\}$ are learned, the prediction is made by

$$y^* = \arg\max_{y \in \mathcal{Y}} w_y^T \cdot \phi(x) \tag{2.6}$$

In the second strategy of *All vs All*, one weight vector is learned for every class pair {y,y'}, treating all examples of class y as positive and treating all examples of class y' as negative. In this case,

the final prediction is made by

$$y^* = \arg\max_{y \in \mathcal{Y}} \sum_{y' \in \mathcal{Y}} w_{yy'}^T \cdot \phi(x) \tag{2.7}$$

2.6 Structured Prediction

In natural language processing, often the output space is a set of structures instead of a set of labels. For example, for the task of part-of-speech tagging, the input space is the space of all possible sentences and output space is the space of all possible POS tag sequences. Since the number of possible POS tag sequences is exponential in the size of the sentence, the techniques for the multi-class classification can not be applied. Moreover, the structure consists of parts and the label of each part depends on the label of other parts. For example, for part-of-speech tagging, the POS tag of a word depends on the POS tag of nearby words. Therefore, structured prediction methods has been an area of active research recently. Some of the well known structured prediction algorithms are structured perceptron (Collins, 2002), conditional random fields (Lafferty et al., 2001) and structured SVMs (Tsochantaridis et al., 2004). While the exact formulations of these methods vary, almost all these methods need to solve one prediction problem per instance during training. The prediction problem for these methods can be written as

$$\mathbf{y}^* = \arg\max_{\mathbf{y} \in \mathcal{Y}} w^T \phi(\mathbf{x}, \mathbf{y}) \tag{2.8}$$

Here $\mathcal{Y}$ is the space of structures and $\phi(\mathbf{x}, \mathbf{y})$ is the feature functions defined over both input x and output structure $\mathbf{y}$. For example, for part-of-speech tagging of an input sentence of length 10, $\mathbf{x}$ is the sentence, $\mathbf{y}$ is a POS tag assignment $\{y_1, y_2, \ldots, y_{10}\}$. $w^T \cdot \phi(\mathbf{x}, \mathbf{y})$ is the score assigned by the model w for structure $\mathbf{y}$. Solving this problem will be intractable if we do not assume some decomposition of the feature function $\phi(x, \mathbf{y})$. It is common to make a Markov assumption, for example, the POS tag of a word at index i can only depend on tags assigned to words at index $i-1$ to $i-k$ where k is the Markov order. This assumption allows us to write efficient dynamic programming algorithms to solve this prediction problem.

The previous paradigm can be thought of as training *global* models since the predictors are

trained to maximize the score for the correct structures. There is also a paradigm of *local* models where we can train classifiers for parts of the structures only. For example, instead of training a predictor that maximizes the score of the POS tag sequence for the entire sentence, we can train a predictor that maximizes the score of the POS tag for every token in the sentence. Local classifiers are myopic but they are easy to train. In the POS tagging case, the local classifier can be simply a multi-class classifier.

However, as noted before, often the label of a part of sentence influences the label of another part of the sentence. For example, in the POS tagging case, an article cannot follow a noun. So in the paradigm of local classifiers, training is performed disregarding this joint structure, but during prediction time or test time, the coherence of different parts of the structure is taken into account. Constrained conditional models (Chang et al., 2008) is one of the popular formulations for ensuring the coherent prediction of structures.

2.6.1 Constrained Conditional Models

In many domains including the domain of natural language processing, often a subset of the space of output structures is legitimate. Based on the task, there are some constraints that must be satisfied for a output structure to be legitimate. For example, in part-of-speech tagging, a legitimacy constraint is that an article cannot be followed by a noun. For semantic role labeling, each verb can have at most one semantic role corresponding to the agent of that verb. In constrained conditional models, such declarative knowledge is incorporated as a constraint in the prediction phase. The score of an output structure is the sum of the scores of the constituent parts of the structure. Out of all structures that satisfy the prior knowledge derived constraints, constrained conditional model selects the highest scoring structure when the score of a structure is the sum of the scores of the parts of the structures as scored by the local model. Mathematically,

$$\mathbf{y}^* = \arg\max_{\mathbf{y} \in \mathcal{Y}'} \sum_{i=1}^{|\mathbf{x}|} w^T \cdot \phi(\mathbf{x}_i, \mathbf{y}_i) \tag{2.9}$$

Here $\mathcal{Y}' \subseteq \mathcal{Y}$ is the set of valid structures, i.e., structures that satisfy certain set of constraints C. Each constraint $c \in C$ is a boolean function $c : \mathcal{X} \times \mathcal{Y} \rightarrow \{0, 1\}$ that maps each joint assignment to

$\mathcal{X}$ and $\mathcal{Y}$ to either 0 or 1. So we can write,

$$\mathcal{Y}' = \{\mathbf{y} \in \mathcal{Y} : c(\mathbf{x}, \mathbf{y}) = 1\} \tag{2.10}$$

These constraints can be thought of as hard constraints since they must be satisfied by every legitimate structure.

An alternate characterization of Equation 2.9 is given by

$$\mathbf{y}^* = \arg\max_{\mathbf{y} \in \mathcal{Y}} \sum_{i=1}^{|\mathbf{x}|} w^T \phi(\mathbf{x}_i, \mathbf{y}_i) + \sum_{j=1}^{|C|} \rho_j * C_j(\mathbf{x}, \mathbf{y}) \tag{2.11}$$

This formulation can be thought of as *soft constraints*. Here, constrained conditional models (Chang et al., 2008) instead of enforcing that the output structure must satisfy each of the constraints. Therefore, a structure that does not satisfy all of the constraints can still be the optimal output structure, if the model score for that structure is high. Each constraint C_j has an associated penalty ρ_j that applies to every structure that violates the constraint.

Since the constraints relate different parts of the structure, they are highly non-local. Solving the inference problem according to Equation 2.9 and Equation 2.11 can be computationally challenging since introducing non-local constraints makes application of dynamical programming algorithms difficult. The most popular paradigm for solving these inference problems are via integer linear programming. Recently proposed dual decomposition (Koo et al., 2010) can also be used. We show below an example formulation of a constrained condition model in terms of an integer linear program in the context of a semantic role labeling task.

Let a sentence be: *Harry ordered pizza*. For the verb *order*, the phrases *Harry* and *pizza* are potential semantic roles. Each phrase can be assigned only one out of 52 semantic roles and each core semantic role (A0, A1, A2, A3, A4, A5, AA) can be assigned to only one phrase. Let the binary variable x_{ij} indicate if phrase i is assigned the semantic role j. Here $i \in \{1, 2\}$ and $j \in \{1, 2, \ldots, 52\}$. Also assume that $j \in \{1, 2, 3, 4, 5, 6\}$ correspond to the six core semantic roles. Let us assume that we have trained a multi-class classifier that can take each phrase as input and produce a probability distribution over possible semantic roles as output. The score s_{ij} for each decision variable x_{ij} is the corresponding probability output by the local classifier $P(y_j|x_i)$.

The prediction problem can be solved by the following integer program

maximize $\sum_{i=1}^{2} \sum_{j=1}^{52} s_{ij} * x_{ij}$

subject to $\sum_{j=1}^{52} x_{ij} = 1, i = 1, 2$

$x_{1j} + x_{2j} <= 1, j = 1, 2, 3, 4, 5, 6$

The first set of constraints ensures that only one semantic role is selected per phrase. The second set of constraints ensures that a core semantic role can be selected either for phrase 1 or for phrase 2.

2.7 Summary

This chapter provided the necessary machine learning backgrounds for this thesis.

Chapter 3

Related Work

3.1 Introduction

This chapter gives an overview of prior research in domain adaptation. Domain adaptation is a diverse problem with many different scenarios and goals. It often happens that we have a large amount of labeled data in one domain (source domain) but our domain of interest (target domain) has only a small amount of labeled data. A focus of active research has been how to effectively combine the labeled data from both source and target domains with an aim to reduce misclassification error on the target domain only. This scenario of adaptation w/ minimal labeled data in the target is often referred to as *Supervised domain adaptation*. In Section 3.2, we will review the proposed approaches in the literature for this type of adaptation. While none of the methods proposed in this thesis falls in this category, it is still important to discuss it for completeness. The second scenario for adaptation assumes that we have plenty of labeled data from one domain but no labeled data exists in the target domain. Both source and target domains contain plenty of unlabeled data. This category of adaptation without any labeled data in the target domain is often referred to as *Unsupervised Domain Adaptation*. Several methods proposed in this thesis will fall in this category. So we intend to give a fairly detailed overview of prior research in this area in Section 3.3.

3.2 Adaptation w/ Labeled Data in Target

In this setting, we will categorize the proposed approaches and discuss the essential principles of each group of approaches.

3.2.1 Source-as-Prior Based Methods

There exists several approaches in the literature (Chelba and Acero, 2004; Roark and Bacchiani, 2003; Bacchiani and Roark, 2003) that fall in this category. The intuition is that from the large source labeled data, we can estimate the optimal model parameters. Then, we can use these parameters as priors for estimating the optimal parameters from the target labeled data. Since we are using a prior distribution based on the source parameters, estimation of the target parameters will be accurate even though the target labeled data may be small.

Let the labeled data in the source domain be $D_s = \{x_s^n, y_s^n :\in \mathcal{X} \times \mathcal{Y}, 1 \leq n \leq M\}$ and the labeled data in the target domain be $D_t = \{x_t^n, y_t^n :\in \mathcal{X} \times \mathcal{Y}, 1 \leq n \leq N\}$. Let $Y_s = \{y_s^n\}, X_s = \{x_s^n\}, Y_t = \{y_t^n\}, X_t = \{x_t^n\}$

The source model Λ_s^* is estimated from the source labeled data D_s and a prior distribution with mean γ.

$$\Lambda_s^* = \arg\max_{\Lambda} P(D_s; \Lambda) P(\Lambda|\gamma) \tag{3.1}$$

Next, the output of a function f with the source parameters as the input of the function, becomes the mean of the prior distribution for estimating optimal parameters for target domain Λ_t^*.

$$\Lambda_t^* = \arg\max_{\Lambda} P(D_t; \Lambda) P(\Lambda|f(\Lambda_s^*)) \tag{3.2}$$

Different choices of prior distribution $P(\Lambda|\gamma)$ are possible, Chelba and Acero (2004) defines it as a Gaussian distribution with mean γ and diagonal covariance matrix. They also define f to be identity mapping and $P(D; \Lambda)$ as the conditional probability of the labels given the instances, and perform conditional log likelihood maximization. The task was predicting the capitalization of each word in a sentence.

Compared to Chelba and Acero (2004), the task in Bacchiani and Roark (2003) is language modeling and so the parameters to estimate are the probability distribution of a word given its n-gram context, instead of weight vectors as in Chelba and Acero (2004). They train Λ_s^* using Katz backoff directly from source domain unlabeled text instead of Equation 3.1. They assume that f is a function that performs element-wise multiplication for the input vector. They assume the prior distribution $P(\Lambda|f(\Lambda_s^*))$ is given by the Dirichlet distribution parameterized by $f(\Lambda_s^*)$.

Under different assumptions of f, they show that the MAP estimation in Equation 3.2 reduces to either merging of counts of n-grams from both domains or linear interpolation of the probabilities estimated from the two domains. In Roark and Bacchiani (2003), the exact same formulation is used, except the task is PCFG parsing and the parameters of interest are the rule probabilities of PCFG.

3.2.2 General Distribution Based Methods

The Source-as-Prior based methods discussed above learn two weight vectors: one for the source and one for the target. The methods discussed in this section learn three weight vectors. Along with the source and target weight vectors, they learn a weight vector from instances assumed to be from the *general* distribution, which is common to both source and target domains. The source, target and general weight vectors are all learned jointly and affect each other, resulting in an accurate target weight vector.

In Daumé III and Marcu (2006); Daumé III (2007), it is argued that some instances are domain independent while others are domain dependent. An example in the POS tagging task is that the instances corresponding to the word *the* are domain independent and all of them are labeled as *Determiner*. But instances corresponding to the word *monitor* will be labeled as *Verb* in news domain and *Noun* in computer monitor related technical documentation. In Daumé III and Marcu (2006), the "generative story" is: For each instance $x_s^n(x_t^n)$, first sample a decision variable $z_s(z_t)$ (binary) from a Bernoulli distribution that indicates whether the instance was drawn from the domain specific distribution $s(t)$ or the general distribution g. Depending on z, sample the instance from the corresponding distribution $\psi_s(\psi_t)$ or ψ_g. There are also three weight vectors Λ_s, Λ_g and Λ_t for the source, target and general distribution respectively. Depending on the value of z sampled for that instance, select the corresponding model and generate the label for that instance. The model is a maximum entropy markov model and model parameters are found by maximizing the conditional expectation (Jebara and Pentland, 1998).

A computationally inexpensive method for adaptation based on the same principle as above is given in Daumé III (2007). This is essentially a feature augmentation approach where each instance keeps two copies of the same feature, one for capturing domain specific weight and the

another for capturing domain independent weight. A source instance x_s^n is represented as $\langle x_s, x_s, 0\rangle$ and a target instance x_t^n is represented as $< 0, x_t, x_t >$. Then a single weight vector is learned using the SEARN algorithm (Daumé III et al., 2009). The weight vector corresponding to the first two parts of the feature vectors is the source weight vector $\Lambda^* s$ and the weight vector corresponding to the last two parts of the feature vectors is the target weight vector Λ_t^*. Whereas prior-based methods find Λ_s^* and Λ_t^* in a pipeline, this approach can be thought of finding Λ_s^* and Λ_t^* jointly. The benefit of this method is that it is very simple and outperforms more complex prior-based methods.

The basic idea of Finkel and Manning (2009) is that the general weight vector will act as a prior over the domain specific weight vectors. As a result, the learning will penalize the difference in weights between the weight vector of each domain and the general weight vector. The intuition is that if a discriminative feature is present in one domain but not in another, that feature will gain high weight in the weight vector of that domain. This will force the weight in the general weight vector to be high, which in turn will force the weight of that feature in the other domain where that feature was absent, to be high as well. Here the objective is:

$$\Lambda_s^*, \Lambda_t^*, \Lambda_g^* = \underset{\Lambda_s, \Lambda_t, \Lambda_g}{\arg\max} P(D_s; \Lambda_s) P(D_t; \Lambda_t) P(\Lambda_s | \Lambda_g, \sigma_s^2) P(\Lambda_t | \Lambda_g, \sigma_t^2) P(\Lambda_g | \gamma) \tag{3.3}$$

In Finkel and Manning (2009), both $P(D_s; \Lambda_s)$ and $P(D_t; \Lambda_t)$ are modeled as conditional probabilities of the labels given the instances, parameterized by Λ_s and Λ_t respectively. $P(\Lambda_s | \Lambda_g, \sigma_s^2)$ and $P(\Lambda_t | \Lambda_g, \sigma_t^2)$ are modelled as Gaussian distributions with mean Λ_g and variances σ_s^2 and σ_t^2 respectively. $P(\Lambda_g | \gamma)$ is modelled as another Gaussian distribution. Given D_s and D_t, $\Lambda_s^*, \Lambda_t^*, \Lambda_g^*$ are found by a joint maximization procedure. After $\Lambda_s^*, \Lambda_t^*, \Lambda_g^*$ are found, Λ_g^* is simply discarded and Λ_s^*, Λ_t^* are used as predictors for source and target domain respectively. It can be shown that if $\sigma_s^2 = \sigma_t^2$, this method is equivalent to that of Daumé III (2007).

3.2.3 Instance Weighting Based Methods

This line of work (Jiang and Zhai, 2007a; Dai et al., 2007; Bickel et al., 2007; Huang et al., 2006) uses careful weighting of source and target domain instances and trains a single model over the combined source and target weighted instances. The notions of general distribution and domain

specific distribution are dealt with implicitly. The goal is to give source domain specific instances close to zero weight such that they do not confuse the learner into learning an inaccurate weight vector for target domain.

The intuition of Jiang and Zhai (2007a) is that some labeled instances in the source domain will be useful for classification in the target domain, others will be harmful. So we can train a model on a subset of the source labeled data and the entire target labeled data. Then we can trust the labels for target domain unlabeled instances for which the model had the highest confidences for prediction and treat them as additional labeled data. Then we can retrain a model in this way until we converge. Mathematically, a joint optimization problem from Jiang and Zhai (2007a) is shown below:

$$\begin{aligned}\Lambda^* = \arg\max_{\Lambda} \lambda^s \sum_{n=1}^{M} \alpha_n \beta_n \log p\left(y_n^s | x_n^s; \Lambda\right) \\ + \lambda^t \sum_{n=1}^{N} \log p\left(y_n^t | x_n^t; \Lambda\right) \\ + \lambda^u \sum_{n=1}^{U} \sum_{y \in \mathcal{Y}} \gamma_n(y) \log p\left(y | x_n^u; \Lambda\right) \end{aligned} \tag{3.4}$$

In Equation 3.4, the first term in the right hand side corresponds to source labeled data, the second term corresponds to target labeled data and the third corresponds to target unlabeled data. λ^s, λ^t and λ^u are domain-level weights for the source labeled data, target labeled data and target unlabeled data,respectively. The purpose of α_n is to remove the instances that are specific to the source domain. Consider, again, the word *monitor* that may be a *verb* in the source domain (news) but a *noun* in the target domain (technical documentation). The weights of all instances in the source domain corresponding to the word *monitor* with label *verb* should be set to 0. To do this, Jiang and Zhai (2007a) train a model from target labeled data alone and sets α_n to 0 for all source domain labeled instances that were misclassified with the highest confidence. α_n was set to 1 for all other instances. While α_n accounts for the difference in $p(y|x)$ across source and target domain, the goal of β_n is to account for the difference of $p(x)$ across source and target domain. Adapting to the change of $p(x)$ is important since during training, the classification error that model tries to

minimize is given by $E_{x \backsim p(x)} \Lambda(x) \neq y$. Therefore, a model trained using $p(x)$ that is different than the target domain $p(x)$ will be suboptimal for classifying target domain instances. β_n should ideally be set as $\frac{p^t(x)}{p^s(x)}$ but this requires density estimation. In Jiang and Zhai (2007a), all β_n variables were set to 1. The remaining parameters were set empirically. After Λ^* was obtained by optimizing according the sum of the first two terms of the r.h.s. of Equation 3.4, $\gamma_n(y)$ were set to 1 for the top k instances from target domain unlabeled data where y was the prediction according to Λ^* and a new Λ^* was obtained again until convergence.

While Jiang and Zhai (2007a) used static weights for all instances from both source and target, the Adaboost like algorithm in Dai et al. (2007) adjusts the weights dynamically through each training iteration. The differences from Adaboost are

- The accuracy of each weak learner is measured over the target domain labeled instances only.
- Similar to Adaboost, when a target domain instance is misclassified, its weight increases. Unlike Adaboost, when a source domain instance is misclassified, its weight goes down by a constant amount, because the source domain instance might have been *dissimilar* to the target domain instances.

For boosting, a weak learner Λ_i is learned at each iteration i of training from the weighted source and target domain instances. Let the weight of nth target (source) domain instance at iteration i be $w^i_{n,t}(w^i_{n,s})$. Let the nth target (source) instance be denoted by (x^t_n, y^t_n) $((x^s_n, y^s_n))$. The misclassification error of the learner at ith iteration is measured over only target instances and is given by

$$\epsilon_i = \frac{\sum_{n=1}^{N} w^i_{n,t} * |\Lambda_i(x^t_n) - y^t_n|}{\sum_{n=1}^{N} w^i_n} \tag{3.5}$$

The weight of the weak learner β_i is calculated as

$$\beta_i = \frac{\epsilon_i}{1 - \epsilon_i} \tag{3.6}$$

On next iteration, weights of the instances will change depending on predictions made by the weak learner learned in the last iteration. The intuition behind the weight update is that if a

target domain instance was misclassified in the last iteration, its weight should go up but if a source domain instance was misclassified, its weight should go down because that source domain instance might have been dissimilar to the target domain instances. The weight of a source domain instance at iteration i is calculated as

$$w_{n,s}^{i} = w_{n,s}^{i-1} * \gamma^{|\Lambda^{i-1}(x_n^s) - y_n^s|} \tag{3.7}$$

The weight of each target domain instance at iteration i is calculated as

$$w_{n,t}^{i} = w_{n,t}^{i-1} * \beta_{i-1}^{-|\Lambda^{i-1}(x_n^t) - y_n^t|} \tag{3.8}$$

In Equation 3.7 and 3.8, $0 \leq \gamma \leq 1$ is a constant for source domain instances. $0 \leq \beta_i \leq 1$ for target domain instances for each training iteration i. Also, $0 \leq |\Lambda^{i-1}(x_n^s) - y_n^s|, |\Lambda^{i-1}(x_n^t) - y_n^t| \leq 1$. So for misclassification on source domain instances, the weight goes down, but for misclassification on target domain instances, the weight goes up.

The final hypothesis is a combination of the weak learners learned in different iterations. It is given by:

$$\Lambda^*(x) = \begin{cases} 1 & \text{if } \prod_i \beta_i^{-\Lambda_i(x)} \geq \prod_i \beta_i^{-\frac{1}{2}} \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$

3.2.4 Support Vector Based Methods

The work of Wu and Dietterich (2004) is a method that makes use of SVM based formulation and can incorporate labeled data from a distribution different than the distribution of interest. The auxiliary sources in that paper can be thought of as the source domain in the terminology of the current thesis. They specifically focus on 1-norm SVM or LP SVM. The LP SVM for the target

domain instances can be written as

$$\Lambda_t^* = \arg\min_{\{\alpha^t\}} \sum_{n=1}^{N} \alpha_n^t + C \sum_{n=1}^{N} \xi_n^t \tag{3.10}$$
$$\text{s.t.: } y_n^t \left(\sum_{m=1}^{N} y_m^t \alpha_m^t K\left(x_m^t, x_n\right) + b \right) + \xi_n^t \geq 1 \; \forall x_n \in D_t$$
$$\alpha_n^t \geq 0, \forall n$$

Note that for each target instance, a constraint is generated and two variables (α_n^t and ξ_n^t) are added to the objective function. The research question asked by Wu and Dietterich (2004) was that if we add more labeled instances from a different distribution, should we add constraints for them or the variables in the objective function or both? Incorporating the source domain labeled data, LP SVM can be written as

$$\Lambda_t^* = \arg\min_{\{\alpha^t\},\{\alpha^s\}} C_1 \sum_{n=1}^{N} \alpha_n^t + C_2 \sum_{n=1}^{N} \xi_n^t + \underline{\underline{C_3 \sum_{n=1}^{M} \alpha_n^s}} + \underline{C_4 \sum_{n=1}^{M} \xi_n^s} \tag{3.11}$$

$$\text{s.t.:} y_n^t \left(\sum_{n=1}^{N} y_n^t \alpha_n^t K\left(x_n^t, x\right) + \underline{\underline{\sum_{n=1}^{M} y_n^s \alpha_n^s K\left(x_n^s, x\right)}} + b \right) + \xi^t \geq 1 \; \forall x \in D_t \tag{3.12}$$

$$\text{s.t.:} y_n^s \left(\sum_{n=1}^{N} y_n^t \alpha_n^t K\left(x_n^t, x\right) + \underline{\underline{\sum_{n=1}^{M} y_n^s \alpha_n^s K\left(x_n^s, x\right)}} + b \right) + \xi^s \geq 1 \; \forall x \in D_s \tag{3.13}$$
$$\alpha_n^t \geq 0, \forall n$$
$$\alpha_n^s \geq 0, \forall n$$

Wu and Dietterich (2004) investigated three strategies:

- Do not require the classifier to perform well on source labeled data. This can be achieved by ignoring the term corresponding to source domain misclassification error (the single underlined term in Equation 3.11) (equivalently setting $C_4 = 0$).
- Do not allow any instance from source labeled data to be a support vector. This can be

accomplished by ignoring all double underlined terms from Equation 3.11. (equivalently setting $\{\alpha_n^s = 0\}$)

- Allow instances from source labeled data to be both support vectors and also require the classifier to perform well on source labeled data

The conclusion of Wu and Dietterich (2004) was that the third strategy performed the best when the amount of target labeled data was small. So source labeled data is useful both as constraints and as training data.

3.2.5 Relation to Current Thesis

All methods proposed in this thesis assume that there is no labeled data in the target domain. Therefore, the methods discussed in this section do not readily apply to our scenario. However, in Chapter 8, we show that it is possible to convert a unsupervised adaptation problem into a supervised adaptation problem by automatically generating some pseudo labeled data in the target domain using some heuristics. We show that then all methods discussed in this section become applicable. Specifically, we evaluate some of the general distribution based methods from Section 3.2.2 and show that after generation of pseudo labeled data in target, these methods give better results than other methods proposed specifically for the paradigm of unsupervised adaptation.

3.3 Adaptation without Labeled Data in Target

The previous section presented an overview of prior research in adaptation to a new domain with some amount of labeled data for the new domain. However, labeling is very expensive and time consuming. Another challenge for NLP is that the number of domains is extremely large. Annotating even a small amount of labeled data for every new domain is almost impossible. Therefore there have been a surge of interest in adaptation to a new domain without any need for labeled data in the new domain. The intuition is that often we have a significant amount of unlabeled data in the new domain as well as the source domain. By leveraging the statistics over such unlabeled data, we can adapt the model trained on the source labeled data to the target domain, without any

use of labeled data from the target domain. For this section, we will first present a brief overview of theoretical findings and then present the overview of different empirical frameworks proposed for this scenario.

3.3.1 Theoretical Studies

There have been several theoretical studies for domain adaptation (Ben-David et al., 2006; Blitzer et al., 2008; Huang et al., 2006; Blitzer et al., 2011; Foster et al., 2011; Kumar et al., 2010). We will discuss Ben-David et al. (2006) because of its importance and relevance to the current section. The point of the paper is that we should have a good feature-based representation for successful domain adaptation. This representation should be such that it will help us learn a hypothesis with a small empirical error on source labeled data and at the same time, this representation should lower the d-distance between the two domains. Let instance space be $\mathcal{X}$, an individual instance x, feature representation ϕ, source distribution D^s, target distribution D^t, distribution over the feature vectors of source and target domain instances be D^s_ϕ and D^t_ϕ. *d-distance* is defined as

$$d_H(D^s_\phi, D^t_\phi) = 2 \sup_{A \in A_H} |Pr_{D^s_\phi}(A) - Pr_{D^t_\phi}(A)| \tag{3.14}$$

Here H is the hypothesis space. A_H is defined as

$$A_H = \{X \subset \mathcal{X} : h(x) = 1, x \in X, h \in H\} \tag{3.15}$$

So, d-distance is the difference in probabilities of source and target domain distribution for drawing a set of instances that is the set of positives for some hypothesis in the hypothesis space. Even if the original representation of the instances was such that D^s and D^t are very different, we can use a good feature function ϕ to reduce the difference between $Pr_{D^s_\phi}$ and $Pr_{D^t_\phi}$. This will then reduce the d-distance also. However, this feature representation should not sacrifice the discriminative properties, so that an accurate classifier for the source labeled data can still be found. Mathematically, given a feature function ϕ, the true error on target can be bounded by the

sum of the empirical error on source labeled data and the d-distance that depends on ϕ.

$$\epsilon_T(h) \leq \hat{\epsilon}_S(h) + d_H(D^s_\phi, D^t_\phi) + \text{const } \forall h \in H \tag{3.16}$$

Here $\epsilon_T(h)$ is the true target error for h, and $\hat{\epsilon}_S(h)$ is the empirical source error of h.

So if we can find a good feature function ϕ such that source empirical error with ϕ remains small (the first term on R.H.S. in Equation 3.16) and d-distance is also small (the second term on R.H.S. in Equation 3.16, we can expect true target error to be small (ϵ_t in Equation 3.16).

3.3.2 Bootstrapping Based Methods

This scenario of adaptation with unlabeled data is very related to semi-supervised learning (Chapelle et al., 2006). The difference is that in semi-supervised learning, unlabeled data is from the same distribution / domain as the labeled data, whereas in domain adaptation, unlabeled data from the target domain is different distribution than the labeled data from the source domain. There have been many semi-supervised learning methods proposed in the literature, e.g. Co-training (Blum and Mitchell, 1998) and Structural Learning (Ando and Zhang, 2005). Due to the relevance, we will discuss Ando and Zhang (2005).

Structural learning can be explained as clustering of features. Consider the fragments *Google Group*, *Google Inc*, *Google Ltd*, *Microsft Group*, *Microsoft Inc*, *Microsoft Ltd*. The "current word" features {Group, Inc, Ltd} should be grouped into a cluster because they have similar co-occurrence patterns. There has been many different ways of building word clusters, e.g. Brown et al. (1992). The method in Ando and Zhang (2005) builds the cluster in several steps: create auxiliary classification problems of different forms, e.g. predict previous words, predict current words etc. Two examples of the first type are: predict if the previous word is Google, predict if the previous word is Microsoft. The example fragments will contribute 3 positive and 3 negative training examples for each of these two problems. Now, the features *current word = Inc*, *current word = Ltd*, *current word = Group* should be grouped together only if this group is useful for classification of multiple auxiliary prediction problems (e.g. Microsoft, Google, Facebook etc.). So in Ando and Zhang

(2005), classifiers of auxiliary prediction problems and this grouping are learned jointly as

$$\{w_l^*\}, \theta^* = \arg\min_{\{w_l\},\theta} \sum_{l=1}^{m} \sum_{i=1}^{n_l} w_l^T x_i + v_l^T \theta x_i \tag{3.17}$$

Here θ is a projection matrix where each row contains high values in columns corresponding to features in same group, w_l is a weight vector for auxiliary predictor for problem l operating on the full dimensionality of the example and v_l is the weight vector for the cluster based features for auxiliary predictor for problem l. Once θ^* is learned, each training example x is augmented with cluster based features $\theta^* \cdot x$ and a weight vector is learned from training data. In terms of our example, let some row r in θ contains high values for the columns corresponding to the features *current word = Inc*, *current word = Ltd*, *current word = Group* and close to zero values for all other features. If instance x has one of these features active, $\theta_r \cdot x$ will be non-zero and act similar to a cluster feature.

Standard Bootstrapping (Yarowsky, 1995) starts with a classifier trained from the source domain labeled data, applies it on target domain unlabeled data and selects the top k instances with highest prediction score from the classifier. Assuming the predicted labels for these k instances as gold labels, these instances are added to the labeled data and the classifier is retrained. Wu et al. (2009) proposes a different approach for selecting the top k instances. There, the scoring for each target instance depends on two factors. First, if the size of the label set is large, the classifier will be more accurate in predicting certain labels. For example, for named entity recognition, PERSON labels are relatively easy to detect but FACILITY, LOCATION and GPE are harder to classify correctly. So the accuracy of the source classifier for every label y, acc(y), is measured on some held out set. For selection of top k confident predictions from target unlabeled data for addition to training data, acc(y) will be taken into account so that we select target instances with those class labels on which we expect the classifier to be accurate. Second, for every target instance, the entropy $h(p^s(y))$ of the source classifier predicted distribution over labels is measured. A target classifier is trained from data labeled automatically by the source classifier and again for every target instance, the entropy $p^t(y)$ of target classifier predicted distribution over labels is measured. The difference $h(p^s(y)) - h(p^t(y))$ is large for a target instance when the source classifier was uncertain but the target classifier was certain. Wu et al. (2009) argues that this target instance contains information

unseen by the source classifier (e.g. features unseen by the source classifier). Final score for each instance is given by

$$\text{score} = \text{acc}(y^*)(h(p^s(y) - h(p^t(y)))), y^* = \arg\max_{y \in \mathcal{Y}} p^s(y) \tag{3.18}$$

The target domain instances with the highest scores calculated according to Equation 3.18 are selected for addition to the training data.

Jiang and Zhai (2007b) proposed a two stage method for domain adaptation. Their setting is where we have labeled data from multiple domains and we want to build a classifier for a new domain with no labeled data. The goal of the first stage is to select features that generalize across domains from labeled data of multiple domains. A projection matrix θ is used to capture the generalizable features. Let $x \in R^d$ and $\theta \in \{0,1\}^{h \times d}$. In addition, each row of θ can contain only one non-zero value. Therefore, $\theta \cdot x$ effectively selects h features from x. Augmenting each x with $\theta \cdot x$, we can learn the set of weight vectors $\{\langle w_i, v \rangle\}$ from the multiple source domains along with θ in a joint fashion such that we maximize the conditional log likelihood of the labeled data from all source domains. Here w_i is the weight vector for x corresponding to the i'th source domain and v is the weight vector for $\theta \cdot x$ learned from all source domains.

$$\{w_i^*\}, v, \theta^* = \arg\min_{\{w_i\}, \theta} \sum_{i=1}^{|s|} \sum_{n=1}^{N^{s_i}} \log p(y_{in} | x_{in}; w_i, v) + \lambda * ||w_i||^2 \tag{3.19}$$

Once θ is learned, the goal of the next stage is to take the unlabeled data from the new domain into account. This is done in a standard self-training fashion, selecting the top k target domain instances with highest prediction score using the general weight vector v, assuming the predicted labels for these instances as gold labels and solving Equation 3.19 again with one more likelihood term added from the newly labeled top k target domain instances.

3.3.3 Representation Learning Based Methods

Blitzer et al. (2006) is one of the first methods for unsupervised domain adaptation. This method is very similar to Ando and Zhang (2005). The differences are:

- Ando and Zhang (2005) addresses semi-supervised learning and therefore, learns predictors for auxiliary problems from unlabeled data from source domain only. Blitzer et al. (2006) addresses domain adaptation and learns predictors for auxiliary problems from combined unlabeled data of both source and target domains.

- Ando and Zhang (2005) learns the predictors for auxiliary problems and projection matrix in a joint fashion whereas Blitzer et al. (2006) follows a pipeline approach. It first trains the predictors for auxiliary problems and then creates the projection matrix.

- Ando and Zhang (2005) used the prediction problems of the form *predict if the next word is* w where w was the most frequent words in one domain. Blitzer et al. (2006) use the 50 most frequent words in both domains as w. The same technique is also used in Blitzer et al. (2007), except instead of choosing the most frequent words as pivots, the words that have the highest mutual information with the class label were selected as pivots. This pivot selection was suitable for the sentiment classification task.

The representation learned by Blitzer et al. (2006) is a linear projection of the feature vector. Comparatively, Huang and Yates (2009) creates a representation by training a Hidden Markov Model over the combined unlabeled data of both source and target domains. In this process, an HMM is trained using the Expectation Maximization algorithm (Dempster et al., 1977) and then the HMM state of each token is added as a feature. Since HMM is learned over the combined unlabeled data of both source and target, it is expected that words that occur in similar contexts will be assigned similar HMM states. Therefore, this representation can bring the target domain closer to the source domain. Huang and Yates (2009) compared with several other possible representations based on latent semantic indexing (Deerwester et al., 1988) and TF-IDF based representation and concluded that HMM-based representation performs the best.

A popular way of learning representations these days is via neural network. In Glorot et al. (2011), task is adaptation of review sentiment classification into positive or negative across multiple domains. The method is to collect unlabeled reviews from many domains and feed each review document to a neural network. The activations of the neurons at different layers when a review doc is given as input, are collected and added as additional features for that review doc, along with the

traditional bigram and unigram features. Then a classifier is trained over the new representation of the source domain instances. Since the representations are learned jointly over many domains and activations of neurons in the hidden layers can be thought of as capturing different aspects of the inputs, the source domain classifier trained over this representation can generalize over domain variations. Training of the network is performed by attempting to minimize reconstruction error. Input to the network is a corrupted version of a review doc and gold output is the original review doc. So the network learns weights such that it can reconstruct the original review from the corrupted version. The layer weights are trained greedily, at first; a one hidden layer network tries to reconstruct the original input and then incrementally, one hidden layer is added that tries to reconstruct the output of the previously added hidden layer's output. This greedy layer-wise training was suggested in Bengio et al. (2007).

Learning representations via neural network can be very time consuming. Moreover, the objective to minimize is non-convex and so stochastic gradient descent can only find a local minima. Also, when neural networks are trained to reconstruct original input from a corrupted version of the input, it is not possible to generate all possible corrupted versions of the input because this set is too large. The approach in Chen et al. (2012) proposes a remedy for all these problems. Again, the goal is to minimize reconstruction error from a corrupted version of input

$$W^* = \arg\min_W \sum_{n=1}^{N} \|x_i^s - W\tilde{x_i^s}\|^2 + \sum_{n=1}^{M} \|x_i^t - W\tilde{x_i^t}\|^2 \tag{3.20}$$

Here $\tilde{x} \in R^d$ is the corrupted version of $x \in R^d$ and $W \in R^{(d \times d)}$ is the reconstruction matrix. Let us assume that $\tilde{x}$ was generated by turning off each feature in x with probability p. Given a randomly generated set of $\{\tilde{x}\}$ in this fashion, we can estimate W^* in closed form. But W^* will be stochastic since $\{\tilde{x}\}$ was stochastic. However, we can estimate an approximation to the expectation of W^* w.r.t. the probability distribution over $\{\tilde{x}\}$. This is given by

$$W = PQ^{-1} \tag{3.21}$$

where,

$$P_{ij} = \begin{cases} \frac{(XX^T)_{ij}}{(1-p)^2} & \text{if } i \neq j \\ \frac{(XX^T)_{ii}}{1-p} & \text{if } i == j \end{cases}$$

and

$$Q_{ij} = \frac{(XX^T)_{ij}}{1-p}$$

Similar to Glorot et al. (2011), stacking can be applied. At the beginning, the network consists of only input nodes and output nodes and W is the weight matrix connecting input and output neurons. After learning W, the output of the neurons at the output layer are passed through a *tanh* filter. Then a new layer of neurons is added for reconstruction of the output of the output layer and neuron layers are added this way, incrementally. Finally, the outputs of all neurons in all layers are added as additional features on top of the neural network inputs. Then, similar to Glorot et al. (2011), a classifier is trained from the extended representation of instances from source domain labeled data.

It is believed that the success of the previous methods relies on the fact that outputs of the neurons at different hidden layers model the structure of the problem very well, at different levels of granularity. But it is not easy to inspect or verify. The work of Raina et al. (2007) explicitly models these more meaningful compositional features. An example task is image classification using a small amount of labeled data. Here, we can leverage a large number of unlabeled images to learn meaningful image features (e.g. image segments depicting edges). Then we can represent the original training instances in terms of the more meaningful features and we will have an easier classification problem. Formally, given many unlabeled instances $\{x \in R^d\}$, a set of basis vectors $\{b \in R^d\}$ are learned and each instance is represented as a linear combination of these basis vectors. The representation of each instance is transformed to be the vector of coefficients in the linear combination and a classifier is learned over this new representation instead of original representation. Mathematically,

$$\{b^*\} = \underset{\{b\},\{a\}}{\arg\min} \sum_{n=1}^{N} \|x_n - \sum_{j=1}^{J} a_n^j \cdot b_j\| + \lambda \|a_n\| \tag{3.22}$$

Here $\{a_n^j\}$ is the new representation for instance x_n and $\{b_j\}$ are the basis vectors. The optimization is performed jointly over $\{a\}$ and $\{b\}$ variables. A sparsity requirement is also imposed on the new representation of each instance in terms of $\|a_n\|$.

While the previous representation learning based methods attempt to reconstruct the instances depending on the task, a more popular line of work is on learning continuous representations for words. A vector of real numbers for each word is learned and then each index in the vector can be thought of as capturing one salient aspect of that word. An early application of this type of word embedding was in language modeling, where the goal is to model the probability distribution of words, given the previous words in context. This can be modeled as a classification task where a classifier will predict the probability distribution of the n'th word, given the previous (n-1) words of context. The approach in Bengio et al. (2006) was to learn a fixed dimensional continuous vector for each word and learn a classifier on top of this word embedding. Let, $C \in R^{(d \times V)}$ be a matrix where the i'th column is the d-dimensional embedding for i'th word. The language model probability of $P(w_n|w_{n-1}, \ldots, w_1)$ is modeled as

$$P(w_n|w_{n-1}, \ldots, w_1) = f(C(w_1), \ldots, C(w_n)) \tag{3.23}$$

In Bengio et al. (2006), f is modeled as a neural network. f and C are found jointly, by maximizing the n-gram probability of the training corpora, using stochastic gradient ascent.

While Bengio et al. (2006) learned the word embeddings from language modeling only, Collobert and Weston (2008) produced word embeddings by joint training over several tasks (e.g. shallow parsing, NER) along with language modeling. The intuition is that if two words have similar labels in any of these tasks, they should have similar embeddings. Again, if two words have different labels in any of these tasks, their embeddings should be different. For example, if one word appears as part of a person name in NER training data and another word appears as a non-entity in NER training data, training the NER architecture will influence the embeddings of these two words to be different. Each task has its own set of features where each feature is mapped into a fixed dimensional real vector. Embeddings of the words are also input to the neural network architecture. In addition, some layers of neurons can be shared across the multiple tasks but the output layer of each task must be different because predictions need to be different for every task. Jointly training

multiple tasks was shown to give improvement training them separately.

The prospect of adding these embeddings as features for every word to model the effect of the semi-supervised learning was proposed in Turian et al. (2010). Given any sequential prediction problem for natural language, for each word, we can simply add each number in its embedding representation as a feature, along with other task-specific features. We can then train a new model. Since the embeddings were trained over a very large corpora, this can be thought of as semi-supervised learning with simplicity and efficiency. Turian et al. (2010) compared word embeddings produced by Collobert and Weston (2008) and Mnih and Hinton (2007) and word clusters produced by Brown et al. (1992), for two NLP tasks (named entity recognition and shallow parsing). The conclusion was that word clusters gave better results, but in general, word embeddings improved over baseline and combining different word embeddings gave additional improvements.

3.3.4 Relation to Current Thesis

All methods proposed in this thesis are for the paradigm for unsupervised adaptation where no labeled data exists for the target domain. The bootstrapping based methods in Section 3.3.2 train a model for each iteration of the bootstrapping process. Compared to these methods, our goal is to avoid any form of training for the new domain. We derive knowledge from diverse places, e.g., lexical resources, related task that is more robust to domain variations and change the inference phase with the same old source domain model.

Some of the methods in Section 3.3.3 learn a representation from the source and target domain unlabeled data. Using features derived from this learned representation, the retrain the source model from the source labeled data. Compared to these methods, we do not aim to learn any representation from unlabeled data directly. Instead, we focus on deriving knowledge from lexical resources, target domain annotation manuals, target domain unlabeled data and from systems for other tasks that are relatively more robust across domain variations. Another category of methods in Section 3.3.3 learns a representation for each word from large unlabeled corpora beyond just target domain unlabeled data and uses features from this representation. The resultant model becomes more robust across domain variations. Cluster of words as used in Ratinov and Roth (2009) for the task of named entity recognition is an excellent example for this type of method.

The methods proposed in this thesis show additional improvements on top of these methods. For example, our baseline NER system always uses the word cluster based features. We show that by using declarative knowledge from various sources at inference time gives us significant improvements on top of this baseline NER system.

3.4 Summary

This chapter presented a thorough overview of prior research in domain adaptation. We categorized the presented approaches into groups and presented the similarities and differences of each approach against others.

Chapter 4

Adaptation Using Transformation

The previous chapters introduced the basic terminology for understanding the methods presented in this thesis. A thorough overview of related works in the literature was also presented to help the reader put the work in this thesis in context. Starting from this chapter, we will present the methods developed in this thesis. The goal of this thesis is to introduce the paradigm of adaptation without training. This chapter will present the first method developed in this direction. The method works by transforming text from the target domain into text similar to the source domain. We will show that with the same source domain model, we can get more accurate predictions on the transformed text compared to the untransformed target text.

4.1 Introduction

In previous chapters, the problem of domain adaptation was introduced. This problem currently hinders the wide-spread deployment of statistical models in many areas, e.g. natural language processing, computer vision. In many NLP tasks, systems trained on annotated data from one domain perform well when tested on the same domain but adapt poorly to other domains. For example, all systems in the CoNLL 2005 shared task (Carreras and Màrquez, 2005) on Semantic Role Labeling showed a performance degradation of almost **10**% or more when tested on a different domain. For the task of POS tagging, performance drops almost **9**% when systems trained on the WSJ domain are tested on the biomedical domain (Blitzer et al., 2006).

The previous chapter provided a thorough overview of the popular domain adaptation methods. There are several possible adaptation scenarios, depending on availability of training data for the target domain. This thesis proposes methods for the most realistic setting, where there is no labeled data available in the target domain. Most works proposed in the literature for this setting have

focused on learning a common representation across training and test domains (Blitzer et al., 2006; Daumé III, 2007; Huang and Yates, 2009). Using this representation, they retrain the model for every new domain. But these are not *Open Domain Systems* since the model needs to be retrained for every new domain. This is very difficult for pipeline systems like SRL where syntactic parser, shallow parser, POS tagger and then SRL need to be retrained, since the cumulative training time for all the components will be very large. Also these methods need a lot of unlabeled data on the new domain to learn meaningful feature correspondences across training and test domains. For example, Blitzer et al. (2006) used 200000 sentences as unlabeled data from the target domain for adaptation experiments. These approaches cannot work when there is not enough unlabeled data or when there are many different domains, for example, in web.

In this chapter, we show that it is possible to adapt to a target domain without any labeled data in the target domain and with the same model from the source domain. We show that we can define transformations based on external knowledge bases and learn transformations from the source domain labeled data itself. Then, given an input sentence from the target domain, we will apply these transformations to generate a set of transformed sentences and then apply the source domain model on these transformed sentences to produce one output for every sentence. Finally, we can combine the outputs from these sentences to produce the final output for the input sentence. We show that this method of transformation of text can give significant improvement for the task of semantic role labeling, when we evaluate an SRL model trained on newswire text on the new domain of literature text, without any change to the model itself.

Contributions

- We introduce the notion of *label-preserving transformation*. We show how to define label-preserving transformations based on knowledge bases and how to learn transformations based on labeled data.

- We show how to apply the transformations to generate multiple transformed sentences from one single sentence and we show how the predictions on these transformed sentences can be combined using a Constrained Conditional Model to output the final prediction for the input sentence.

4.2 Problem Formulation

Let the source domain distribution over instances be D_s and target domain distribution be D_t. We have a model f trained over a set of labeled instances drawn from D_s. If D_s and D_t are very dissimilar, f will not perform well on instances drawn from D_t. The problem is to get good performance from f on D_t.

For an input instance x, a transformation function generates a set of output instances E. The function is label preserving if all instances in E have the same gold standard label as x.

At evaluation time, for test instance x, we can apply all label-preserving transformations and make union of all output instances from the transformation functions. Out of these instances, we only keep those instances that have higher probability than x according to source distribution D_s. The idea is that in this way, we generate some instances that are more like the source domain than the input instance itself, nevertheless, the generated instances have the same label as the input instance. Therefore, the source model f will perform better on the output instances and we can combine the predictions of f on these output instances subject to some natural consistency constraints to produce the final output.

4.3 Motivating Examples

One of the key reasons for performance degradation of an NLP tool is unseen features such as words in the new domain that were not seen in the training domain. But if an unknown word is replaced by a known word without changing the labeling of the sentence, tools perform better. For example, in the task of syntactic parsing, the unknown word *checkup* causes the Charniak parser to make a wrong co-ordination decision on the sentence

> *He was discharged from the hospital after a two-day checkup and he and his parents had what Mr. Mckinley described as a "celebration lunch" at the cafeteria on the campus.*

If we replace the word *checkup* with its hypernym *examination* which appears in the training data, the parse gets corrected. Figure 4.1 and Figure 4.3 show the original and corrected parse trees respectively.

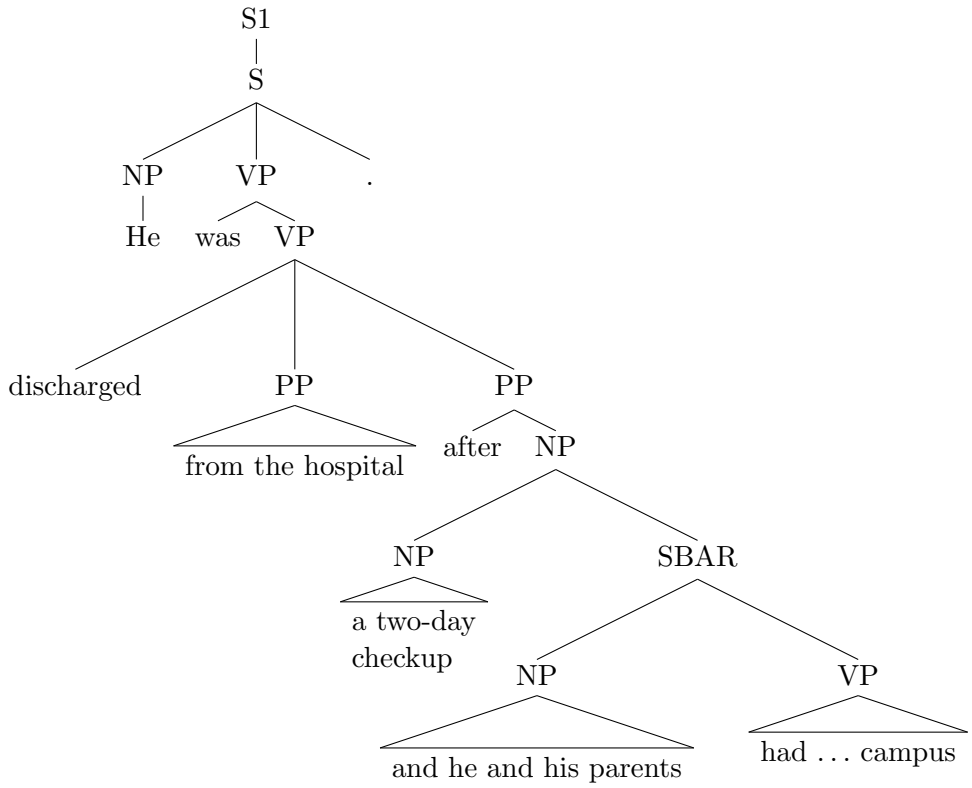


Figure 4.1: Original Parse tree

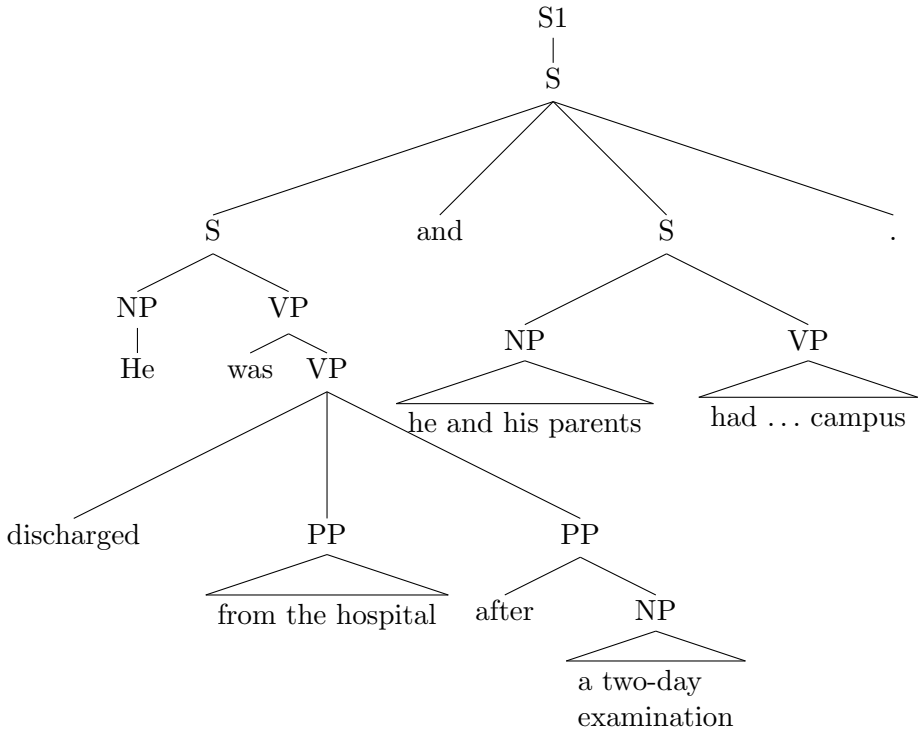


Figure 4.2: Corrected Parse tree after replacement of unknown word *checkup* by *examination*

For the task of semantic role labeling, systems do not perform well on predicates that are infrequent in the training domain. But if an infrequent predicate is replaced with a frequent predicate from the training domain such that both predicates have similar semantic argument structure, the system performs better. Consider the following sentence

Scotty gazed out at ugly gray slums.

The semantic role for the phrase *at ugly gray slums* with respect to predicate *gaze* is *A1*. But the predicate *gaze* appears only once in training data and our model predicts *at ugly gray slums* as *AM-LOC* instead of *A1*. But if *gaze* is replaced with *look*, which occurs 328 times in training data and has similar argument structure (in the same VerbNet class as *gaze*), the system makes the correct prediction.

We want to argue that the problems mentioned above are not exclusively due to improper representation of features. For example, in the example of semantic role labeling, all the features involving the word *gaze* do not get any weight in the model trained from labeled data of news wire because the verb *gaze* does not even appear once in news wire. One way to get over this limitation of sparsity is to use cluster based features, e.g. a feature indicating which cluster the word *gaze* belongs to, a feature indicating which VerbNet cluster the verb *gaze* belongs to, etc.. All these features are part of the state-of-the-art semantic role labeling system of Punyakanok et al. (2008). However, these cluster based features may not have sufficient weights in the source model. So for the verb *gaze*, only the cluster features will be active, however, if we replace the verb *gaze* with *look*, then all lexical features for *look* along with the cluster based features will be active. So the source model will perform better.

4.4 Transformation Functions

After applying a transformation function to get a new sentence from a test sentence, we remember the mapping of segments across the original and transformed sentence. Thus, after annotating the transformed sentence with SRL, we can transfer the roles to the original sentence through this mapping. Transformation functions can be divided into two categories. The first category is *Transformations From List*, which uses external resources like WordNet, VerbNet and Word

Clusters. The second is *Learned Transformations* which uses transformation rules that have been learned from training data.

4.4.1 Transformation From List

Using resources like WordNet, VerbNet and Word Clusters, we derive a sequence of transformation functions. A brief description of these resources is provided below:

WordNet WordNet is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. The most frequently encoded relation among synsets is the super-subordinate relation (also called hyperonymy, hyponymy or ISA relation). It links more general synsets like *furniture, piece_of_furniture* to increasingly specific ones like *bed* and *bunkbed*. Thus, WordNet states that the category furniture includes bed, which in turn includes bunkbed; conversely, concepts like bed and bunkbed make up the category furniture.[1]

VerbNet VerbNet is the largest on-line verb lexicon currently available for English. It is a hierarchical domain-independent, broad-coverage verb lexicon with mappings to other lexical resources such as WordNet. VerbNet is organized into verb classes extending Levin (Levin, 1993) classes through refinement and addition of subclasses to achieve syntactic and semantic coherence among members of a class. Each verb class in VN is completely described by thematic roles, selectional restrictions on the arguments, and frames consisting of a syntactic description and semantic predicates with a temporal function.[2]

SemLink SemLink is a project whose aim is to link together different lexical resources via set of mappings. These mappings make it possible to combine the different information provided by different lexical resources. For example, SemLink provides mapping between VerbNet and PropBank.

Next, we describe a series of transformations for the task of semantic role labeling based on the

[1]The above paragraph is taken directly from the web site `http://wordnet.princeton.edu/`

[2]The above paragraph is taken directly from the web site `http://verbs.colorado.edu/~mpalmer/projects/verbnet.html`

above resources:

I. Replacement of Predicate:

As noted in Huang and Yates (2009), 6.1% of the predicates in the Brown test set do not appear in WSJ training set and 11.8% appear at most twice. Since the semantic roles of a sentence depend on the predicate, these infrequent predicates hurt SRL performance on new domains. Note that since all predicates in PropBank are verbs, we will use the words *predicate* and *verb* interchangeably.

Our heuristic for transformation in this case is defined below
We count the frequency of each predicate and its accuracy in terms of F1 score over the training data. If the frequency or the F1 score of the predicate in the test sentence is below a threshold, we perturb that predicate. We take all the verbs in the same class of VerbNet[1] as the original verb (in cases where the verb is present in multiple classes, we take all the classes). In cases where the verb is not present in VerbNet, we take its synonyms from WordNet. If there is no synonym in WordNet, we take the hypernyms.

From this collection of new verbs, we select verbs that have a high accuracy and a high frequency in training. We replace the original verb with each of these new verbs and generate one new sentence for each new verb; the sentence is retained if the parse score for the new sentence is higher than the parse score for the original sentence.[2] VerbNet has defined a set of verb-independent thematic roles and grouped the verbs according to their usage in frames with identical thematic roles. But PropBank annotation was with respect to each verb. So the same thematic role is labeled as different roles for different verbs in PropBank. Semlink[3] provides a mapping from the thematic role to PropBank role for each verb. After the SRL annotates the new sentence with PropBank roles for the new verb, we map the PropBank roles of the new verb to their corresponding thematic roles and then map the thematic roles to the corresponding PropBank roles for the original verb.

For example, consider the following sentence as the test sentence "He was warned repeatedly." For the verb *warn*, the gold standard thematic role in VerbNet for *He* with respect to *warn* is *Recipient*. The verb *advise* belongs to the same VerbNet class as *warn* and therefore, has similar thematic roles. So, if *warn* is replaced by *advise*, the gold standard thematic role for *He* in the

[1] http://verbs.colorado.edu/ mpalmer/projects/verbnet.html
[2] Parse score is the parse probability returned by Charniak or Stanford parser.
[3] http://verbs.colorado.edu/semlink/

transformed sentence "He was advised repeatedly", with respect to *advise* remains the same as before (*Recipient*). However, the PropBank annotation of roles to arguments did not remain consistent with the thematic role from VerbNet. For example, *He* will have gold standard semantic role of *A2* for verb *warn* in the test sentence but *A1* for verb *advise* in the transformed sentence. The semantic role labeling system is trained on PropBank annotations. After the system annotates the transformed sentence "He was advised repeatedly" with PropBank semantic roles, we map those semantic roles with respect to verb *advise* to thematic roles and then map these thematic roles to the PropBank semantic roles for the verb *warn*. This mapping of PropBank semantic role for each verb to the corresponding thematic role is found in Semlink.

II. Replacement and Removal of Quoted Strings:

Quoted sentences can vary a lot from one domain to another. For example, in WSJ, quoted sentences are like formal statements but in Brown, these are like informal conversations. We generate the transformations in the following ways:

1) We use the content of the quoted string as one sentence. 2) We replace each quoted string in turn with a simple sentence (*This is good*) to generate a new sentence. 3) If a sentence has a quoted string in the beginning, we move that quoted string after the first NP and VP that immediately follow the quoted string. For example, from the test sentence, *"We just sit quiet", he said.* we generate the sentences 1) *We just sit quiet* 2) *"This is good", he said.* 3) *He said, "We just sit quiet".*

III. Replacement of Unseen Words:

A major difficulty for domain adaptation is that some words in the new domain do not appear in the training domain. In the Brown test set, 5% of total tokens was never seen in the WSJ training set.

Given an unseen word which is not a verb, we replace it with WordNet synonyms and hypernyms that were seen in the training data. We used the clusters obtained in Liang (2005) from running the Brown algorithm (Brown et al., 1992) on the Reuters 1996 dataset. But since this cluster was generated automatically, it is noisy. So we chose replacements from the Brown clusters selectively. We only replace those words for which the POS tagger and the syntactic parser predicted different

tags. For each such word, we find its cluster and select the set of words from the cluster. We delete from this set all words that do not take at least one part-of-speech tag that the original word can take (from WordNet). For each candidate synonym, hypernym or cluster member, we get a new sentence. Finally we only keep those sentences that have higher parse scores than the original sentence.

IV. Sentence Split based on Stop Symbols:

We split each sentence based on stop symbols like *;* and *. .* Then we run SRL model on each piece separately and make union of the semantic roles predicted by SRL model on the splitted sentences. The union of the splitted sentences becomes one transformation of the original sentence.

V. Sentence Simplification:

We have a set of heuristics for simplifying the constituents of the parse tree; for example, replacing an NP with its first and last word, removal of PRN phrases etc. We apply these heuristics and generate simpler sentences until no more simplification is possible. An example for each type of transformations is given in Table 4.1.

Node	Input Example	Simplified Example	Operation
NP	**He and she** ran.	**He** ran.	replace
NP	**The big man** ran.	**The man** ran.	replace
ADVP	He ran **fast**.	He ran.	delete
PP	He ran **in the field**.	He ran.	delete
PRN	He − **though sick** − ran.	He ran.	delete
VP	He **walked** and ran.	He ran.	delete
TO	I want him **to** run.	I want that he can ran.	rewrite

Table 4.1: Examples of Simplifications (Predicate is run)

Note that we can use composition of multiple transformation functions as one function. A composition $p_1 \odot p_2(s) = \cup_{a \in p_1(s)} p_2(a)$. We apply II⊙I, III⊙I, IV⊙I and V⊙I. These composite transformations were selected to keep the number of transformed examples small.

4.4.2 Learned Transformations

The source model is inaccurate over sentences that contain verbs and roles that are infrequent in the training data. The purpose of the learned transformation is to transfer such a phrase in the test sentence in place of a phrase of a simpler sentence; this is done such that there exists a mapping from the role of the phrase in the new sentence to the role of the phrase in the original sentence.

Consider the sentence "Mr. Mckinley was entitled to a discount". We will refer to this sentence as the *sentence A*. The gold standard semantic role for the phrase *Mr. Mckinley* with respect to the verb *entitle* is *A2*. There are several sentences with exactly similar construction (noun phrase with semantic role *A2* followed by auxiliary verb (e.g. *was*) and word *entitled* and prepositional phrase with head word *to*) in training data. But the source model makes mistakes on these sentences in training data by not predicting *A2* where it should. The reason for this is that the label *A2* and the verb *entitle* are relatively infrequent in training data. However, they are frequent in test data from a different domain. As a result, the model makes a lot of mistakes and the performance on the test domain drops significantly.

So for the cases where either the gold semantic role is infrequent or the verb is infrequent in training data, we can learn rules from the training data to supplement the source model. Consider the following sentence "But it did not allow". We will refer to this sentence as *sentence B*. In this sentence, the gold standard semantic role for the phrase *it* is *A0* for the verb *allow*. Now, let us consider the rule: given a test sentence, if the sentence contains a noun phrase that is followed by an auxiliary verb and word *entitled* and prepositional phrase with head word *to*, replace the word *it* in sentence B with that noun phrase from the test sentence. So if the test sentence is sentence A, the rule will generate a transformed sentence "But Mr. Mckinley did not allow". Next we will apply the source model on the transformed sentence. Since the verb *allow* is frequent in training data, we hope that the source model will correctly predict the semantic role *A0* for *Mr. Mckinley* for the transformed sentence. In sentence A, the gold standard semantic role of "Mr. Mckinley" was *A2*. So the transformation rule has changed the label of the phrase from *A2* to *A0*. However, if this change of label is always consistent for all cases where the rule applies (for all noun phrases that are followed by an auxiliary verb and word *entitled* and prepositional phrase with headword *to*), we can include a post-processing step of role mapping from *A0* to *A2* in the rule itself. Intuitively, we

want to discover that whenever in a test sentence, a noun phrase is followed by an auxiliary verb and word *entitled* and prepositional phrase with head word *to*, that noun phrase behaves similarly to the word *it* in sentence B. However, since the labeling of arguments was done independently for each verb, this may induce a change of roles for that noun phrase but it can remedied if the change is deterministic. These rules essentially capture equivalence of phrases with respect to verbs and contexts. They can be thought of generalizations of most of the transformations discussed before (e.g. replacing a word with its synonym or replacing a verb with another verb in the same VerbNet cluster).

In the following, we discuss what is the syntax of transformation rules, how to learn them from training data and how to use them in the test phase.

Phrase Representation: We represent each sentence in terms of a sequential list of phrases from the parse tree of that sentence. For sentence A, the parse tree is given in figure 4.3.

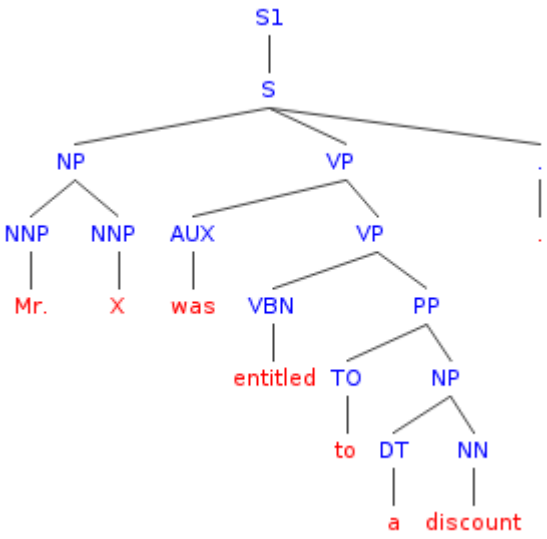


Figure 4.3: Parse Tree for Sentence *Mr. X was entitled to a discount*

The phrases that are either siblings to the verb *entitle* or siblings to the parent of the verb *entitle* are *Mr. X*, *was*, *to a discount* and . . We take the index of the verb as 0 and the index of the phrases left of the verb as negative and the index of the phrases right of the verb as positive. Therefore, the indices of the phrases *Mr. X*, *was*, *entitled*, *to a discount* and . are respectively -2, -1, 0, 1, 2. For each phrase, we keep the head word of the phrase and also node label of that phrase

in the parse tree. The parse tree label and headwords of the phrases *Mr. X*, *was*, *to a discount* and . are respectively (NP, X), (AUX, was), (PP, to) and (., .). Therefore, the phrase representation of the sentence becomes [-2, NP, X] [-1, AUX, was] [0, V, entitled] [1, PP, to] [2, ., .].

Similarly, phrase representation of the sentence B is

[-4, CC, But] [-3, NP, it] [-2, AUX, did] [-1, RB, not] [0, VB, allow] [1, ., .]

We can express the premise "if a noun phrase is followed by an auxiliary verb and word *entitled* and prepositional phrase with head word *to*" in terms of the notion of phrase representation as follows

[-2, NP,] [-1, AUX,] [0, , entitled] [1, , to]

This can alternatively be stated as

node label of phrase at index -2 = NP $\wedge$ node label of phrase at index -1 = AUX $\wedge$ headword of phrase at index 0 = entitled $\wedge$ headword of phrase at index 1 = to

For a test sentence, if this premise holds, we will take the phrase at index -2 (e.g. Mr. X) and put it in place of phrase *it* in the sentence B. The resulting transformed sentence will be "But Mr. X did not allow". This changes the semantic role of Mr. X from *A2* in test sentence to *A0* in the transformed sentence. However, if this change is always deterministic, we can map the role from A0 to A2, in a post processing phase of this rule. Therefore, we need a function that maps from the role in the transformed sentence to the role in the test sentence. In this case, the function is: if the predicted role of the phrase is A0, map it back to A2, otherwise keep it the same.

The algorithm for transformation rule generation has been shown in Algorithm 2. This algorithm calls Algorithm 1 for generating the initial set of rules.

As mentioned before, we generate rules for each (verb, semantic role) pair for which either the verb or the semantic role was infrequent in training data. For each such pair, we first generate some rules using Algorithm 1. R is the set of rules which is set to empty at beginning (Line 1) and each time a new rule is generated, it is added to R (Line 12). Suppose we want to generate rules for semantic role *A2* and verb *entitle*. Then we identify all parse tree nodes in the training data that have the gold semantic role of *A2* for verb *entitle* (Line 2). For example, we can select the phrase *Mr. Mckinley* from sentence A in training data. Then we create the phrase representation of this sentence and mark the index of *Mr. Mckinley* (Line 3). Next, we search for all parse tree nodes

Algorithm 1 Algorithm for Generation of Initial Rules for verb v and semantic role r

Input: verb v_1, semantic role r, Training sentences D, Source Model M
Output: Set of initial rules R

$R \Leftarrow \phi$
for each parse tree node p_1 in training data that has the gold standard label r for predicate v_1 in sentence s_1 **do**
  $i_1 \Leftarrow$ index of p_1 in phrase representation of s_1
  for each parse tree node p_2 in training data that has the same node type as p_1 and that has the gold labeling of a frequent core argument (A0 / A1) with respect to a frequent verb v_2 in sentence s_2 **do**
    Rule premise $\Leftarrow$ Subsequence of the phrase representation of s_1 that includes only phrases that occur from p_1 to v and two phrases after v
    Create a new rule r that given a test sentence containing verb v_1, if the phrase representation of test sentence matches the rule premise, generate a transformed sentence by replacing p_2 in s_2 with the phrase at index i_1 in the phrase representation of test sentence
    for each sentence $s_3 \in D$ containing verb v_1 such that the phrase representation of s_3 matches the rule premise **do**
      generate a transformed sentence by replacing p_2 in s_2 with the phrase at index i_1 in the phrase representation of s_3
      annotate the transformed sentence with source model and let the predicted semantic role for the replacement phrase in the transformed sentence be r_2 and let the gold standard semantic role for the replacement phrase in s_3 be r_1
      add the pair (r_2, r_1) to the role transformation function of the rule if neither r_1 nor r_2 was added to the transformation function before
    end for
    add the rule to R
  end for
end for

Algorithm 2 Algorithm for Generating Transformation Rules

Input: predicate v, semantic role r, Training sentences D, Source Model M
Output: set of rules R

$R \Leftarrow$ Algorithm 1 (v, r, D, M)
repeat
  $J \Leftarrow$ CreateNewRules(R)
  $K \Leftarrow R \cup J$
  sort K based on accuracy, support
  select some rules $R \subset K$ to cover all the phrases in training data with semantic role r for predicate v a certain number of times
until all rules in R have been expanded before
return R

in training data that have the same phrase type as *Mr. Mckinley* and that have a gold semantic role that is more frequent with respect to a more frequent verb (Line 4). For example, this list will contain the phrase *it* from the sentence B, because, *it* in sentence B has the parse tree node label of a noun phrase similar to the phrase *Mr. Mckinley* and at the same time, it has a gold semantic role of *A0* which is more frequent and the relevant verb is *allow* which is also frequent. Next, for each such phrase (e.g. *it*), we create a rule. The premise of the rule is the subsequence of the phrase representation of sentence A from the verb to the phrase and two phrases after the verb (Line 5). In Line 6, we update the rule components. In Line 7-11, we find the role mapping function. To do this, we go over all sentences in training data and find the phrases that match the rule premise. For example, suppose a sentence in training data is "Every senior employee is entitled to retirement benefits". Applying the rule will generate a transformed sentence "But every new employee did not allow". In Line 9, we will apply source SRL model on this transformed sentence. If the predicted semantic role for *every new employee* is *A0*, we will add the mapping (A2, A0) to the transformation function (Line 10).

Algorithm 2 shows the complete algorithm. In Line 1, we call Algorithm 1 to generate the set of initial rules. Then we search the space of rules from specific to general and select a set of rules that can cover all phrases of a semantic role for a specific verb.

For example, for verb *entitle* and role *A2*, an initial rule is:
rule premise = [-2, NP, X] [-1, AUX, was] [0, V, entitled] [1, PP, to] [2, ., .], phrase with index -2 will replace the phrase *it* in the sentence "But it did not allow".

Next we find new rules from these rules. From each rule, we create new rules that are more generally applicable, by dropping some conditions in the rule premise. For example, consider the initial rule
rule premise = [-2, NP, X] [-1, AUX, was] [0, V, entitled] [1, PP, to] [2, ., .], phrase with index -2 will replace the phrase *it* in the sentence "But it did not allow".

We can create a new rule by dropping the condition that the phrase at index -2 in the above rule must have headword X. The new rule is
rule premise = [-2, NP,] [-1, AUX, was] [0, V, entitled] [1, PP, to] [2, ., .], phrase with index -2 will replace the phrase *it* in the sentence "But it did not allow".

The new rule is more general than the initial rule since the premise of this new rule will match the sentence "Mr. Y was entitled to a discount" but the premise of the initial rule will not match this sentence.

We create new rules by dropping either the headword condition or the phrase type condition for any of the phrases in the rule premise independently. In Line 7, the rules are sorted by decreasing order of accuracy, support (the number of sentences to which the rule applies). In Line 8, a set of rules are selected to cover all occurrences of the semantic role r with the predicate v a specific number of times. This process continues until no new rules are found.

4.5 Joint Inference

The transformation functions transform an input sentence into a set of sentences T. From each transformed sentence t_i, we get a set of argument candidates S_i. Let $S = \bigcup_{i=1}^{|T|} S_i$ be the set of all arguments. An argument classifier assigns scores for each argument over the output labels (roles) in S that is then converted into a probability distribution over the possible labels using the softmax function (Bishop, 1996). Note that multiple arguments with the same span can be generated from multiple transformed sentences.

First, we take all arguments from S with a distinct span and put them in S'. For each argument *arg* in S', we calculate scores over possible labels as the sum over the probability distribution (over output labels) of all arguments in S that have the same span as *arg* divided by the number of sentences in T that contained *arg*. This results in a set of arguments with distinct spans and, for each argument, a set of scores over possible labels. Following the joint inference procedure in Punyakanok et al. (2008), we want to select a label for each argument such that the total score is maximized subject to some constraints. Let us index the set S' as $S'^{1:M}$ where $M = |S'|$. Also assume that each argument can take a label from a set P. The set of arguments in $S'^{1:M}$ can take a set of labels $c^{1:M} \in P^{1:M}$. Given some constraints, the resulting solution space is limited to a feasible set F; the inference task is: $c^{1:M} = \arg\max_{c^{1:M} \in F(P^{1:M})} \sum_{i=1}^{M} score(S'^i = c^i)$.

The constraints used are: 1) In the final output, there can not be multiple arguments with non-null label that either overlap or contain each other. 2) For each verb, each of the labels from A0-A5 and AA can not be assigned to more than one argument. 3) C-arg means the continuation

of an argument "arg". An argument can only be labeled as "C-arg" if at least one other argument is labeled as "arg".

4.6 Experimental Results

In this section, we discuss our experimental setup for the semantic role labeling system. Similar to the CoNLL 2005 shared tasks, we train our system using sections 02-21 of the Wall Street Journal portion of Penn TreeBank labeled with PropBank. We test our system on an annotated Brown corpus consisting of three sections (ck01 - ck03).

Since we need to annotate new sentences with syntactic parse, POS tags and shallow parses, we do not use annotations in the CoNLL distribution; instead, we re-annotate the data using publicly available part of speech tagger and shallow parser[1], Charniak 2005 parser (Charniak and Johnson, 2005) and Stanford parser (Klein and Manning, 2003).

Our baseline SRL model is an implementation of Punyakanok et al. (2008), which was the top performing system in the CoNLL 2005 shared task. We omit the details of the system and refer readers to Punyakanok et al. (2008).

Results for ADUT using only the top parse of Charniak and Stanford are shown in Table 4.2. The Baseline model using top Charniak parse (*BaseLine-Charniak*) and top Stanford parse (*BaseLine-Stanford*) score respectively 76.4 and 73.3 on the WSJ test set. Since we are interested in adaptation, we report and compare results for Brown test set only. On this set, both *ADUT-Charniak* and *ADUT-Stanford* significantly outperform their respective baselines. We compare with the state-of-the-art system of Surdeanu et al. (2007). In Surdeanu et al. (2007), the authors use three models: Models 1 and 2 do sequential tagging of chunks obtained from shallow parse and full parse. Model 3 assumes each predicate argument maps to one syntactic constituent and classifies it individually. So Model 3 matches our baseline model. *ADUT-Charniak* outperforms the best individual model (Model 2) of Surdeanu et al. (2007) by 1.6% and Model 3 by 3.9%. We also tested another system that used cluster features and word embedding features computed following Collobert and Weston (2008). But we did not see any performance improvement on Brown over baseline.

[1] http://cogcomp.cs.illinois.edu/page/software

System	P	R	F1
BaseLine-Charniak	69.6	61.8	65.5
ADUT-Charniak	72.75	66.1	**69.3**
BaseLine-Stanford	70.8	56.5	62.9
ADUT-Stanford	72.5	60.0	65.7
(Surdeanu et al., 2007)(Model 2)	71.8	64.0	67.7
(Surdeanu et al., 2007)(Model 3)	72.4	59.7	65.4

Table 4.2: Comparing single parse system on Brown.

All state-of-the-art systems for SRL are a combination of multiple systems. So we combined *ADUT-Stanford*, *ADUT-Charniak* and another system *ADUT-Charniak-2* based on 2nd best Charniak parse using joint inference. In Table 4.3, We compare with Punyakanok et al. (2008) which was the top performing system in CoNLL 2005 shared task. We also compare with the multi parse system of Toutanova et al. (2008) which uses a global joint model using multiple parse trees. In Surdeanu et al. (2007), the authors experimented with several combination strategies. Their first combination strategy was similar to ours where they directly combined the outputs of different systems using constraints (denoted as *Cons* in Table 4.3). But their best result on the Brown set was obtained by treating the combination of multiple systems as a meta-learning problem. They trained a new model to score candidate arguments produced by individual systems before combining them through constraints (denoted as *LBI* in Table 4.3). We also compare with Huang and Yates (2010) where the authors retrained an SRL model using HMM features learned over unlabeled data from WSJ and Brown.

System	P	R	F1	Retrain
(Punyakanok et al., 2008)	73.4	62.9	67.8	×
(Toutanova et al., 2008)	NR	NR	68.8	×
(Surdeanu et al., 2007) (Cons)	78.2	62.1	69.2	×
(Surdeanu et al., 2007) (LBI)	81.8	61.3	70.1	×
ADUT-combined	74.3	67.0	**70.5**	×
(Huang and Yates, 2010)	77.0	70.9	**73.8**	✓

Table 4.3: Comparison of the multi parse system on Brown.

Table 4.3 shows that *ADUT-Combined* performs better than *Surdeanu et al. (2007) (Cons)* when individual systems have been combined similarly. We believe that the techniques in Surdeanu

et al. (2007) of using multiple models of different kinds (two based on sequential tagging of chunks to capture arguments whose boundaries do not match a syntactic constituent) and training an additional model to combine the outputs of individual systems are orthogonal to the performance improvement that we have, and applying these methods will further increase the performance of our final system, which is a research direction we want to pursue in future.

We did an ablation study to determine which transformations help and by how much. Table 4.4 presents results when only one transformation is active at a time. We see that each transformation improves over the baseline.

Transformation	P	R	F1
Baseline	69.6	61.8	65.5
Replacement of Unknown Words	70.6	62.1	**66.1**
Replacement of Predicate	71.2	62.8	**66.8**
Replacement of Quotes	71.0	63.4	**67.0**
Simplification	70.3	62.9	**66.4**
RuleTransformation	70.9	62.2	**66.2**
Sentence Split	70.8	62.1	**66.2**
Together	72.75	66.1	**69.3**

Table 4.4: Ablation Study for ADUT-Charniak

The effect of the transformation of *Replacement of Predicate* on infrequent verbs is shown in Table 4.5. This transformation improves F1 as much as 6% on infrequent verbs. It is interesting to note that the performance of verbs that had frequency 0 is higher than the performance of verbs that had frequency more than 0. We found that this is due to the nature of the data set. The non-core arguments are annotated with similar semantic roles across all verbs. For example, temporal argument, location argument. So the source model may perform well on annotating semantic roles for infrequent verbs if they occur with semantic roles from these types.

Frequency	Baseline	Replacement of Predicate
0	64.2	**67.8**
less than 3	59.7	**65.1**
less than 7	58.9	**64.8**
all predicates	65.5	**66.78**

Table 4.5: Performance on Infrequent Verbs for the Transformation of *Predicate Replacement*

The running time for *ADUT-Charniak* on Brown set is 8 hours compared to SRL training time of 20 hours (speedup is 2.5). Average number of transformed sentences generated by *ADUT-Charniak* for every sentence from Brown is 36. The times are calculated based on a machine with 2x 6-Core Xeon X5650 Processor with 48G memory. Important thing to note here is that the running time of *ADUT-Charniak* depends on the size of the test set only and it has no dependence on the time required for SRL training, whereas, any adaptation solution that requires retraining the SRL system will at least require the SRL training time of 20 hours, regardless of the size of the test set. For example, if the test set size is 0.1 times the size of the Brown set, *ADUT-Charniak* will have a running time of 0.8 hours and will have a speed up of at least 25 over a solution that includes the SRL training time. However, per sentence processing time of *ADUT-Charniak* is larger than the per sentence processing time of the original SRL system. So after the SRL system is retrained for new domain, it will process sentences faster than *ADUT-Charniak*. So if the test set size is very large, the solutions involving the training of SRL system will require less time overall.

4.7 Summary

This chapter proposes a novel solution to domain adaptation: applying a diverse range of transformations on input text and combining the predictions on the transformed text. Since the model need not be changed and only input text is changed, this is a form of *adaptation of text* instead of *adaptation of model*. This is a new direction in domain adaptation research since all contemporary methods require training a new model for every new domain. Since in natural language processing, models are often complex and pipeline, using the same models across all domains is very advantageous. Our experiments on adaptation of a state-of-the-art semantic role labeling system from newswire to literature show that our proposed approach is very effective, even though we use the same model from the newswire domain.

Chapter 5

Prior Knowledge Driven Adaptation

The previous chapter discussed the method of domain adaptation using transformation where some transformations were derived from knowledge bases. In this chapter, we further extend this paradigm of using prior knowledge by making use of declarative prior knowledge from the target domain. As before, we do not change the source domain model. Instead, we make predictions on the target domain text using the source domain model such that the predictions are consistent with the constraints derived from the prior knowledge of the target domain. Since the source domain model remains unchanged, this chapter extends the proposed paradigm of adaptation without training.

5.1 Introduction

One important resource that has largely been ignored in domain adaptation efforts is prior knowledge of the difference between the source and the target domain. Moreover, such prior knowledge is easy to collect and is available for almost all domains (e.g. annotation manuals are available for every domain). Prior knowledge may be available about the content of the domain like the vocabulary or structure of the sentences or styles of text. For example, transcribed text usually does not have any capitalization that is present in manually written text and this information is often known a priori. Prior knowledge may also be available about the annotation differences between the source and the target domain. For example, the names of all entities are annotated as proper nouns in the WSJ domain whereas the names of genes are annotated as common nouns in the Biomedical domain. As another example, in the CoNLL 2007 shared task of domain adaptation (Nivre et al., 2007) for dependency parsing, there were significant annotation differences across the source and target domain data. The participating teams did not have access to any labeled data in the target domain and so they could not learn a model to account for the annotations in the target domain.

In the end, no team could improve the results substantially above the result obtained by the source domain model applied directly to the target domain data.

Over the years, many adaptation algorithms have been proposed in the literature (Finkel and Manning, 2009; Chelba and Acero, 2004; Daumé III, 2007; Jiang and Zhai, 2007a; Huang and Yates, 2009; Blitzer et al., 2006). But these techniques do not have any information about the difference between annotations across the source and target domain. To our knowledge, all the existing adaptation algorithms do **not** use prior knowledge about the target domain to improve adaptation.

Contributions

- We show the importance of prior knowledge about the target domain for adaptation. We show that prior knowledge is often cheap to obtain and using it can provide significant improvements.
- We present our findings on the best method for applying prior knowledge.
- We compare prior knowledge with labeled data and show that prior knowledge is more effective than labeled data in the target domain.

On the tasks of POS tagging and semantic role labeling, we show that our framework for incorporating prior knowledge (Prior Knowledge Driven Adaptation (PDA)) can lead to significant improvements.

5.2 Tasks and Datasets

We evaluate PDA on two natural language tasks: POS tagging and semantic role labeling. POS tagging assigns a part of speech tag to each word in a sentence. Semantic role labeling assigns semantic roles to different parts of a sentence for each predicate in the sentence. Both these tasks are critical for natural language understanding and so adapting them is very important.

POS Tagging Adaptation One of our experiments is on the task of POS tagging. The source domain is the WSJ domain and the target domain is the Biomedical domain. Specifically, we

follow the experimental setting from Jiang and Zhai (2007a). In that paper, the authors use 6166 WSJ sentences from Sections 00 and 01 of Penn Treebank as the training data and 2730 PubMed sentences from the Oncology section of PennBioIE corpus as the target domain data[1]. It turns out (Table 5.1) that the model trained using only Section 00 and 01 from Penn Tree bank yields accuracy of 96.8 on WSJ test set. This is very close to the state-of-the-art result of 97.3 using models trained over all sections from Penn Tree bank. Since we wanted to compare our results with those of Jiang and Zhai (2007a), we trained models using Section 00 and Section 01 only similar to theirs. The learning algorithm we used is Structured SVM. We use as features the current word, previous and next word, prefixes and suffixes of up to length 3, capitalization, hyphenation and presence of number in the current word.

SRL adaptation The second set of experiments are conducted on the task of semantic role labeling (SRL) (Carreras and Màrquez, 2005). The source domain is the WSJ domain and the target domain is the Ontonotes news section. Specifically, we use Section $02 - 21$ from Penn TreeBank labeled with PropBank as training data and Section 00 of voa (Voice of America) of Ontonotes release 4.0 as the testing data. The baseline SRL model is an implementation of Punyakanok et al. (2008) which was the top performing system in the CoNLL 2005 shared task (Carreras and Màrquez, 2005). The first phase of the system is *Pruning*. In this phase, we use the heuristic from Xue and Palmer (2004) for pruning away unlikely arguments. The second phase is *Argument Identification*, which utilises a binary classifier to decide if a candidate argument supplied by the pruning phase is an argument or not. The third phase is *Argument Classification*. In this phase, a multi class classifier is used to predict the types of the argument candidates. The final phase is *Inference*, where a global consistent labeling of argument candidates is produced subject to some linguistics and structural constraints, such as *arguments cannot overlap*, or *each verb can take at most one core argument*. The features used by the SRL system are either based on the predicate (like lemma, POS tag, voice, subcategorization, verb class, negation, modality etc.), or based on the argument (like phrase type, head word or POS tag of the head word, length etc.), or based on the relation of the argument to the predicate (like path in the parse tree from the argument to the predicate, syntactic frame etc.). The system also uses conjunctions of every two features. For more

[1] http://bioie.ldc.upenn.edu/publications/latest release/data/

details, interested readers are referred to Punyakanok et al. (2008).

In the training data of WSJ, semantic role annotation is available for all verbs except *be* verbs e.g. *am, is, are* etc. But in Ontonotes, semantic roles are also annotated for these verbs. Since some semantic roles (core arguments $A0-A5$) depend on the verb instead of being verb independent, the model trained over WSJ performs poorly on Ontonotes since it did not see any labeled data for *be* verbs.

The performances of our baseline models for POS tagging and SRL are shown respectively in Table 5.1 and Table 5.2. In both tasks, our systems achieve state-of-the-art results when tested on the same domain (WSJ). Table 5.1 also shows the performance of the baseline model of Jiang and Zhai (2007a) in the Bio domain. In SRL data, predicates are given for each sentence and the system has to predict semantic roles for each predicate. In WSJ, *be* verbs are not marked as predicates but in Ontonotes, they are marked. So the system predicts some semantic roles for *be* verbs in Ontonotes and so the performance is not 0 , despite not observing any labeled data with *be* verb.

System	WSJ	Bio
Baseline	96.8	86.2
(Jiang and Zhai, 2007a)	NR	86.3

Table 5.1: Results (F1) of Baseline model of POS Tagging

System	all verbs WSJ	all verbs Ontonotes	be verbs Ontonotes
Baseline	76.4	55.2	15.5

Table 5.2: Results (F1) of Baseline model of SRL

5.3 Motivating Examples

On the task of POS tagging, certain differences exist between the source and target domain annotations. From the *annotation manual*[2] for WSJ, we find that hyphenated compounds are treated as either adjective or adverb. But from the *annotation manual*[3] for the Bio domain, hyphenated

[2] ftp://ftp.cis.upenn.edu/pub/treebank/doc/tagguide.ps.gz
[3] http://bioie.ldc.upenn.edu/wiki/index.php/POS_tags

compounds are gene names and so are labeled as common nouns. The following knowledge is directly extracted from the annotation manual for Bio domain:

1. Hyphenated words should be tagged as common nouns (NN).
2. Digit-letter combinations should be tagged as common nouns (NN).
3. Hyphens should be tagged as HYPH.

The annotation manual also says: *There are few proper nouns in these files, mostly names of individual persons or organizations. Proper names of persons, organizations, places, and species names should be tagged as NNP.Conversely, trademarked drugs or other substances that are capitalized are tagged as NN. This rule also includes gene names and symbols, abbreviations for diseases, and other pieces of biochemical jargon.* Motivated by this, we also add the following knowledge:

1. If any word unseen in the source domain is followed by the word *gene*, it should be tagged as a common noun(NN).
2. If any word does not appear with proper noun tag (NNP) in training data, predict the tag NN instead of NNP for that word.
3. If any word does not appear with proper noun plural tag (NNPS) in training data, predict the tag NNS instead of NNPS for that word. Equivalently, we map the tag NNP to NN.

On the task of SRL, the model trained over WSJ is unaware of the semantic role pattern for *be* verbs. But *the frame file* for *be* verbs in Ontonotes (included in Ontonotes release 4.0) contains significant information. The knowledge used in our experiments is derived from the frame file and is listed below:

1. If a *be* verb is immediately followed by another verb, it can not have any core argument.
2. If a *be* verb is immediately followed by the word *like*, it can have core arguments of $A0$ or $A1$.
3. Otherwise, it can have core arguments of $A1$ or $A2$.

The knowledge above is derived from the frame file for three different senses of the *be* verb and annotation decisions for each case. So knowledge used in our experiments is readily available and comes at little cost.

5.4 Inference w/ Prior Knowledge

In this section, we try to answer the first research question about the usefulness of prior knowledge.

For both POS tagging and SRL, we use CCM with the local model and global inference based approach; that is, we learn models ignoring the prior knowledge and only enforce them during testing. Algorithm 3 formally describes the algorithm. It is worth noting that the baseline model of Punyakanok et al. (2008) uses an inference step with some structural and linguistic constraints, as discussed in Section 5.2. The constraints from the target domain prior knowledge are added on top of these existing constraints. Results for this case are listed in Table 5.3.

From Table 5.3, we see that prior driven adaptation with knowledge (PDA-KW) improves 5% for POS tagging, 3% for all verbs and 18% for *be* verbs. Note that these results are obtained **without any retraining** whereas all traditional adaptation algorithms need to retrain their model either using labeled or unlabeled data from the target domain.

Algorithm 3 Inference with Prior Knowledge without *Retraining*

Input: Source Model w_s, Prior knowledge ρ, Test Set T
Output: Class Labels for T

1: **for** each sentence $\mathbf{x} \in T$ **do**
2: predict the label $\mathbf{y}^*$ according to

$$\mathbf{y}^* = \arg\max_{\mathbf{y} \in \mathcal{Y}} w^T \phi(\mathbf{x}, \mathbf{y}) + \sum_{j=1}^{|C|} \rho_j * C_j(\mathbf{x}, \mathbf{y}) \tag{5.1}$$

3: **end for**

System	POS	SRL	
		All verbs	Be verbs
Baseline	86.2	55.2	15.5
PDA-KW	**91.8**	**58.7**	**34.1**

Table 5.3: Comparsion of results(F1) for the Baseline Model versus PDA-KW

5.5 Self-training w/ Prior Knowledge

In this section, we try to answer the second research question about how to apply prior knowledge. We evaluate self training with source domain labeled data (gold standard) and target domain

labeled data (labeled by the source domain model) as a way to incorporate prior knowledge and evaluate the performance gain by training vs no-training. The insight is that our knowledge is very accurate but applies rarely (high precision but low recall). So we want to add the sentences where the knowledge changed predictions to the training set and learn a new model, so that the new model can generalize from the sentences where knowledge applied. The algorithm is given in Algorithm 4.

Algorithm 4 Self training with Prior Knowledge

Input: Source Model w_s, Prior knowledge ρ, Source Domain Labeled Data D_s, Target Domain Unlabeled Data D_u, Test Set from Target Domain D_t

Output: Class Labels for D_t

1: $D_l \Leftarrow D_s$
2: **for** each sentence $x \in D_u$ **do**
3: predict label y_a for x by w_s
4: predict label y_b for x by solving Eq. 2.11 with w_s and ρ
5: **if** $y_a \neq y_b$ **then**
6: $D_l \Leftarrow D_l \cup (x, y_b)$
7: **end if**
8: Train a new model w_t over D_l
9: **end for**
10: **for** each sentence $x \in D_t$ **do**
11: predict label y for x by solving Eq. 2.11 with w_t and ρ
12: **end for**

When we do not have a significant amount of unlabeled data D_u, we set $D_u = D_t$. For example, in the POS tagging experiment, a significant amount of D_u is not available, so we set $D_u = D_t$. For the SRL experiment, we had a significant amount of unlabeled data from the target domain. Specifically, we used Section 01 of the VOA (Voice of America) portion of Ontonotes as unlabeled data D_u and Section 00 of the voa portion of Ontonotes as test data D_t. The resulting model is denoted as *PDA-ST* in Table 5.4. We also experiment with the scenario where we self train on target domain unlabeled data without prior knowledge. In terms of Algorithm 4, in line 8, instead of using (x, y_b), we use (x, y_a). The resulting model is denoted as *Self-training* in Table 5.4.

For self-training with SRL, there is no labeled data in the WSJ domain for *be* verbs. So in Algorithm 4, $D_s = \phi$. Since training an SRL model takes a long time, we train a new model only for the argument classification phase, and only for *be* verbs. So during testing of *Self-training* or *PDA-ST*, we use the baseline model of the argument identifier phase for all verbs, including *be*

verbs. Then we use the baseline model of the argument classification phase for all verbs other than *be* verbs and the newly learned model for *be* verbs. For learning the new model, all features of the baseline model for the argument classification phase are used with one additional feature, which is the label predicted by the baseline model for that argument.

Since we assume in all experiments that no labeled data is available in the target domain, we do **not** perform any parameter tuning. For self training of POS tagging, the C parameter is set to 1, and for SRL of *be* verbs, the C parameter is set to the value of 0.4. These values are the same as those used by the baseline model.

5.6 Experimental Results

From Table 5.4, we see that *PDA-ST* performs the best for both the tasks. Also *PDA-ST* performs slightly better than *PDA-KW*.

System	POS	SRL	
		All verbs	Be verbs
Baseline	86.2	55.2	15.5
PDA-KW	91.8	58.7	34.1
PDA-ST	**92.0**	**59.0**	**35.7**
Self-training	86.2	55.0	14.0

Table 5.4: Comparsion of results(F1) of different ways of incorporating knowledge

In Table 5.5, we compare our final model of POS tagging with Jiang and Zhai (2007a). In the first setting of Jiang and Zhai (2007a), the authors learn a model over 300 labeled sentences from the target domain. Then they gradually remove sentences from the source domain training set where gold label differ from the label predicted by the target domain model. We compare with the best result they obtain in this setting (denoted as Jiang and Zhai (2007a)-1 in Table 5.5). In the second setting, they use all (2730) labeled sentences from the target domain and learn a model over weighted combination of source domain and target domain labeled data. We compare with the best result they obtain in this setting (denoted as Jiang and Zhai (2007a)-2 in Table 5.5).

From Table 5.5, we see that even without using any labeled data from the target domain, *PDA-ST* significantly outperforms Jiang and Zhai (2007a) when little labeled data are available

System	POS	Amount of Target Labeled Data
PDA-ST	**92.0**	0
(Jiang and Zhai, 2007a)-1	87.2	300
(Jiang and Zhai, 2007a)-2	**94.2**	2730

Table 5.5: Comparsion of results(F1) for POS Tagging with Jiang and Zhai (2007a)

and recovers 72% of accuracy gain that Jiang and Zhai (2007a) had after adding a lot of labeled data from the target domain and using sophisticated weighting schemes.

For the task of SRL adaptation, we are not aware of any work that reports results of adaptation from WSJ to Ontonotes and so we are unable to compare with others.

5.7 Summary

In this chapter, we introduce *PDA*, a new framework of adaptation based on prior knowledge of the target domain. We want to emphasize that prior knowledge about the domains is available for many domains and such knowledge is usually cheap to collect and can result in significant improvements. We show that using prior knowledge can give results that are competitive with using labeled data from the target domain. We also evaluate self training with prior knowledge (*PDA-ST*) as a way of incorporating prior knowledge and show that *PDA-KW* without training performs as good as *PDA-ST*.

Chapter 6

Adaptation Using Learned Constraints

The previous chapters focused on faster domain adaptation by avoiding the retraining of the source domain model and augmenting the inference phase with some transformations derived from various resources and declarative knowledge derived from the target domain. For many tasks or domains, such resources or knowledge may not be available. The focus of this chapter is on extending the paradigm of adaptation without training proposed in the previous chapters for these scenarios where resources and knowledge about the target domain are not available, but a significant unlabeled data from the target domain and other domains are available.

6.1 Introduction

Scalable adaptation as proposed in the previous chapters consists of keeping the source model unchanged and modifying the inference phase to take into account different transformations of the input sentence or target domain knowledge. In Chapter 4, we discussed how we can automatically generate sentences from an input sentence of the target domain such that the generated sentences are closer to the source domain and then apply the source domain model on the generated sentences and combine the outputs in the inference step. The benefit of this is that we can keep the source model unchanged. In Chapter 5, we discussed how we can use knowledge about the target domain such that, given an input sentence from the target domain, the source model can be forced to produce an output that is consistent with the knowledge. In this case also, we can use the same model from the source domain. However, the applicability of these two approaches require availability of resources for the task at hand or human experts that can provide knowledge about the target domain. In many scenarios, such guidance is not available. For example, we may not have gazetteers for target domain (company emails) for the task of named entity recognition. Often

a large amount of unlabeled data from the target domain is readily available, which the previous approaches cannot make use of.

In the absence of knowledge or resources, we can attempt to learn this knowledge automatically from the unlabeled data. Almost all prior works on inference with prior knowledge in the literature (Punyakanok et al., 2008; Roth and Yih, 2004; Riedel and Clarke, 2006) assume that the knowledge is available from an expert. In this chapter, we show that it is possible to learn knowledge automatically from the unlabeled data and then we can use this knowledge during inference on the target domain test set. We show that knowledge can be derived using a bootstrapping process from the unlabeled data of the target domain or from an intermediate domain between the source and the target domain. This knowledge needs to be learned for every target domain and can be used in the inference phase with the same source domain model. Experimental results on multiple named entity recognition tasks and multiple data sets show that this is indeed a highly effective method of adaptation.

Contributions

- We introduce the notion of *strongly discriminative features (SDFs)* for each label. We show how to identify a set of SDFs from the source domain labeled data and then augment this set of SDFs with target domain specific SDFs, iteratively using a *bootstrapping algorithm*. Using these SDFs, we identify the label of phrases in the target domain. Then, for each phrase, we generate constraints that force the phrase to take the corresponding label across all occurrences of that phrase throughout the entire target domain.

- We show how to extract constraints automatically from a large, intermediary, unlabeled corpus that we process with the source-trained model.

The rest of this chapter is structured as follows. Section 6.2 motivates the need of constraints for adaptation of named entity recognition systems and Section 6.3 presents the algorithm. Experiments demonstrating the utility of the proposed approach are presented in Section 6.4. Section 6.5 concludes the chapter.

6.2 Motivating Examples

The question is: How can we learn constraints automatically from target domain unlabeled data ? Below we outline some key observations made in the context of a named entity recognition task. These observations, although made on the task of named entity recognition task, are very general. These observations provide valuable insights based on which we will come to an algorithm for learning constraints useful for domain adaptation of NER systems.

Observation I In the target domain, there is often a need to enforce the "one sense per discourse" heuristic. The reason is that only a few of the contexts in which a word / phrase appears are discriminative, but most are not. For example, consider the task of part-of-speech tagging. In the domain of technical documentation of computer monitors, the gold standard tag of the word *monitor* is almost exclusively *noun*. For a POS tagging model trained on news wire, it would be easier to tag the word *monitor* as a *noun* in the sentence " The monitor will open" but it might be difficult to tag the word *monitor* as a *noun* in the sentence " The monitor problem persists" since the word *monitor* in the latter sentence can be either *noun* or *adjective*. So we need to tag the word *monitor* in easy contexts and then we need to transfer the tag to harder contexts. Again, consider the task of named entity recognition. Suppose we train on text from the yeast species and test on text from the fly species. The gene name *Apc* does not occur in source text but appears in target text. The source domain classifier will predict *Apc* as the name of a gene in contexts like "Apc genetic expression ..." but it will not predict *Apc* as a gene name in other contexts like "Apc has been shown ...", because in the later case, words surrounding the word *Apc* are not strong indicators of a gene name. Although the source domain model can make similar errors for a test set from the same domain, the magnitude of this type of errors is much larger in domains other than the source domain. This example illustrates that there is a need for constraints to enforce coherence of predictions for the same phrase / word across the entire domain. This is equivalent to the Yarowsky's one sense per discourse heuristic (Yarowsky, 1995).

Observation II There is a need to identify *generalizable* strongly discriminative features (SDFs) from the source domain. Often the most discriminative features from the source domain are specific

to source domain. As a result, when we train a classifier from this source domain, this classifier assigns large weights to these features. In the target domain, since these features are absent, the classifier's performance drops. However, there are other features in the source domain that generalize better to other domains, but they may not be as discriminative as the aforementioned group of features and hence do not get proper weights in the source domain classifier. For example, suppose we train a NER system from the labeled text of fly species and test it on the text of yeast species. *Apc* and *photolyase* are names of two genes in the fly domain. As these two genes appear in many sentences, lexical features capturing these words get the highest weights in the classifier trained from labeled data of that domain. However, these gene names do not appear in the text of the yeast species, so the classifier trained from fly species suffers on the target domain text of yeast species. But features like "next-word=gene" and "next-word=expression" are general pattern features, that appear in both the source and the target domain text. Unfortunately, they get small weights in the source classifier because of the presence of more discriminative features that are actually domain specific.

Observation III There is also a need to identify *target domain specific* SDFs automatically without any labeled data. For example, when a NER system trained on news is used on email data (specifically emails related to meetings), the distribution of the entities' occurrences is very different. In these emails, there are many lists of names of participants in the meeting mentioned in the email. As a result, "previous word = & and the label assigned to the word before the previous word by NER is PERSON" is a strong feature for the label of current word being a PERSON. But this feature was not significantly present in news text. So for successful adaptation to a target domain, we need to identify target domain specific strongly discriminative features (SDFs).

Observation IV Adaptation can be very difficult when the target and source domains are *very different*. For example, consider training on grammatical news text and testing on text from conversational telephone speech. In this case, unlabeled text from target domain is very noisy that makes adaptation very difficult. So there is a need to consider text beyond that available from target domain only, specifically we need to consider text that sits between the source and target domain and use it for successful adaptation to target domain.

Observation V For information extraction tasks, when a model trained on one domain is tested on a different domain, a common problem is the drop of recall. For example, a named entity recognition system trained on the source domain fails to mark a significant fraction of the entity names in target domain. Most of the entity names were *unseen* in the source domain. So recall drops, leading to the drop of F1.

6.3 Algorithm for Learning Constraints

6.3.1 Intermediate Domain Constraints

As discussed in Observation IV of Section 6.2, adaptation can be very difficult if the source and target domains are significantly different. For example, consider snippets of text from two of the target domains used in our experiments: conversational telephone speech and emails in Table 6.1. The source model was trained from annotated news domain text.

Conversational Telephone Speech	Emails
Suzanne Hi I 'm **Suzanne** from Philadelphia. **Sara** Hi , this is **Sara**	on 01/24/2001 07:05:50 AM To: **Dave** .fuller@enron.com Oslo Doorstep Meeting for **Cindy** Mime-Version: 1.0
Suzanne Hi I 'm **Suzanne** from Philadelphia. **Sara Hi** , this is **Sara**	on 01/24/2001 07:05:50 AM To: **Dave .fuller@enron.com** Oslo Doorstep Meeting for **Cindy Mime-Version:** 1.0

Table 6.1: The upper part of the table shows target domain text with gold labels. The lower part of the table shows the prediction of NER trained from news text Only entities of type person are shown in bold.

From Table 6.1, we see that in the conversational speech domain, the punctuation between the speaker name and the sentence is missing which forces NER to make mistakes. Similarly, in the emails domain, the headers of the emails contain many capitalized words specific to email headers which forces NER to make mistakes. However, only looking in the target domain unlabeled text (as large as it may be, it will still be noisy) may not be enough for correcting these mistakes. Instead, using knowledge capturing our general belief about named entities (e.g. "names of persons should

not contain words like *Hi* and should not contain *punctuation*") can correct the mistakes shown in Table 6.1. Unfortunately, the size of the source domain labeled data may not be enough for learning these constraints automatically.

We define an intermediate domain to be a domain that covers the same topics as the target domain but should contain text written in syntactically similar way to the source domain.

We learn constraints into a simple representation language that can express both character level and word level constraints, which we found sufficient in our work:

- Character Level: $\exists$ character c in word $w \wedge w$ has label l
- Word Level: $\exists$ word $w \wedge w$ has label from a list of labels L

Character level constraints are designed to capture facts like "named entities of type PERSON are not expected to contain punctuation symbols or digits". We predict labels for the unlabeled text from the intermediate domain using the source domain classifier and take statistics. If any word containing a specific character appears over p% of the time with a certain label, we create a constraint that in the target domain, any word containing this character must be assigned that label only[1].

Word level constraints are designed to capture facts like "words like *Hi*, *Hello* are not expected to be part of person names". As before, we predict labels for the unlabeled text from the intermediate domain using the source domain classifier and take statistics. For each word type in the intermediate domain, we make a list of labels predicted by the source domain classifier for token level occurrences of that word type. For each word type, we create a constraint that in the target domain, that word type can not take any label other than those in its list across the token level occurrences of the word type in the target domain text.

6.3.2 Target Domain Constraints

The pseudo code of learning constraints from target domain is given in Algorithm 5. Below we discuss the key parts of this algorithm. For this section, the target domain unlabeled text also contains the target domain test set without the labels.

[1] p should be set to a very high value (0.99 in our experiments)

6.3.2.1 Extract Entity Words from the Source Domain

As discussed in Observation V of Section 6.2, the performance drop of an NER system stems from the failure to predict names containing words that were unseen in the source domain. So we want to characterize the set of words from the target domain that are likely to contain entity names (e.g. after removing all stop words). If the probability distribution over labels for features from the source domain is calculated from all words in the source domain (e.g. including stop words), this will not be suitable for the set of target domain candidate entity words. Instead, we create a set of candidate entity words from the source domain in an analogous way of creating the set of candidate entity words from the target domain and we estimate the distribution over labels for features from this set of source domain entity words only.

For the task of gene name recognition, for each domain d, we make a vocabulary of words appearing in the unlabeled text of that domain and remove all words that appear in a common list of English words. Furthermore, for domain d, we remove all words appearing in the other domain and words that never appear with a part-of-speech tag of noun. For the task of traditional named entity recognition like people, locations etc, we make a vocabulary of words appearing in unlabeled text of that domain and remove all words that appear in a common list of English words. The set of candidate entity words for source and target domains has been mentioned as V^s and V^t respectively in Algorithm 5. We use the superscripts s and t to denote the source and target domain respectively.

6.3.2.2 Extract Entity Phrases from the Target Domain

Named entities often consist of phrases that contain more than one words. We extract a set of potential candidate entity phrases from the target domain unlabeled text. For the task of named entity recognition with entities like people, locations etc., we run a shallow parser and a POS tagger on target domain unlabeled text. We collect all chunks predicted as "noun phrases" by the shallow parser and chunks that only contain words that have been tagged as "proper nouns" by a part-of-speech tagger. For the task of recognizing names of genes, most entities consist of only one word. So we use all words in candidate entity words V^t as the set of candidate entity phrases. This set of candidate entity phrases has been referred to as E^t in Algorithm 5.

6.3.2.3 Extract Generalized SDFs From the Source Domain

A feature f is a SDF if it exhibits high $p^s(l|f)$ for some entity label l as calculated in Equation 6.3. In addition, it is a generalized feature if it occurs with many distinct word types with label l (for any entity label l).

Strongly Discriminative Feature For each word type in the source, we create a type level feature vector that contains every feature with strength 1 if it appeared in at least one feature vector corresponding to different token level occurrences of that word type.

$$\Phi^s(w) = \cup_{T^s(w)} \phi(w) \tag{6.1}$$

Here $\cup$ stands for the operation of union of sets. $\Phi^s(w)$ is the type level feature vector for word w in the source domain. $\phi(w)$ is the token level feature vector for the word w. $T^s(w)$ is the collection of token level occurrences of w in the unlabeled text of the source domain s. We will use $\mathscr{F}$ to denote the set of all features over both source and target domain.

The feature functions used in calculating $\phi(w)$ include all discrete emission feature functions from ψ, the feature functions used in the source domain classifier. In addition, $\phi(w)$ may contain additional features as discussed in the experimental section.

Since the same word type can appear with different labels in its token level occurrences, we create a distribution over possible labels for each word type (from source domain only).

$$p^s(l|w) = \frac{count(w, l)}{\sum_{l' \in L} count(w, l')} \tag{6.2}$$

Here count(w, l) is the number of token level occurrences of w that appears with label l in the source domain labeled data and L is the set of labels. In our experiments, L consisted of labels in IO format (e.g. L={PER, O} in our person name recognition experiments in Section 6.4.2.2).

we compute the strength of each feature with each label that it appears with in the following way:

$$p^s(l|f) = \frac{\sum_{w \in V^s} \mathbb{1}_{f \in \Phi^s(w)} p^s(l|w)}{\sum_{w \in V^s} \mathbb{1}_{f \in \Phi^s(w)}} \tag{6.3}$$

For each label l, we create a list of features F_l^s that are source domain SDFs for that label.

$$F_l^s = \{f \in \mathscr{F} | p^s(l|f) \geq \tau\} \tag{6.4}$$

Details of how this threshold τ is set will be discussed later in the experimental Section 6.4.

Generalized Feature As mentioned in Observation II of Section 6.2, often the most discriminative features in the source domain do not generalize to the target domain. For example, lexical features capturing the words *Apc* & *photolyase* get the highest weights in NER trained from the labeled data of the fly species because they are names of genes and they appear many times in fly domain but never in other domains. We observed that features like "next-word = gene" are more generalized features and they apply to many names instead of a single name as in the aforementioned cases. We define the frequency of a feature by the numerator of Equation 6.3. As a result, the frequency of a feature depends on how many times it appears in feature vectors of distinct word types belonging to some entity label l and features like "current-word=Apc" get frequency 1 w.r.t. entity label *Gene* but features like "next-word=gene" retains high frequency. Then we remove all features that have frequencies below a certain threshold (5 in our experiments). Remaining features in F_l^s are generalized SDFs from the source domain.

6.3.2.4 Extract SDFs Specific to Target Domain

As discussed in Observation III of Section 6.2, there are often target domain specific features that are strongly discriminative for certain entity labels. For example, a feature capturing the pattern of lists, e.g., 'previous word = & and label predicted by the classifier for the word before previous word = PERSON' is very frequent in email meetings domain (target domain) and this feature provides accurate indication that the label of the current word should be PERSON. But, this feature is absent from the news wire domain (source domain). At this step of our algorithm, we want to extract these features using a bootstrapping process. We start by initializing this set of features to {}.

For each phrase $e \in E^t$, we compute the set of possible labels. A label is possible for a phrase only if the type level feature vector for each word in the phrase has a significant overlap with the

set of SDFs (from both source and target domain) for that label. Mathematically,

$$L_e = \{l \in L - \{O\} : \tag{6.5}$$

$$|\Phi^t(e_i) \cap (F_l^s \cup F_l^t)| \geq \mu, \forall i \in \{1, \ldots, |e|\} \tag{6.6}$$

$\Phi^t(e_i)$ is the type level feature vector for i'th word in phrase e, calculated from the target domain unlabeled text.

If $|L_e| > 1$, the label of that phrase is ambiguous. If $|L_e| = 1$, we add this (phrase, label) pair (e, L_e) to a set D^t. Details of μ will be discussed in the Experimental Section 6.4.

$$D^t = \{(e, L_e) : e \in E^t \; and \; |L_e| = 1\} \tag{6.7}$$

Since we have feature vectors at the level of word type instead of phrase type, we need to create a target domain specific distribution over labels for each word type from the (phrase, label) pairs in D^t.

Each word type in V^t that does not appear in any phrase in D^t, is assigned a uniform distribution over possible labels including the non-entity label.

$$p^t(l|w) = \frac{1}{|L|} \tag{6.8}$$

Otherwise, its label distribution is defined as

$$p^t(l|w) = \frac{|(e, l) \in D^t : e \; contains \; w|}{\sum_{l' \in L} |(e, l') \in D^t : e \; contains \; w|} \tag{6.9}$$

Example 1. *Let L={PER,ORG,O}, E^t={Bill Clinton, Clinton Administration, River}, D^t={(Bill Clinton, PER), (Clinton Administration, ORG)}*

Then, p^t(PER | Clinton) = p^t(ORG | Clinton) = 0.5
p^t(PER | Bill) = 1.0, p^t(ORG | Administration) = 1.0
p^t(PER | River) = p^t(ORG | River) = p^t(O | River) = $\frac{1}{3}$.

Similar to Eq 6.3, with $p^t(l|w)$ and $\Phi^t(w)$, we estimate the strength of each feature with respect

to the labels.

$$p^t(l|f) = \frac{\sum_{w \in V^t} \mathbb{1}_{f \in \Phi^t(w)} p^t(l|w)}{\sum_{w \in V^t} \mathbb{1}_{f \in \Phi^t(w)}} \tag{6.10}$$

Again, we select the features with strength above τ with respect to a label l and add them to the list of SDFs for label l for target domain, F_l^t.

$$F_l^t = \{f \in \mathscr{F} | p^t(l|f) \geq \tau\} \tag{6.11}$$

We then keep iterating until convergence.

6.3.2.5 Extract Constraints from the Target Domain

As discussed in Observation I of Section 6.2, there is a need to enforce coherence of prediction across all occurrences of a phrase since many contexts in which a phrase appears do not provide significant evidence for the label of that phrase. To accomplish this, after the algorithm converges, for each (phrase, label) pair (e, l) in D^t, we create one constraint that enforces the phrase e to take label l across the entire target domain, irrespective of context. This is similar in spirit to the popular heuristic of "one sense per discourse". In our case, we enforce the label of a phrase across all occurrences. The label of a phrase shows much less variability across different occurrences than the label of an individual token.

6.3.3 Prediction

We simply run the source model on the target data, and predict using this model, subject to the constraints learned in Section 6.3.1 and Section 6.3.2 using a constrained conditional model. In the CCM formulation for these experiments, we directly use external knowledge to set the constraint penalty ρ to ∞ which means that all constraints in this paper are hard constraints. Λ can be any structured prediction model. For the task of gene name recognition, we used a structured SVM. To find the best assignment of Eq 2.11, we implemented a version of Viterbi that takes the constraints into account. For the task of recognition of names of people, locations etc, we used a publicly available NER system that uses a per token classifier. In this case, the best assignment of Eq 2.11 was obtained by simply selecting, for each token, the label with the highest score from the classifier

Algorithm 5 Learning Constraints From Target Domain

Input: Source Model Λ with feature function ψ, Label Set L, non-entity label O, Feature Set $\mathscr{F}$, Learned Constraints from Intermediate Domain C_1, Target Domain Unlabeled data U^t, extended feature function ϕ, tuning parameters τ and μ

Output: Target Domain (Phrase $\Rightarrow$ Label) Constraints

Extract Source domain candidate entity words V^s and Target domain candidate entity words V^t (Section 6.3.2.1)

Extract Target domain candidate entity phrases E^t (Section 6.3.2.2)

Extract Generalized SDFs From Source Domain (Section 6.3.2.3)

for each $l \in L$ and $w \in V^s$, calculate $p^s(l|w) = \frac{count(w,l)}{\sum_{l'} count(w,l')}$, Eq 6.2
for each $l \in L$ and $f \in \mathscr{F}$, calculate $p^s(l|f) = \frac{\sum_{w \in V^s} \mathbb{1}_{f \in \Phi^s(w)} p^s(l|w)}{\sum_{w \in V^s} \mathbb{1}_{f \in \Phi^s(w)}}$, Eq 6.3
for each $l \in L - \{O\}$, calculate $F_l^s = \{f \in \mathscr{F} | p^s(l|f) \geq \tau\}$, Eq 6.4

Extract Target Domain Specific SDFs (Section 6.3.2.4)

$D^t = \{\}$, $F_l^t = \{\}$
repeat
  for each $e \in E^t$, calculate $L_e = \{l \in L - \{O\} : |\Phi^t(e_i) \cap (F_l^s \cup F_l^t)| \geq \mu, \forall i \in \{1, \ldots, |e|\}\}$, Eq 6.6
  $D^t = \{(e, L_e) : e \in E^t \text{ and } |L_e| = 1\}$, Eq 6.7
  for each word $w \in V^t$ **do**
    if $\exists (e, l) \in D^t$ such that w is a word in e **then**
      $p^t(l|w) = \frac{|(e,l) \in D^t : e \text{ contains } w|}{\sum_{l' \in L} |(e,l') \in D^t : e \text{ contains } w|}$, Eq 6.9
    else
      $p^t(l|w) = \frac{1}{|L|}$ for all $l \in L$, Eq 6.8
    end if
  end for
  for each $l \in L$ and $f \in \mathscr{F}$, $p^t(l|f) = \frac{\sum_{w \in V^t} \mathbb{1}_{f \in \Phi^t(w)} p^t(l|w)}{\sum_{w \in V^t} \mathbb{1}_{f \in \Phi^t(w)}}$, Eq 6.10
  for each $l \in L - \{O\}$, $F_l^t = \{f \in \mathscr{F} | p^t(l|f) \geq \tau\}$, Eq 6.11
until D^t converges

Extract Target Domain Constraints (Section 6.3.2.5)

return $C_2 = \{$ phrase e must have label l irrespective of context : for each $(e, l) \in D^t\}$

such that the selected label is consistent with the constraints.

6.4 Experimental Results

Constrained Conditional Adaptation Model (CCAM) has two parameters τ and μ. Together, they control how many constraints are generated for the target domain. If τ is set higher, smaller number of features in the source and target domain will make their way into the list of strongly discriminative features. If μ is higher, smaller number of phrases in the target domain will be assigned a label (because of Eq 6.6). As a result, smaller number of constraints will be generated for the target domain. If the source and target domains are similar such that we can expect the source domain classifier to perform reasonably well on target, τ and μ should be set higher. Otherwise, τ and μ should be set lower to allow generation of more constraints from the target domain. If a small amount of labeled data is available in the target domain, τ and μ can be tuned on it. In our experiments, we found that setting τ in the range of $0.9-1$ and μ in the range of $3-6$ gave significant improvements over baselines across all experiments. Our reported experimental results were obtained by setting $\tau = 0.9$ and $\mu = 3$.

6.4.1 Gene Recognition in the Biomedical Domain

Data The data we used is from BioCreAtIvE Task 1B (Hirschman et al., 2005). This data set contains three subsets of MEDLINE abstracts with gene names from three species (fly, mouse, and yeast). We used the data set from Jiang and Zhai (2006). The authors used 7500 sentences from each species for their experiments, where half of the sentences contain named entities. They further split the 7500 sentences of each species into two sections, section A of 5000 sentences and section B of 2500 sentences.

For training, we used JLIS (Chang et al., 2010) to train a structured SVM with sequential tagging. We set the soft margin parameter C = 1.0 while learning the structured SVM classifier. The features used were adopted from Finkel et al. (2005a) that achieved the state-of-the-art in gene recognition in the BioCreative Evaluation. The same system was adopted in Jiang and Zhai (2006, 2007b). Using this system, we were able to reproduce the results from Jiang and Zhai (2006, 2007b). The features used are shape features capturing patterns like capitalization, number etc., lexical features in a window, POS tags, abbreviations, suffix or prefix features etc. For details see Finkel et al. (2005a). We treat the text from each species as a separate domain. Each time we

train on the section A of one species and test it on the section A data of the remaining two species.

The feature functions ϕ contained all discrete emission feature functions from ψ, the feature functions used in the source domain classifier. In addition, we added conjunctions of every pair of features.[2]

In this task, we were unable to find appropriate intermediate domains. So we only used constraints generated from the target domain according to Section 6.3.2.

The results are shown in Table 6.2. All results reported in all experiments of this paper are phrase level F1 scores which is the standard for NER evaluation.

Source	Target	FD05	HY09	CCAM
Fly	Mouse	30.6	31.4	**53.0**
Fly	Yeast	38.6	39.0	**55.5**
Mouse	Fly	18.6	21.4	**28.5**
Mouse	Yeast	50.8	52.8	**58.2**
Yeast	Fly	10.4	11.2	**36.5**
Yeast	Mouse	29.7	31.5	**53.1**
Avg.		29.8	31.2	**47.5**

Table 6.2: Adaptation results on Gene Recognition Task, FD05= (Finkel et al., 2005a) and HY09= Huang and Yates (2009)

From Table 6.2, we see that CCAM outperforms Finkel et al. (2005a) on average by 18%. We also compare with the HMM based adaptation method from Huang and Yates (2009). In this method, an HMM is learned from the combined unlabeled text of both source and target domains and then the HMM state of each word is added as a feature and a classifier from the source domain labeled data is retrained and then applied on the target domain text. We followed Huang and Yates (2009) and set the number of hidden states to be 80. It was not clear from Huang and Yates (2009) how to initialize the HMM learning process. So we repeated the above experiments 10 times, each time with a randomly initialized transition matrix and emission matrix before running the Baum-Welch process. The reported results are average of the 10 runs. We see that CCAM outperforms HMM-adapt by 16%.

[2]For a fair comparison, we tried adding these conjunctions of features to the source domain classifier also. But this led to a prohibitively long training time for the source domain model. Preliminary experiments on several data sets did not show significant gains after adding these features to source domain classifier due to presence of more discriminative features that are actually source domain-specific

6.4.2 Entity Recognition in News Domain

The following experiments show the validity of our approach on the recognition tasks of more conventional named entities like person, organization etc.

6.4.2.1 Experiments on ACE 2005 data set

ACE 2005 is an excellent data set for named entity adaptation experiments since it contains annotated text from multiple domains. We use four domains in our experiments: broadcast conversations (bc), conversational telephone speech (cts), news wire (nw) and weblog (wl). Our experimental settings are identical to those of Wu et al. (2009).

There are seven entity types in ACE 2005 data set: Person, Organization, Geo-political Entity, Location, Facility, Vehicle and Weapon.

We experiment on each pair of domains, taking one domain as the training domain and the other domain as the test domain. We have taken a state-of-the-art publicly available named entity recognition system[3] described in Ratinov and Roth (2009) and trained it over the training data of a domain and tested it over the test data of another domain. We tuned the parameters by cross validation over the training data and found that tuning did not impact the cross validation results significantly over the default parameter settings of Ratinov and Roth (2009). So in all our experiments, we decided to use the default parameter settings in Ratinov and Roth (2009).

The feature functions ϕ contained all discrete emission feature functions from ψ, the feature functions used in Ratinov and Roth (2009). We could not add conjunctions of features since it results in a very large number of features.

For CCAM, we find the constraints from two sources:

- Gigaword: As mentioned in Section 6.3.1, we run the source domain classifier over the Gigaword unlabeled corpus and extract both word and character level constraints. Gigaword in this case is the intermediate domain.

- Target Domain: We extract constraints from the target domain according to Section 6.3.2.

[3]available in `http://cogcomp.cs.illinois.edu/page/software_view/NETagger`

Train	Test	Wu09	RR09	CCAM
bc	cts	75.4	81.1	**86.8**
bc	nw	64.4	**78.6**	77.0
bc	wl	47.9	61.5	**62.6**
cts	bc	49.6	**67.4**	65.4
cts	nw	59	**70.5**	69.3
cts	wl	42	55.2	**55.5**
nw	bc	58.4	69.6	**71.7**
nw	cts	73.5	52.8	**77.5**
nw	wl	52.3	**69**	**69**
wl	bc	58.6	68.5	**70.9**
wl	cts	**81**	53.3	79.5
wl	nw	69.3	**80.5**	78.5
average		61.0	67.3	**72.0**

Table 6.3: Experimental results on ACE 2005 data set, Wu09= Wu et al. (2009), RR09= Ratinov and Roth (2009)

System	F1
RR09	64.1
CCAM	73.4
CCAM^{-w}	**75.4**
HY09	66.0
in-domain training(M05)	77.9

Table 6.4: Experimental results on CoNLL to Enron Adaptation, RR09= Ratinov and Roth (2009), HY09= Huang and Yates (2009), M05= Minkov et al. (2005),CCAM^{-w}=CCAM without using the word level constraints from Section 6.3.1

Wu et al. (2009) used a bootstrapping method for adaptation with several different configurations. We presented their best results in Table 6.3. From Table 6.3, we can see that Ratinov and Roth (2009) performs 6% better than Wu et al. (2009) without any adaptation and CCAM further improves over Ratinov and Roth (2009) by 5% on average. So, overall CCAM advances the previous state-of-the-art in NER adaptation on ACE data set by 11%.

6.4.2.2 Experiments on Email dataset

Our third set of experiments is on adapting a named entity recognizer from the news domain to the email domain. This is challenging since the structure of emails is very different from the structure

of text in news domain. For example, names often appear in different header fields of an email which are not grammatically correct sentences and the style of writing in the body of emails is very informal.

We use the corpus of emails extracted from the "meetings" or "calendar" folder of the employees of Enron and annotated with person names from Minkov et al. (2005). Minkov et al. (2005) split the corpus into train, tune and test sets. Since our purpose is to study adaptation, we train on the labeled corpus from CoNLL 2003 shared task (Tjong Kim Sang and De Meulder, 2003) and use this model on their test set. CoNLL corpus contains annotations for names of persons, organizations, locations and miscellaneous entities. We remove all names other than person names from the CoNLL corpus (we make them non-entity) and retrain the publicly available implementation of Ratinov and Roth (2009). For CCAM, we find the constraints from two sources:

- Gigaword: As mentioned in Section 6.3.1, we run the source domain classifier over the Gigaword unlabeled corpus and extract both word and character level constraints.
- Target Domain: We extract constraints from the target domain according to Section 6.3.2.

From Table 6.4, we can see that CCAM improves 9% over the baseline of no adaptation with Ratinov and Roth (2009) and 7% over a state-of-the-art HMM based adaptation method from Huang and Yates (2009). The results of Minkov et al. (2005) can be treated as an oracle score for any adaptation method without any labeled data in email domain since Minkov et al. (2005) used a large in-domain set to train their model. We see that without any labeled data in the email domain, CCAM was able to reach very close to the best published result on that data set that uses large in-domain training set. We can further see that the word level constraints from Section 6.3.1 hurt in this experiment (CCAM^{-w} without word level constraints is 2% better than CCAM). For example, a word level constraint learned from Gigaword is: word type "Lay" cannot be part of a person name but "Lay" appears as a person name in Enron corpus. This is expected: since Gigaword contains mostly news text and contains names of prominent people, whereas Enron corpus contains discussions between company employees. A good intermediate domain for Enron emails would be formally written company reports which we did not have access to.

6.5 Summary

This chapter presents one of the very few attempts in literature to learn constraints automatically from unlabeled data. We learn constraints from unlabeled data from the target domain and also from an intermediate domain between the source and target domains. We show how accurate constraints effective for domain adaptation can be learned from unlabeled data. While retaining the scalability of adaptation by avoiding the training of a new model, our approach also gives very strong empirical results, outperforming several state-of-the-art baselines across a wide range of data sets.

Chapter 7

Adaptation Using Wikipedia

The previous chapters focused on domain adaptation using external resources without any training. In this chapter, we extend this paradigm by showing how to leverage Wikipedia, a large encyclopedia for adapting a named entity recognition system across a range of domains. We show how we can use the same named entity recognition system across all domains and just modify the inference phase to perform a joint inference with a system for grounding mentions to Wikipedia. Our joint inference not only improves adaptability of the named entity recognizer but also improves the system for linking text to Wikipedia.

7.1 Introduction

Entity Linking is the task of grounding mentions in text to concepts in a Knowledge base, e.g. Wikipedia. Most previous systems proposed for this task have assumed that the gold mention boundaries are given and focused on linking the mentions to correct Wikipedia concepts. Identifying the boundary and providing the correct type of entity (e.g. People, Location, Organziation etc.) is known as the task of named entity recognition. Most previous works in the literature have focused on these tasks in isolation, e.g. for entity linking (Hoffart et al., 2011; Han and Zhao, 2009; Cucerzan, 2007; Bunescu and Pasca, 2006; Ratinov et al., 2011), for named entity recognition (Ratinov and Roth, 2009; Finkel et al., 2005b). But performing these two tasks in joint can yield significant benefit. For example, consider the following sentence

> Cricket – England beat Pakistan by 107 runs in second one dayer.

Wikifier or the entity linking system links the mention *England* to `http://en.wikipedia.org/wiki/England`. The correct Wikification should be `http://en.wikipedia.org/wiki/England_cricket_team`. If we use NER on this sentence, NER predicts *England* as an entity of type

ORG. Since the predicted concept `http://en.wikipedia.org/wiki/England` is of type LOC and the correct concept `http://en.wikipedia.org/wiki/England_cricket_team` is of type ORG, by using consistency constraints, we can make Wikifier select the correct candidate. So jointly assigning a named entity label and Wikipedia URL that are consistent with each other can improve over doing them separately.

There is a significant amount of difference between the operations of named entity recognition systems and Wikification systems. Typically named entity recognition systems are based on sequential labelers, using features in local and global contexts. On the other hand, Wikification systems first generates mentions from text using a set of heuristics. Then, they take the set of Wikipedia pages that the mention linked to in Wikipedia as the potential set of disambiguation pages for that mention. Then they use a ranker to select the highest scoring page as the final output for that mention.

Named entity recognition systems, from the inception of the task have used list of names of persons, locations, organizations, films etc. Typically, these lists have been used in generating features for the NER system (e.g. for a mention, whether the mention appears in a particular list). Ratinov and Roth (2009) created this list from Wikipedia. A problem of this approach is that often a mention is ambiguous in the sense that it can refer to multiple Wikipedia concepts. For example, the mention *Chicago* can refer to the city of Chicago or the sports team *Chicago Bears*. For generating features from gazetteers, NER systems do not attempt to solve this ambiguity problem. The task of Wikification is exactly this, given an ambiguous mention, identify what Wikipedia concept it refers to. On the other hand, this task relies on NER to detect a set of mention boundaries that it can attempt to link to Wikipedia. The previous systems for Wikification assumed that the gold mention boundaries were given and linked those mentions to Wikipedia. In this work, we have proposed a principled joint formulation for Wikification and named entity recognition.

Only one work(Sil and Yates, 2013), to the best of our knowledge, has attempted joint NER and Wikification. But their joint system does not perform true named entity recognition. The job of a true named entity recognition system is to perform both identification of mention boundary and providing the type of entity for that mention. The system of Sil and Yates (2013) does not predict the type of entity and instead just outputs the boundary of the mention, along with the

Sentence	Phyllis Wise is the Chancellor of University of Illinois
Gold NER	Phyllis Wise$_{PER}$ is the Chancellor of University of Illinois$_{ORG}$
Gold Wiki	Phyllis Wise is the Chancellor of University of Illinois(`http://en.wikipedia.org/wiki/University_of_Illinois_at_Urbana_Champaign`)
(Sil and Yates, 2013)	Phyllis Wise is the Chancellor of University of Illinois(`http://en.wikipedia.org/wiki/University_of_Illinois_at_Urbana_Champaign`)
Our system	Phyllis Wise$_{PER}$ is the Chancellor of University of Illinois$_{ORG}$(`http://en.wikipedia.org/wiki/University_of_Illinois_at_Urbana_Champaign`)

Table 7.1: Definition of Tasks for NER, Wikifier, (Sil and Yates, 2013) and Our system

Wikipedia concept for that mention if any.

Table 7.1 shows an example explaining the difference of setting of ours and that of Sil and Yates (2013). The joint formulation of Sil and Yates (2013) only detects the mention boundaries but does not provide any entity label. The important facts that *Phyllis Wise* is a person and *University of Illinois* is an organization are missing from the output of Sil and Yates (2013). In their joint formulation, they extract a set of mention candidates from the sentence, extract a list of candidate Wikipedia pages for each mention candidate and uses a ranker that outputs the best set of mention boundaries and the corresponding Wikipedia concepts. They train the ranker from Wikpedia pages. These pages provide gold standard mention boundaries and the gold standard Wikifications. But gold standard named entity annotations are not available from these pages. Therefore, it is non-trivial to extend their formulation to output named entity type for every mention, including mentions that will not be linked to Wikipedia.

Contributions

- The tasks of named entity recognition and linking are intrinsically related. A named entity recognition system can predict the entity type of a mention which can help the linking system to find a Wikipedia concept with a matching entity type. Again, if a linking system can accurately disambiguate a mention to a Wikipedia concept, the entity type of the concept can aid the NER in predicting the correct type for that mention. We show that this intuitive principle does hold in empirical evaluations.

- We formulate a joint inference across NER and Wikifier. Our formulation does not need

retraining of any kind. It uses the already trained models of NER and Wikifier and simply modifies the inference phase such that for each mention in text, the predicted NER type and Wikipedia concepts are consistent with each other.

- We show that joint inference with Wikifier can help the NER performance across a range of domains. Both in in-domain and out-of-domain test sets, joint inference with Wikifier improves NER performance significantly.

- We show that in-domain setting, joint inference improves Wikifier significantly, while showing no performance drop for a state-of-the-art named entity recognition system.

7.2 Motivating Examples

Consider the following sentence

> Cricket – England beat Pakistan by 107 runs in second one dayer.

The gold level annotation of named entity label and Wikipedia page for the mention *England* in the sentence above are *ORG* and `http://en.wikipedia.org/wiki/England_cricket_team`. Unfortunately Wikipedia does not provide the type of each article in terms of *PER, LOC, ORG, MISC*. Freebase is a large collaborative knowledge base consisting of meta data composed by volunteers. It contains a rich hierarchy of types for all of the articles of Wikipedia. This set of types in Freebase contains the types that we are interested in (PER, LOC, ORG, MISC). England_cricket_team has the type *ORG* in freebase. Again consider the following sentence

> He was captured in Worthing , a town on England 's southern coast.

The gold level annotation of named entity label and Wikipedia page for the mention *England* in the sentence above are *LOC* and `http://en.wikipedia.org/wiki/England`. The Wikipedia concept for `http://en.wikipedia.org/wiki/England` has the type *LOC* in freebase.

It is clear that we need to assign named entity labels and Wikipedia URLs to mentions such that the joint assignment is consistent. A summary of the first example along with the confusion set is given in Table 7.2. In that table, we also show the freebase types for each Wikipedia concept.

Sentence	Mention	Wikifier confusion set	Freebase Types	NER confusion set
Cricket – England beat Pakistan by 107 runs in second one dayer.	England	{`http://en.wikipedia.org/wiki/England`, `http://en.wikipedia.org/wiki/England_cricket_team`}	LOC ORG	{LOC, ORG}

Table 7.2: Confusion set of Wikifier and NER for the mention *England*. The gold label for Wikification & NER for England are respectively `http://en.wikipedia.org/wiki/England_cricket_team` and ORG

7.3 Formulation of Joint Inference of NER and Wikifier

Description of NER Model The NER model used in Ratinov and Roth (2009) is an Averaged Perceptron. It uses BILOU encoding to encode the NER mention boundaries and labels. U is used to encode mentions of length 1. B and L are used to mark the beginning and end of mentions respectively. I is used to mark the intermediate tokens of a mention. O is used for tokens outside any NER mention. Therefore, the total number of labels is 17 ((Y)={B,I,L,U} $\times$ {PER, LOC, ORG, MISC} $\cup$ {O}). For each token x and a label $y \in \mathcal{Y}$, NER model generates a score $w_y^T \cdot \phi(x)$.

For each mention consisting of potentially multiple words, we need a single score s for each of the label {PER, LOC, ORG, MISC, O} for performing joint inference. This is done as follows: For mentions consisting of one token x

$$s^{ner}(l) = \begin{cases} w_{U-l}^T \cdot \phi(x) & \forall l \in \{PER, LOC, ORG, MISC\} \\ w_O^T \cdot \phi(x) & l = O \end{cases} \tag{7.1}$$

For mentions consisting of multiple tokens $x_1, x_2, \ldots, x_n$

$$s^{ner}(l) = \begin{cases} \frac{w_{B-l}^T \cdot \phi(x,1) + \sum_{i=2}^{n-1} w_{I-l}^T \cdot \phi(x,i) + w_{L-l}^T \cdot \phi(x,n)}{n} & \forall l \in \{PER, LOC, ORG, MISC\} \\ \frac{\sum_{i=1}^{n} w_O^T \phi(x,i)}{n} & l = O \end{cases} \tag{7.2}$$

$\phi(x, i)$ is the feature vector for NER for token i'th in the sentence. According to BILOU scheme,

a mention of one token will be classified as label $l \in \{PER, LOC, ORG, MISC\}$ if the classifier predicts $U-l$ for that token, justifying Equation 7.1. A mention of multiple tokens will be classified as label $l \in \{PER, LOC, ORG, MISC\}$ if the classifier predicts $B-l$ for first token $L-l$ for the last token and $I-l$ for all intermediate tokens, justifying Equation 7.2. If a token is not part of any mention, the classifier simply predicts label O for it. Note that here we talk about predicting the type of each mention, not the boundary of the mention. We can get the boundaries of the mentions from the union of the NER predicted mention boundaries and words in test set that did not appear in the training set. Since many of the mentions may not belong to an entity type, we also consider the label O.

Description of Wikifier Model The Wikifier used in our experiments is the state-of-the-art wikifier described in Ratinov et al. (2011). This is a pipeline system of multiple modules. The first module selects the most frequent candidate Wikipedia pages (upto 20) for each mention. Next it uses a ranker based on SVM-rank[1] to rank the candidates. For each candidate page t, it generates a score

$$s^{wiki}(t) = w^T \cdot \psi(t) \tag{7.3}$$

$\psi(t)$ is the feature vector for Wikifier. Wikifier selects the candidate with the highest score.

Let $\mathcal{L}$ be the set of NER Labels. For a mention m,, let $\mathcal{T}$ be the set of Wikipedia concepts for m short-listed by the Wikifier after pruning($\mathcal{T}$ is at most of size 20). Let x_l^{ner} be a binary variable with $x_l^{ner} = 1$ indicates that NER label of mention m is l ($l \in \mathcal{L}$). Similarly, let x_t^{wiki} be a binary variable with $x_t^{wiki} = 1$ indicates that the Wikipedia concept for mention m is t for $t \in \mathcal{T}$. Let the set $\mathcal{T}_l$ denote the set of titles that have NER label l from freebase.

The joint ILP program for each mention can be written as

$$\max\ \alpha * \sum_{l \in \mathcal{L}} c^{ner}(l) x_l^{ner} + (1-\alpha) * \sum_{t \in \mathcal{T}} c^{wiki}(t) x_t^{wiki} \tag{7.4}$$

subject to the constraints

$$\sum_{l \in \mathcal{L}} x_l^{ner} = 1 \tag{7.5}$$

[1] http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html

$$\sum_{t \in \mathcal{T}} x_t^{wiki} = 1 \tag{7.6}$$

$$x_l^{ner} \geq x_t^{wiki}, \forall t \in T_l \tag{7.7}$$

Equation 7.4 defines the objective value for each assignment of NER label and Wikipedia page to a mention. $c^{ner}(l)$ is the normalized score assigned by the NER model to label l for mention m. It is obtained after normalization of the scores from either Equation 7.1 or 7.2 over all labels. $c^{wiki}(t)$ is the normalized score assigned by the Wikifier model to Wikipedia page t for mention m. It is obtained after normalization of the scores from Equation 7.3 over all candidate pages. α is a parameter in the range [0,1] that weights the contribution of the score from the NER and Wikifier.

Equation 7.5 ensures that only one NER label is selected. Equation 7.6 ensures that only one Wikipedia page is selected. Equation 7.7 ensures a consistent assignment of label and wikipedia page across NER and Wikifier. Specifically, if the Wikipedia page t is selected with freebase type l, NER label l must also be selected.

Integer linear programming is needed due to the constraints from Equation 7.7 that ensure consistent assignment of NER and Wikifier predictions. However, it is possible to avoid integer linear programming and perform a direct comparison of the scores for each compatible (NER prediction, Wikifier prediction) pair.

7.4 Experimental Results

For wikification, we use the publicly available Wikification system[2] described in Ratinov et al. (2011). This Wikification system comes pre-trained from a subset of Wikipedia pages. In all our experiments, we used this Wikification system without performing any additional training.

We use the publicly available NER system[3] described in Ratinov and Roth (2009). This is the state-of-the-art publicly available system for named entity recognition.

[2] `http://cogcomp.cs.illinois.edu/page/software_view/Wikifier`
[3] `http://cogcomp.cs.illinois.edu/page/software_view/NETagger`

7.4.1 In-domain Experiments

For our first group of experiments, we train the NER system on the CoNLL 2003 shared task data set Tjong Kim Sang and De Meulder (2003). CoNLL corpus contains annotations for names of persons, organizations, locations and miscellaneous entities. This data set was annotated with Wikipedia URLs for named entities from Hoffart et al. (2011).

The constraints that we used in these experiments are based on consistency of predicted labels across NER and Wikifier. We noted that in the data set, some mentions were linked to Wikipedia concepts belonging to any of the types of {*PER,LOC,ORG*} but the corresponding gold named entity type was *MISC*. For example, *Israeli* refers to the nationality or citizens of Israel. Its gold named entity label is *MISC* and gold Wikification label is `http://en.wikipedia.org/wiki/Israel` of type *LOC*, since there is no page in Wikipedia for *Israeli*. So our constraints allow a Wikipedia concept of any of the types {*PER,LOC,ORG*} from Wikifier to be consistent with the named entity type *MISC* from the named entity recognizer. The constraints used for this data set are listed below

- For a mention, if the selected Wikipedia concept has type *PER* from freebase, the type from named entity recognizer can be only *PER* and *MISC*.
- For a mention, if the selected Wikipedia concept has type *LOC* from freebase, the type from named entity recognizer can be only *LOC* and *MISC*.
- For a mention, if the selected Wikipedia concept has type *ORG* from freebase, the type from named entity recognizer can be only *ORG* and *MISC*.

Figure 7.1 shows how the F1 scores of NER and Wikifier vary on the CoNLL development set as the parameter α is varied. The parameter α is used only in the joint formulation of NER and Wikifier to weight the contribution of the scores coming from NER and Wikifier for making the joint assignment of NER and Wikifier labels. This parameter is not used in the Disjoint NER or Disjoint Wikifier. Therefore, the performance of Disjoint NER or Disjoint Wikifier remain constant throughout the variation of this parameter. For the joint case, NER score is multiplied by α and Wikiifer score is multiplied by $1 - \alpha$. Therefore, if α is small, the score from the Wikifier plays a bigger role than the score from NER and Wikifier overturns many decisions of NER. Joint NER

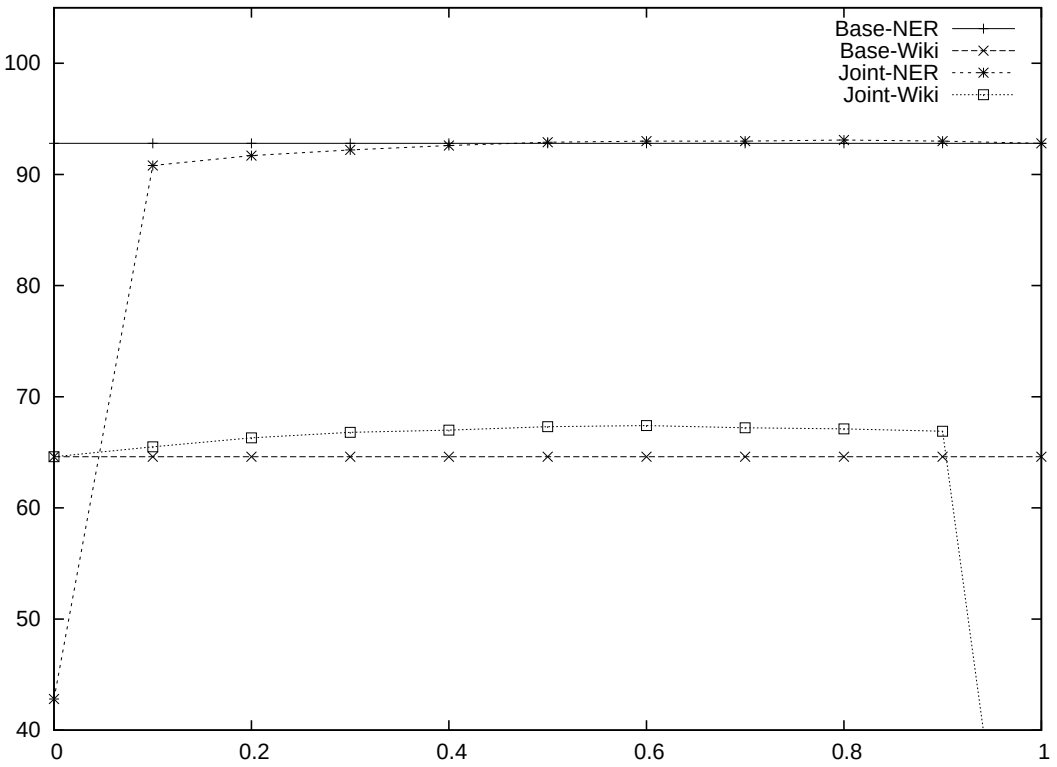


Figure 7.1: Variation of F1 scores of Joint NER, Joint Wikifier, Disjoint NER and Disjoint Wikifier (y axis) on the CoNLL Dev Set as α is varied (x axis)

performs worse than Disjoint NER. This is partly due to the fact that Disjoint NER is very accurate on this data set. On the other hand, as α becomes larger, the score from the NER starts playing a bigger role and overturns Wikifier decisions. Joint Wikifier improves steadily over Disjoint Wikifier as α is increased. When $\alpha = 1$, Wikifier score does not play any role and random selections are made for Wikifier decision variables. This is the reason for the big drop of Joint Wikifier at $\alpha = 1$. Similarly, When $\alpha = 0$, NER score does not play any role and random selections are made for NER decision variables. This is the reason for the big drop of Joint NER at $\alpha = 0$. For values of α closer to 1, Joint Wikifier gives significant improvement (3% in absolute F1) over Disjoint Wikifier and the performance of Joint NER remains the same as the Disjoint NER.

Is the α value tuned over the dev set representative for unseen test set? The value of α that obtained the highest F1 score for both NER and Wikifier over the dev set was 0.9. We evaluate the Joint system over the CoNLL test data set using $\alpha = 0.9$. Results over the test set are shown in Table 7.3.

System	Base	Joint
NER	87.4	87.4
Wikifier	64.3	66.5

Table 7.3: Disjoint NER, Joint NER, Disjoint Wikifier and Joint Wikifier F1 on CoNLL test set using α=0.9

The performance of the Joint system on the CoNLL test set shows characteristics similar to those on CoNLL dev set. From Table 7.3, we see that performance of Joint NER remains the same as that of Disjoint NER. This is nice, considering that the system used for Disjoint NER is the publicly available system of Ratinov and Roth (2009) which is the state-of-the-art NER system and NER systems on this data set have almost reached an upper ceiling. From Table 7.3, we see that performance of Joint Wikifier improves over that of Disjoint Wikifier by 2%. Our error analysis shows that most of this improvement is due to the fact that for a mention, NER forces a candidate from Wikifier to be selected such that the candidate's category from freebase (in terms of LOC, ORG, PER) matches the category predicted by NER for that mention. Note that the tuned value of α over the dev set was 0.9. So, for a mention, the weight of the NER is 0.9 and the weight of the Wikifier is 0.1. Due to the higher weight on the NER part, the joint inference process almost always does not change the NER prediction, instead changes the prediction of the Wikifier to be

compatible with the prediction made by the NER. Since the dev & test set is from the same domain as the training set for NER, NER's performance is very good (close to 93 F1 on dev, 90 F1 on test). So when the joint inference process changes the Wikifier prediction to be compatible with the NER prediction, Wikifier performance improves.

The proportion of named entities with a Wikipedia article in CoNLL dev and test set is respectively 81 & 80%. This proves that there is a high degree of overlap across the two tasks and joint should improve both tasks.

Running time of NER over 5 documents from CoNLL dev set is 41s while running time of Wikifier over 5 documents is 239s. However, the running time of Wikifier can be improved by efficient implementation.

7.4.2 Domain Adaptation Experiments

Given that NER did not improve significantly on the CoNLL data set, one natural question is that will the Joint NER be better in adaptation scenarios? The learning protocol is that we train NER independently of the the Wikifier from source domain labeled data. In target domain, we perform joint inference of NER and Wikifier.

How joint inference for Wikifier and NER can help adaptation of NER will be explained below with an example. Let the source domain be political news and the target domain be financial news. Table 7.4 shows text snippet from the two domains. Entities with their boundaries are marked with underlines and the corresponding entity type is listed as subscript. Note that organization entities vary significantly from one domain to the other. In source domain, organizations are mostly political or military organizations and in target domain, organizations are business companies. Note that surface forms of the organizations as well as contextual words of the organizations vary significantly from source domain to target domain. Therefore, an NER system trained from labeled data of the source domain will perform poorly on text from the target domain.

Table 7.5 shows the entities predicted by the NER (trained on source domain) on target domain text. Note that NER correctly predicted *Pierre Omidyar* as a person but failed to predict *eBay* as an organization. This is because the surface form of *eBay* was not capitalized, whereas in the source domain most entities were capitalized. Also, the words in context of *eBay* are very different

Source Domain = Political News	Target Domain = Financial News
About 1,000 soldiers from the US Army$_{ORG}$'s 173rd Airborne Brigade$_{ORG}$ parachuted into an airfield in a Kurdish - controlled area of northern Iraq$_{LOC}$, establishing a base through which to bring in more troops and tanks , Pentagon$_{ORG}$ officials said .	eBay$_{ORG}$ was founded by Pierre Omidyar$_{PER}$ in 1995, and became a notable success story of the dot-com bubble; it is a multi-billion dollar business with operations localized in over thirty countries.

Table 7.4: Text snippet from source and target domains

Gold NER on Target Domain	Source Domain Trained NER predicted entities on Target Domain
Pierre Omidyar (PER) eBay (ORG)	Peirre Omidyar (PER)

Table 7.5: Named entity annotation (gold & predicted by the NER trained on source labeled data) for target domain text

from the contexts of organizations in source domain.

Table 7.6 shows the predictions made by the Wikifier on target domain text. We can see that Wikifier correctly links both *Pierre Omidyar* and *eBay* to their respective Wikipedia pages. Freebase provides types for each wikipedia page. We can see that it has annotations of *PER* and *ORG* for the Wikipedia pages of *Pierre Omidyar* and *eBay* respectively.

It is clear that the mistake of NER on the mention *eBay* can be corrected by looking at the Wikipedia URL predicted for this mention and the Freebase type for that Wikipedia URL. Since Wikipedia covers a broad range of domains (e.g. political, financial etc.), we can expect the Wikifier will improve NER performance on a broad range of domains.

For studying impact on NER adaptation, we run experiments on the ACE 2005 data set. There are seven entity types in ACE 2005 data set: Person, Organization, Geo-political Entity, Location,

Wikifier Predictions on Target Domain	Freebase (URL, type) pairs
Pierre Omidyar (`http://en.wikipedia.org/wiki/Pierre_Omidyar`) eBay (`http://en.wikipedia.org/wiki/EBay`)	(`http://en.wikipedia.org/wiki/Pierre_Omidyar`,PERSON) (`http://en.wikipedia.org/wiki/EBay`,ORG)

Table 7.6: Wikipedia URLs predicted for mentions in the target domain text by the Wikifier and the types listed in freebase for those URLs

Source Domain	Target Domain	Disjoint NER	Joint NER
conversational dialogue(bc)	devset (bc)°	76.6	**77.0**
	testset(bc)°	80.1	80.2
	testset(newswire)⋆	68.2	**69.9**
	average	75.0	**75.7**

Table 7.7: In-domain° & Out-domain⋆ experimental results for NER

Facility, Vehicle and Weapon. This data set contains four domains: broadcast conversations (bc), conversational telephone speech (cts), news wire (nw) and weblog (wl). Our experimental settings are identical to those of Wu et al. (2009).

For the experimental results reported in Table 7.7, NER was trained on a subset of the annotated data from the broadcast conversation domain. We split the total annotated data from this domain into train, dev & test portion according to a 70%-15%-15% split. We ran both Disjoint NER and Joint NER on the dev set and test set. In the dev set, Joint NER outperformed Disjoint NER significantly. Since training and testing are performed in data from the same domain, these experiments show that Joint NER can improve over Disjoint NER even in in-domain scenarios.

We also ran NER trained on the training portion of bc data set on the test portion of the newswire domain available in ACE2005 data set. Broadcast conversion domain consists of dialogues between news hosts and invited guests or random callers to the shows. This domain mostly covers stories related to US interests e.g., Iraq War in 2005. Newswire domain consists of news stories published by news agencies like Xinhua, AFP and contains stories from all over the world e.g. India, Indonesia etc. Compared to the well studied CoNLL data set, training data here is substantially smaller (2200 sentences in BC domain), number of labels to predict is larger (7 instead of 4 in CoNLL). Also the syntax seen in dialogues (BC domain) is different than the syntax seen in published news stories (Newswire domain). Therefore, this experimental setting is a difficult setting for NER.

From Table 7.7, we see that Joint NER improves significantly over Disjoint NER for out-of-domain test set (testset(news wire)) but improvement in in-domain devset (devset(bc)) and in-domain testset (testset(bc)) is not significant. The value of α used in this experiment was the value of α that obtained the highest F1 on the development set of source domain (devset(bc)). From Table 7.7, we see that when the model is trained and tested on the same domain (bc),

the performance does not improve much (the results of devset (bc) & testset(bc) in Table 7.7). But when the model is tested on a testset from a different domain (result on testset(news wire)), the F1 improves 1.7%. An error analysis shows that Disjoint NER trained on dialogue fails to identify many organization names in newswire. However, Wikifier correctly links them to their corresponding Wikipedia pages and the joint inference process succeeds in correcting the NER mistakes.

We do not have the gold standard Wikifications for ACE data set. So we have reported results for only the task of named entity adaptation on this data set.

7.5 Summary

In this chapter, we presented a method for joint inference of NER and Wikifier that improves the performance of Wikifier in in-domain experiments. It also improves the performance of NER in out-of-domain settings due to the robust nature of Wikifier in terms of domain variations. This work, other than proposing a promising method for adaptation of NER across domains, shows that any task that is relatively robust to domain variations can improve the performance of other tasks, that are susceptible to domain variations if the prediction of outputs for both tasks can be performed jointly with constraints on consistency on their outputs.

Chapter 8

Adaptation using Small Training on Target

The previous chapters focused on adaptation to a target domain without retraining the source model. The method proposed in Chapter 6 captures strongly discriminative target domain specific patterns for each label. These patterns are then converted to hard constraints. During the inference phase on the target domain test set, we use the constrained conditional model formulations with the source model and these hard constraints for prediction. Since all the learned constraints were treated as hard constraints during inference time, we could only use very accurate target domain patterns with accuracy close to 100%. However, there may be target domain specific patterns that are not very strongly discriminative but that are useful for the classification task at hand. We cannot learn and convert these patterns into hard constraints. However, if we can train a model on some target domain labeled data (automatically generated), these patterns will get appropriate weights. The trained model might perform better than constrained condition models with only hard constraints. In this chapter, we show how we can train a simple model for the target domain using only target domain labeled data (automatically generated) and perform joint inference with the model from the source domain. This gives significant performance improvement in accuracy while training time is kept to a minimum.

8.1 Introduction

The performance of a machine learning model trained on some labeled data of a domain degrades severely when tested on data from a different domain (Blitzer et al., 2006). Since expensive human annotation of data for every new domain in infeasible, we need methods that can adapt models trained on the labeled data from a domain (source) to a new domain (target).

There are two distinct paradigms in the current domain adaptation literature: the first one

assumes that a large amount of labeled data exists in the source domain and a small amount of labeled data exists in the target domain. This paradigm will be referred to as the *Supervised Adaptation* paradigm in this paper. The second paradigm assumes that labeled data is available from the source domain only. Only *unlabeled* data is available in the target domain. This paradigm will be referred to as the *Unsupervised Adaptation* paradigm in this paper.

In this work, we claim that this difference is rather artificial. Even if we face an unsupervised adaptation scenario, our choice of algorithm need not be restricted from those proposed only for this scenario e.g. Huang and Yates (2009) and Blitzer et al. (2006). Instead, if we can generate some *accurate* labeled data in the target domain automatically by using the source model (model trained on the labeled data from the source domain), we can apply any method proposed for the paradigm of supervised adaptation e.g. Daumé III (2007) and Chelba and Acero (2004). In our experiments, we show that these methods give better results than the purely unsupervised methods.

Of course, this depends on how we generate the labeled data in the target domain by the source domain model. Most of the labels predicted by the source model on target domain unlabeled data will not be correct. So if a supervised adaptation method treats all labels predicted for the target domain instances as gold standard labels, it will not perform well. We show that instead of trusting labels for all target domain instances, we should trust only labels for those target domain instances that also appeared in the source domain with identical lexical forms. A supervised adaptation method that treats the predicted labels for this set of target domain instances as gold standard labels, performs better than purely unsupervised methods.

A problem with all supervised adaptation methods, e.g. (Daumé III, 2007; Chelba and Acero, 2004; Finkel and Manning, 2009; Dai et al., 2007) is that they throw away the model trained on only source labeled data. These methods train a new model on the combined source labeled data and the target labeled data for every target domain. This leads to a large training time. We propose a method named FAJIST (Fast Adaptation Using Joint Inference of Source and Target Models) where we do not throw away the source model. Instead, we train a *target only* model on the *selected* target domain instances for every target domain using identical features and learning settings as was used for training the *source only* model. Next, we perform joint inference of the *source only* and *target only* models. Since FAJIST trains the *target only* model for every target

Target Text	NER predictions	Gold NER
Mtg. w/ John, Larry & Jacob	Mtg. w/ $\underline{\text{John}}_{PER}$, $\underline{\text{Larry}}_{PER}$ & $\underline{\text{Jacob}}_{PER}$	Mtg. w/ $\underline{\text{John}}_{PER}$, $\underline{\text{Larry}}_{PER}$ & $\underline{\text{Jacob}}_{PER}$
Mtg w/ David Anderson tomorrow	Mtg w/ $\underline{\text{David Anderson}}_{PER}$ tomorrow	Mtg w/ $\underline{\text{David Anderson}}_{PER}$ tomorrow
Mtg w/ **Fastow**, Kerry	Mtg w/ **Fastow** , $\underline{\text{Kerry}}_{PER}$	Mtg w/ $\underline{\textbf{Fastow}}_{PER}$, $\underline{\text{Kerry}}_{PER}$

Table 8.1: Text snippets from the target (email) domain, predictions of an NER (trained on source (news)) on this text and corresponding gold NER annotations. Each underlined text segment represents a mention either predicted by the NER model or from gold annotations. The corresponding predicted entity type and the gold entity type are shown as subscript. The portion of text where the NER predictions did not match the gold NER annotations has been shown in bold.

domain, the training time is significantly smaller compared to other approaches, e.g. (Daumé III, 2007) that requires expensive training over the combined source and target labeled data.

Contributions

- We show that supervised adaptation methods can be applied to unsupervised adaptation scenarios with superior performance, if some accurate labeled data can be generated automatically in the target domain. For the task of named entity recognition, we propose a selection policy for selecting *accurate* predictions on the target domain instances by the source domain model.
- We propose an efficient adaptation method FAJIST that trains a model on only target domain instances and then, performs joint inference of source and target models on target domain test instances. FAJIST exhibits performance comparable to Daumé III (2007) but is 8 times more efficient.

8.2 Motivating Example

Table 8.1 shows some text snippets from the target domain (email) and predictions made by the NER trained on the source domain (news) and the corresponding gold standard annotations. The target domain consists of mails from the calendar or meetings folder of the employees of a company.

So many emails contain lists of the names of attendees of a meeting. These lists are often followed by contexts like "meeting w/", "mtg w/" etc. So features like "the prev word = w/" is a strong indicator of the current word being a PERSON. But this feature never appears in the source domain (news) which consists of formally written English news articles.

From Table 8.1, we can see that the source model correctly predicts phrases like *John*, *Larry*, *Jacob*, *David Anderson* and *Kerry* as entities of type PERSON. The reason is that these phrases appeared as entities of type PERSON in the source domain or in a gazetteer (list of person names). But the source model failed to predict the phrase *Fastow* as an entity of type PERSON in the target domain text. Because this phrase did not appear in the source domain labeled data or in the list of person names or gazetteers used for feature generation in the system. When the model encounters such an unseen phrase, the context should guide the model into making the correct decision. Unfortunately the context in which this phrase appears (left context = "mtg w/" and right context = ", Kerry") was also unseen in the source domain. So the source model failed to predict *Fastow* as an entity of type PERSON.

Our goal is to automatically learn from the target domain unlabeled data that words in context "w/", "mtg" are strong indicators of entities of type PERSON in the target domain. This will help in correcting the mistake on the phrase *Fastow*.

8.3 Algorithm

Algorithm 6 Key Steps of FAJIST

Input: Source model w_s, Target domain unlabeled text $\{x_t\}$
Output: Target model w_t

1: Predict labels $\{y_t\}$ for $\{x_t\}$ using w_s
2: Select a subset of instances $\{x'_t, y'_t\}$ from $\{x_t, y_t\}$
3: Train a target model w_t on $\{x'_t, y'_t\}$
4: **for** each target domain instance x_t **do**
5: predict label $y_t^\star$ using joint inference of models w_s and w_t
6: **end for**

Algorithm 6 gives an overview of the algorithm FAJIST (Fast Adaptation by Joint Inference of Source and Target). The key steps are prediction (line 1), selection (line 2), training (line 3) and joint inference (lines 4-5). They are discussed below:

Prediction In this step, the source model is used to predict labels for all target domain instances. In our experiments, these instances contained the target domain test set instances also, but without the labels. This is not a requirement though.

Selection In this step, a subset of the target domain instances and their labels predicted by the source model are selected as training data for a new model. The selection policy determines the quality of this training data and so is very important. In our experiments, we have found that the source model is very accurate on the phrases that appear in either the source labeled data or in the list of gazetteers. For example, referring to Table 8.1, the source model correctly predicted phrases like "John", "Kerry" as entities of type PERSON and also words like "tomorrow", "," as words of type non-entity, because, these words or phrases either appeared in the source labeled data or in a list of gazetteers. Therefore, the selection policy we use in our experiments is:

> Select a phrase and the label predicted for it by the source model if that phrase appeared with the same label in the source domain labeled data or in a list of gazetteers.

In terms of the examples in Table 8.1, some selected target domain instances will be *(David, B-PERSON)*, *(Anderson, I-PERSON)*, *(John, B-PERSON)*, *(tomorrow, O)* etc. Here *O* refers to the non-entity label. However, the instance *(Fastow, O)* will not be selected since *Fastow* did not appear in the source labeled data or in gazetteers.

Our selection policy for *accurate* target domain instances has an implicit assumption. It assumes that the source and target domains should have some overlap in terms of words / phrases and also, these words / phrases preserve the labeling from the source to the target domain. Empirical evaluation of this selection policy on the email domain shows that the accuracy of labels for the selected target domain instances is 100%.

Training In this phase, a model is trained from only the selected set of instances and their labels from the selection phase. The features and learning algorithm are identical to those used for training the source model.

Joint Inference In this phase, for each target domain instance, we perform a joint inference of the source and target models. For the task of named entity recognition, the inference problem is

formulated as

$$\mathbf{y}^{\star} = \arg\max_{\mathbf{y}} \sum_{i} \alpha * w_s^T \phi(\mathbf{x}, i, y_i) + (1 - \alpha) w_t^T \phi(\mathbf{x}, i, y_i) \tag{8.1}$$

In Equation 8.1, α is a parameter for adjusting the contribution from the source and target models, $\mathbf{x}$ is a document, $\mathbf{x}_i$ is the i'th word in the document, $\mathbf{y}$ is the sequence of labels for the words in the document, $\phi(\mathbf{x}, i, y_i)$ is the feature vector for i'th word in the document and w_s and w_t are respectively the source and target models.

8.4 Experimental Results

We want to evaluate two research questions

- How does the performance of FAJIST compare to that of other state-of-the-art unsupervised methods (Huang and Yates, 2009)?
- Can supervised adaptation methods using automatically generated target labeled instances outperform the state-of-the-art unsupervised adaptation methods (Huang and Yates, 2009)?

Method	F1		Training Time	
	meetings	random	Absolute	Relative
Source Model	61.3	73	350s	8x
Target Model	62.6	73.6	45s	1x
FAJIST	**68**	**76.3**	45s	1x
HY09	64	75	700s	16x

Table 8.2: F1 score on two email data sets (emails related to meetings and randomly selected emails) and training time for one email data set (meetings). The source domain is news wire. In terms of accuracy, FAJIST outperforms both source only and target only models. It also outperforms a state-of-the-art adaptation method Huang and Yates (2009) (referred to as HY09 in table). In terms of training time, FAJIST has a speedup of 16x compared to Huang and Yates (2009).

In all of our experiments, we use the publicly available implementation of Ratinov and Roth (2009). This software achieves the state-of-the-art result when trained and tested on the CoNLL 2003 shared task data set (Tjong Kim Sang and De Meulder, 2003). CoNLL corpus contains annotations for names of persons, organizations, locations and miscellaneous entities. Since our target domain (email) has annotations for person names only, we remove all names other than

person names from the CoNLL corpus (we make them non-entity) and retrain the software of Ratinov and Roth (2009).

As our target domain test sets, we use two two email data sets:"meetings" and "random". The "meetings" corpus was created by selecting some emails from the "meetings" or "calendar" folder of the employees of Enron. The "random" corpus was created by randomly selecting emails from all folders of the employees of Enron. We use the test sets from the train-dev-test split by Minkov et al. (2005) as our target domain test sets.

In Table 8.2, the source model refers to the model trained using the publicly available software of Ratinov and Roth (2009) from CoNLL data set. We see that this model achieves F1 below 80% for both email data sets. This is far below the F1 on CoNLL test set (90%). The reason for this drop is the acute difference of the source and target domains. The structure of emails is very different from the structure of text in CoNLL training set. In emails, names often appear in different header fields along with other email-specific non-name words (e.g. words like "FW"and email addresses). Also, the body of emails is written in a very informal way compared to structured news text.

In Table 8.2, Target model refers to the model trained using the same software from the *selected* target domain instances. FAJIST performs joint inference of the source and target models. We see that for both data sets, FAJIST outperforms both source and target only models significantly.

Table 8.2 also shows the performance of the HMM based adaptation method. In this method, using Baum-Welch algorithm, a Hidden Markov Model is learned over the combined source and target domain unlabeled data. After doing optimal Viterbi decoding, each word in each sentence is assigned a HMM state. This new state is added as a feature and the source model is retrained from the source domain labeled data. This method was shown to yield better results over other unsupervised adaptation methods like Blitzer et al. (2006). From Table 8.2, we see that FAJIST outperforms the method of Huang and Yates (2009) on both data sets.

Table 8.2 also performs comparison of absolute training time and the relative magnitude of training time across the source model, target model, FAJIST and Huang and Yates (2009). The relative magnitude of training time is calculated w.r.t. FAJIST,i.e., relative time of method $x = \frac{\text{training time of method x}}{\text{training time of FAJIST}}$. In this case, smaller is better.

From Table 8.2, we see that training the source model is 8 times slower than training the target model. This is due to the fact that the number of source domain labeled instances is much larger than the number of *selected* target domain instances (17 times in our experiments). FAJIST requires no additional training beyond training the target only model. The time of Huang and Yates (2009) can be decomposed into two parts: time for learning the HMM over combined source and target domain unlabeled data and time for retraining the source model using the HMM based features. We observed that in our experiments, both phases took roughly same amount of time. As stated before, the second step (retraining the source model) is $8x$ times slower. Therefore, overall the method of Huang and Yates (2009) is 16x times slower than FAJIST. Note that we do not include the training time of the source model in the training time of FAJIST, because, in FAJIST, the source model does not need to be trained for every target domain, whereas, in Huang and Yates (2009), both phases (training of an HMM and retraining of the source model) need to be done for every new domain.

Method	F1		Relative Time
	meetings	random	
concat	65	76.8	8x
(Daumé III, 2007)	65.5	**77.8**	8x
FAJIST	**68**	76.3	1x
HY09	64	75	16x

Table 8.3: Comparison of F1 scores and time required by the FAJIST with supervised adaptation methods. concat refers to retraining the model of Ratinov and Roth (2009) on the concatenation of the source and *selected* target labeled instances. Daumé III (2007) refers to the method of Daumé III (2007) trained over both the source and *selected* target labeled instances. Firstly, FAJIST outperforms both these approaches on the meetings data set and performs comparable on the random data set. FAJIST does not perform any training over the source labeled data and so it is 8x times faster than other approaches. Secondly, the supervised adaptation methods (concat & Daumé III (2007)) both outperform the state-of-the-art unsupervised adaptation method of Huang and Yates (2009) (referred to as HY09)

To answer our second research question, we want to compare the performance of supervised adaptation methods (using gold standard source labeled data and automatically generated and selected target labeled data) with that of the unsupervised adaptation methods. In Table 8.3, concat refers to retraining the model of Ratinov and Roth (2009) on the concatenation of the source and *selected* target labeled instances. Daumé III (2007) refers to the method of Daumé III (2007)

trained over both the source and *selected* target labeled instances. We see that both supervised adaptation methods outperform the unsupervised adaptation method of Huang and Yates (2009) significantly on both data sets.

From Table 8.3, we see that FAJIST outperforms both supervised adaptation approaches on the meetings data sets and achieves comparable performance on the random data set. However, both supervised adaptation approaches require training over the source labeled data and so they are 8x times slower than FAJIST.

Note that the performance of the supervised adaptation methods (concat & Daumé III (2007)) in Table 8.3 would be much worse if we select all target domain instances for inclusion in the target labeled data. This work wants to point out that only after the selection of target domain instances according to the selection criteria of FAJIST, these methods perform so well.

8.5 Summary

We showed that if we can create a target domain labeled data set by selecting the accurate predictions made by the source model on the target domain instances, we can apply the supervised adaptation approaches in unsupervised settings with superior performance. We proposed FAJIST that avoids training over the source labeled data and achieves comparable performance to the supervised approaches, with significantly less training time.

Chapter 9

Conclusion

Statistical machine learning based methods are currently the state-of-the-art for many natural language tasks. These models are typically trained on an annotated corpus of news stories, and then evaluated on data drawn from a similar source. The problem is that natural language differs considerably across different domains (e.g. text used in the news domain is very different from text in the biomedical domain, both in terms of vocabulary and structure). As a result, natural language processing systems which perform reasonably well on the news domain turn out to be significantly worse on other domains. There is a need to adapt statistical models trained on the labeled data from a domain so that they can be used on other domains. Consequently, Domain Adaptation has become one of the key problems of Natural Language Processing and in Machine Learning in general.

Although domain adaptation is important and essential for NLP to work in practice, all existing approaches for domain adaptation suffer from the requirement of model retraining. Retraining big models for every new domain is time consuming (e.g. 1 day for semantic role labeling (SRL)). Pipeline systems are especially difficult to retrain. Moreover, retraining the model for the target domain requires access to the labeled data from the source domain which may not be available to end users due to copyright issues.

In this thesis, I have studied whether it is possible to adapt to a target domain without building a new model for it. Several frameworks have been proposed, including transformation of text from the target domain to look statistically more similar to the training domain, using prior knowledge from the target domain, mining knowledge from both target domain unlabeled data and unlabeled data from an intermediate domain between the source and target domains, and joint inference with a task that is more robust across domain variations.

9.1 Future Research Directions

Several promising directions for future research are discussed below.

9.1.1 Extension of the Proposed Methods to Labeled Adaptation Scenario

Almost all methods proposed in this thesis assume that no labeled data is available in the target domain. How can we extend these methods to the scenario where a small amount of labeled data is available in the target domain? Specifically, the following research questions are worth investigation

- In Chapter 8, we showed that we can train a model only on the selected target domain instances where we believe the prediction of the source model. After training this model, at inference time, we do joint inference of the source model and the target model. Training the target model is much less time-consuming than training a single model over combined source and target labeled data. In this case, our problem was unsupervised adaptation, i.e., we did not have any labeled data in the target domain at the beginning.

  For the labeled adaptation case, we start with a small amount of labeled data in the target domain. Almost all existing methods proposed for this setting (e.g. (Daumé III, 2007)) discard the source model and train a new model on the entire source and target labeled data. The question is how we can use the source model, minimize training time for the target model and combine them during inference time, such that a similar level of accuracy w.r.t. existing methods can be achieved, with significantly less computational cost.

- In Chapter 7, we showed that the joint inference of a named entity recognition system and an wikification system improved the performance of the NER across different domains. Again, if we have some labeled data in the target domain, we can learn a reranker to rerank the outputs of the models for the two tasks such that the performance of the models for the two tasks improves on the target domain.

9.1.2 Joint Inference of NLP Tasks

In Chapter 7, we showed that if there are two tasks, where the system for one of the tasks exhibits robust performance across domains and the system for the other task has a weak performance

on domains other than the training domain, joint inference of the systems for the two tasks can help the performance of the weaker system on target domains. This principle can be extended to adaptation of tasks other than named entity recognition or Wikification. Specifically, joint inference for the following tasks is worth investigation

- Shallow parsing is believed to be more robust against domain variations whereas syntactic parsing is more vulnerable to domain variations. Jointly performing these two tasks might improve the performance of a syntactic parser across domains.

- Joint inference of the tasks of named entity recognition and coreference resolution can also help in adaptation to new domains. In many cases, when an NER system trained on one domain is applied to a new domain, the system fails to predict consistent entity types for different realizations of surface forms of the same underlying entity. For example, the NER system might predict "George Bush" as an entity of type PERSON but fail to predict "George H.W. Bush" as PERSON. But a coreference system might detect the mentions of "George H.W. Bush" and "George Bush" to be coreferent based on matching the last and first names of the two mentions. In Chapter 6, we use a constraint that tries to enforce the same entity type for different occurrences of the same phrase. This constraint will not apply to the example mentioned above since the two phrases are not exactly the same but are minor variations of each other. This is where a coreference system can be used to generalize the constraint of the same entity type for different surface form realizations of the same entity.

9.1.3 Domain Robust Models

A natural extension of the paradigm proposed in this thesis is to consider the scenario where we will build a model that can work well on many different domains without any need for adaptation. One way to accomplish this is to generate word embeddings from corpora spanning many different domains and use these word embeddings for training the models to provide robustness against domain variations.

References

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

Bacchiani, M. and Roark, B. (2003). Unsupervised language model adaptation. In *ICASSP*.

Ben-David, S., Blitzer, J., Crammer, K., Pereira, F., et al. (2006). Analysis of representations for domain adaptation. In *NIPS*.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007). Greedy layer-wise training of deep networks. In *NIPS*.

Bengio, Y., Schwenk, H., Senécal, J.-S., Morin, F., and Gauvain, J.-L. (2006). Neural probabilistic language models. In *Innovations in Machine Learning*. Springer.

Bickel, S., Brüeckner, M., and Scheffer, T. (2007). Discriminative learning for differing training and test distributions. In Ghahramani, Z., editor, *ICML*.

Bishop, C. M. (1996). *Neural networks for pattern recognition*.

Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Wortman, J. (2008). Learning bounds for domain adaptation. In *NIPS*.

Blitzer, J., Dredze, M., and Pereira, F. (2007). Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*.

Blitzer, J., Kakade, S., and Foster, D. P. (2011). Domain adaptation with coupled subspaces. In *International Conference on Artificial Intelligence and Statistics*, pages 173–181.

Blitzer, J., McDonald, R., and Pereira, F. (2006). Domain adaptation with structural correspondence learning. In *EMNLP*.

Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *COLT*.

Brown, P., Pietra, V. D., deSouza, P., Lai, J., and Mercer, R. (1992). Class-based n-gram models of natural language. *Computational Linguistics*.

Bunescu, R. C. and Pasca, M. (2006). Using encyclopedic knowledge for named entity disambiguation. In *EACL*.

Carreras, X. and Màrquez, L. (2005). Introduction to the conll-2005 shared task: Semantic role labeling . In *CoNLL*.

Chang, M., Ratinov, L., Rizzolo, N., and Roth, D. (2008). Learning and inference with constraints. In *AAAI*.

Chang, M.-W., Srikumar, V., Goldwasser, D., and Roth, D. (2010). Structured output learning with indirect supervision. In *ICML*.

Chapelle, O., Schölkopf, B., Zien, A., et al. (2006). *Semi-supervised learning*, volume 2. MIT press Cambridge.

Charniak, E. and Johnson, M. (2005). Coarse-to-fine n-best parsing and maxent discriminative reranking. In *ACL*.

Chelba, C. and Acero, A. (2004). Adaptation of maximum entropy capitalizer: Little data can help a lot. In *EMNLP*.

Chen, M., Xu, Z., Weinberger, K., and Sha, F. (2012). Marginalized denoising autoencoders for domain adaptation. In *ICML*.

Ciaramita, M. and Altun, Y. (2005). Named-entity recognition in novel domains with external lexical knowledge. In *Workshop on Advances in Structured Learning for Text and Speech Processing (NIPS)*.

Collins, M. (2002). Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *EMNLP*.

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*.

Cucerzan, S. (2007). Large-scale named entity disambiguation based on wikipedia data. In *EMNLP-CoNLL*.

Dai, W., Yang, Q., Xue, G., and Yu, Y. (2007). Boosting for transfer learning. In *ICML*.

Daumé III, H. (2007). Frustratingly easy domain adaptation. In *ACL*.

Daumé III, H., Langford, J., and Marcu, D. (2009). Search-based structured prediction. *Machine learning*, 75.

Daumé III, H. and Marcu, D. (2006). Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence*.

Deerwester, S., Dumais, S. T., Furnas, G. W., and Landauer, T. K. (1988). Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 41.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*.

Finkel, J., Dingare, S., Manning, C. D., Nissim, M., Alex, B., and Grover, C. (2005a). Exploring the boundaries: Gene and protein identification in biomedical text. *BMC Bioinformatics*, 6.

Finkel, J. R., Grenager, T., and Manning, C. (2005b). Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL*.

Finkel, J. R. and Manning, C. D. (2009). Hierarchical bayesian domain adaptation. In *NAACL*.

Foster, D., Kakade, S., and Salakhutdinov, R. (2011). Domain adaptation: Overfitting and small sample statistics. *arXiv preprint arXiv:1105.0857*.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*.

Han, X. and Zhao, J. (2009). Named entity disambiguation by leveraging wikipedia semantic knowledge. In *CIKM*.

Hirschman, L., Colosimo, M., Morgan, A., and Yeh, A. (2005). Overview of biocreative task 1b: normailized gene lists. *BMC Bioinformatics*, 6.

Hoffart, J., Yosef, M. A., Bordino, I., Fürstenau, H., Pinkal, M., Spaniol, M., Taneva, B., Thater, S., and Weikum, G. (2011). Robust disambiguation of named entities in text. In *EMNLP*.

Huang, F. and Yates, A. (2009). Distributional representations for handling sparsity in supervised sequence-labeling. In *ACL*.

Huang, F. and Yates, A. (2010). Open-domain semantic role labeling by modeling word spans. In *ACL*.

Huang, J., Gretton, A., Borgwardt, K. M., Schölkopf, B., and Smola, A. J. (2006). Correcting sample selection bias by unlabeled data. In *NIPS*.

Jebara, T. and Pentland, A. (1998). Maximum conditional likelihood via bound maximization and the cem algorithm. In *NIPS*.

Jiang, J. and Zhai, C. (2006). Exploiting domain structure for named entity recognition. In *NAACL*.

Jiang, J. and Zhai, C. (2007a). Instance weighting for domain adaptation in nlp. In *ACL*.

Jiang, J. and Zhai, C. (2007b). A two-stage approach to domain adaptation for statistical classifiers. In *CIKM*.

Klein, D. and Manning, C. D. (2003). Fast exact inference with a factored model for natural language parsing. In *NIPS*.

Koo, T., Rush, A. M., Collins, M., Jaakkola, T., and Sontag, D. (2010). Dual decomposition for parsing with non-projective head automata. In *EMNLP*.

Kumar, A., Saha, A., and Daumé, H. (2010). Co-regularization based semi-supervised domain adaptation. In *Advances in Neural Information Processing Systems*, pages 478–486.

Kundu, G. and Roth, D. (2011). Adapting text instead of the model: An open domain approach. In *CoNLL*.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Levin, B. (1993). *English Verb Classes and Alternations: A Preliminary Investigation*.

Liang, P. (2005). Semi-supervised learning for natural language. Master's thesis, Massachusetts Institute of Technology.

Minkov, E., Wang, R. C., and Cohen, W. W. (2005). Extracting personal names from email: applying named entity recognition to informal text. In *EMNLP*.

Mnih, A. and Hinton, G. (2007). Three new graphical models for statistical language modelling. In *ICML*.

Nivre, J., Hall, J., Kübler, S., McDonald, R., Nilsson, J., Riedel, S., and Yuret, D. (2007). The conll 2007 shared task on dependency parsing. In *CoNLL*.

Punyakanok, V., Roth, D., and Yih, W. (2008). The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*.

Raina, R., Battle, A., Lee, H., Packer, B., and Ng, A. (2007). Self-taught learning: transfer learning from unlabeled data. In *ICML*.

Ratinov, L. and Roth, D. (2009). Design challenges and misconceptions in named entity recognition. In *CoNLL*.

Ratinov, L., Roth, D., Downey, D., and Anderson, M. (2011). Local and global algorithms for disambiguation to wikipedia. In *ACL*.

Riedel, S. and Clarke, J. (2006). Incremental integer linear programming for non-projective dependency parsing. In *EMNLP*.

Roark, B. and Bacchiani, M. (2003). Supervised and unsupervised pcfg adaptation to novel domains. In *NAACL*.

Roth, D. and Yih, W. (2004). A linear programming formulation for global inference in natural language tasks. In *CoNLL*.

Sil, A. and Yates, A. (2013). Re-ranking for joint named-entity recognition and linking. In *CIKM*.

Surdeanu, M., Màrquez, L., Carreras, X., and Comas, P. R. (2007). Combination strategies for semantic role labeling. *Journal of Artificial Intelligence Research*, 29.

Tjong Kim Sang, E. F. and De Meulder, F. (2003). Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *CoNLL*.

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

Tsochantaridis, I., Hofmann, T., Joachims, T., and Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. In *ICML*.

Turian, J., Ratinov, L., and Bengio, Y. (2010). Word representations: a simple and general method for semi-supervised learning. In *ACL*.

Wu, D., Lee, W. S., and Ye, N. (2009). Domain adaptive bootstrapping for named entity recognition. In *EMNLP*.

Wu, P. and Dietterich, T. G. (2004). Improving svm accuracy by training on auxiliary data sources. In *ICML*.

Xue, N. and Palmer, M. (2004). Calibrating features for semantic role labeling. In *EMNLP*.

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervied methods. In *ACL*.