STRUCTURED PREDICTION WITH INDIRECT SUPERVISION

BY

MING-WEI CHANG

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, 2011

Urbana, Illinois

Doctoral Committee:

Professor Dan Roth, Chair, Director of Research
Professor Gerald DeJong
Assistant Professor Derek Hoiem
Associate Professor Noah Smith, Carnegie Mellon University

Abstract

Structured tasks, which often involve many interdependent decisions for each example, are the backbone for many important applications such as natural language processing tasks. The models built for structured tasks need to be capable of assigning values to a set of interdependent variables. In this thesis, we point out that the strong dependencies between the decisions in structured tasks can be exploited to simplify both the learning task and the annotation effort — it is sometimes possible to supply partial and indirect supervision to only some of the target variables or to other variables that are derivatives of the target variables and thus reduce the supervision effort significantly.

Based on this intuition, this thesis addresses the problem of reducing the cost of labeling for structural tasks. We tackle this problem by developing advanced machine learning algorithms that can learn and generalize from *indirect supervision* in addition to labeled data. Indirect supervision can come in the form of constraints or weaker supervision signals. Our proposed learning frameworks can handle both structured output problems and problems with latent structures. We demonstrate the effectiveness of the learning with indirect supervision framework for many natural language processing tasks.

To my wife, Yang-Tsun Lee, and my daughter, Amber Yong-Ching Chang.

Acknowledgments

I would like to thank my advisor Dan Roth for his supervision. It is really my pleasure to be in his research group, where he provides his students full support to pursuit their research goals. Dan always encourages us to conduct high-impact research and is willing to accept new ideas. Outside the research lab, I truly appreciate the fact that I can discuss everything including my personal issues with him. Above all, his commitment to research is the most important factor that makes him an amazing advisor.

It is my great honor to have Gerald DeJong, Derek Hoiem, and Noah Smith in the dissertation committee. Their academic work has been truly inspirational for this thesis. I am very grateful for their personal guidance and their valuable feedback.

I would also like to thank my brilliant and hard-working colleagues. Most of the ideas of this thesis are coming from the exciting discussions with them. I would like to thank Lev Rantiov, who is always very passionate about his research and often has a unique point of view. It is always fun to discuss with Vivek Srikumar, who truly understands the beauty of artificial intelligence and statistical machine learning. You always want to discuss your idea with Dan Goldwasser, as he is so bright such that he can pinpoint the key issue of your problem directly. It is my pleasure to work with James Clarke. He is one of few persons who truly know how to build useful stuffs. I enjoy challenging and being challenged by Rajhans Samdani with theoretical questions. It is also my privilege to work with Michael Connor, Quang Do, Alex Klementiev, Gourab Kundu, Nick Rizzolo, Mark Sammons, Kevin Small and Yuancheng Tu. This thesis cannot be completed without their help. This thesis also benefits from the discussions with Kai-Wei Chang, Ivan Titov, Sander Canisius, Yee Seng Chan, Alla Rozovskaya and Eric Bengston.

I am also like to express my deepest gratitude to my former advisor Chih-Jen Lin. I would like to thank him for his personal guidance and support. He is a genuine scientist and I am still learning from him everyday.

Finally, I want to thank my family for their support during my studies. Especially, I would like to thank my wife Yang-tsun Lee for her sacrifice and support. It has been really difficult for her because we need to be separated in the last year of my Ph.D. I also need to mention my lovely daughter, Amber, who brings

much joys to me and my family. You are one of the best things that ever happened to me.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AER	Alignment error rate
AP	Alternating Projection method
CCCP	Concave-Convex procedure
CCM	Constrained Conditional Model
CE	Constraints Estimation
CP	Cutting plane methods
CRF	Conditional Random Field
CoDL	COnstraint-Driven Learning
DCD	Dual coordinate descent methods
EG	Exponentiated gradient descent algorithm
EM	Expectation Maximization
GD	Gradient descent algorithm
HMM	Hidden Markov Model
IE	Information Extraction
ILP	Integer linear programming formulation
JLIS	Joint Learning with Indirect Supervision
LCLR	Learning over Constrained Latent Representations
MIRA	Margin Infused Relaxed Algorithm
MLN	Markov Logic Networks
ML	Maximum likelihood
MRR	Mean Reciprocal Rank
NELL	Never-Ending-Language-Learner
NE	Named entity
NLP	Natural Language Processing.
POS	Part-of-Speech

PR	Posterior Regularization
RTE	Recognizing textual entailment
SGD	Stochastic gradient descent algorithm
SP	Structured Perceptron
SSVM	Structural Support Vector Machines
UEM	Unified Expectation Maximization

Chapter 1

Introduction

Natural language processing (NLP) techniques have already made a significant impact on our daily life. Novel NLP techniques, from speech recognition to machine translation, make computers seem smarter and human communication easier. Nevertheless, while recent advances in statistical NLP are exciting, we are still far from providing a level of natural language understanding that can support challenging applications such as accurate information extraction from unstructured text. One of the key reasons that hinders the progress in this direction is that training statistical models for complex NLP tasks requires many training examples.

Natural language tasks are often referred to as *"structured tasks"* since they involve many interdependent decisions. The models built for these tasks need to be capable of assigning values to a set of interdependent variables. Unfortunately, it is challenging to collect labeled examples for these tasks, given that many decisions need to be made for a single example. We begin this chapter by first pointing out the problem of the standard supervision protocols.

1.1 Challenges: Motivating Example

Consider the task of *semantic parsing*, there the goal is to map a human query to a meaning representation. This task is important because it is at the heart of communicating with Artificial Intelligence agents. Let us consider how to build an intelligent database interface such that the machine can accept a natural language query directly such that it can translate the corresponding meaning representation into a database query to get the answer. One example human query is in Figure 1.1.

Previous works for semantic parsing [Zelle and Mooney, 1996; Tang and Mooney, 2001; Zettlemoyer and Collins, 2005; Ge and Mooney, 2005; Zettlemoyer and Collins, 2007; Wong and Mooney, 2007] proposed to use supervised learning algorithms for the semantic parsing task. Their methods therefore require a certain amount of *labeled* sentence-representation pairs. In other words, they assume that **the full supervision is provided at the level of the complex structures**, which unfortunately is very costly.

INPUT	What is the largest state that borders New York and Maryland?
OUTPUT	`largest( state( next_to( state(NY) ) AND next_to( state(MD) )))`

Figure 1.1: Semantic Parsing: Translate a human query into a meaning representation.

Note that multiple interdependent decisions are required in order to generate the output meaning representation. For example, New York can be a state (`state(NY)`) or a city (`city(NY)`). In the example, New York probably means New York state because Maryland is a state (`state(MD)`). Therefore, the mappings between words and logical predicates are interdependent. It is also important to know how to compose the final logical expression, given that `state(next_to(·))` and `next_to(state(·))` have very different meanings. Since the output structures are very complicated, annotating one example is extremely expensive (especially compared to annotating one example for standard classification tasks, e.g., a message for spam filtering). In other words, in the supervised learning setting, we assume that we have not only labels for every example but also labels for every decisions. Importantly, for this task, the annotators need to be *trained* such that they can write valid meaning representations. The need for expertise makes supervising this task even more expensive.

Semantic parsing is just one example of structured prediction task; there are many other different tasks with complex structures. The ultimate problem for the supervised approaches in the area of natural language processing is the diversity of natural language. There are many different languages and many explored and unexplored NLP applications, so it is just too expensive to provide enough supervision at the level of complex structures for all applications and languages.

1.2 Insight of the Thesis : Indirect Supervision

When learning models that assign values to variables, one needs to supply supervision for each target variable of interest. When learning to make structured predictions, the problem is significantly harder, since the learned model needs to assign values to multiple interdependent variables. The thesis presented here argues that, in fact, the interdependencies between the target variables can be exploited to simplify both the learning task and the annotation effort — it is sometimes possible to supply partial and indirect supervision to only some of the target variables or to other variables that are derivatives of the target variables and thus reduce the supervision effort significantly.

This intuition brings us the main insight of this thesis. In the binary classification problem, for a given example, you either have labels or not (see the upper part in Figure 1.2). However, in the structure output

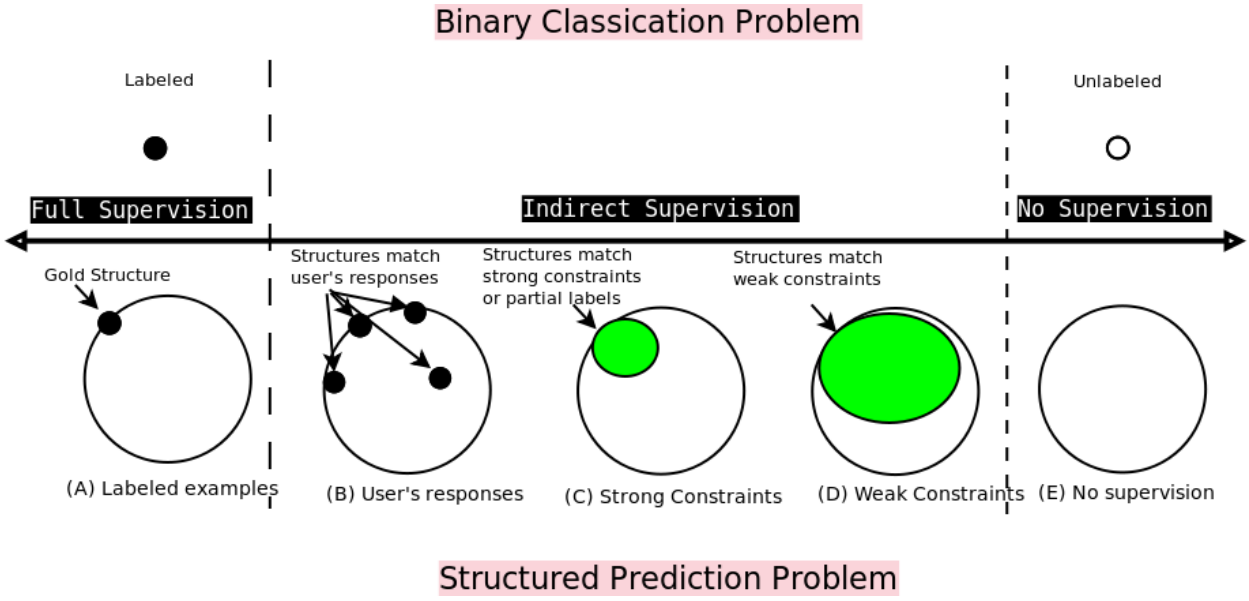


Figure 1.2: Insight for indirect supervision. (**TOP**) Binary classification problem. For a given example, there are only two modes – either it has label or not. (**BOTTOM**) Structured Prediction problem. A (big) circle represents all possible output structures of this example. In (A), we have the full supervision, which indicates the gold structure. In the right most circle (E), there is no supervision. Unlike binary classification problems, we now have *multiple levels* of supervision signals. We refer to these signals as indirect supervision. We can view constraints, partial labels and user responses as different indirect supervision signals. In (B), (C) and (D), we illustrate the structures that match the corresponding indirect supervision signals. While the indirect supervision signals does not reveal the gold structure, they still carry information that we might be able to take advantages from. See the text for more detailed discussions.

prediction, for a given example, there exists *other levels of supervision signals*. In Figure 1.2, we use a (big) circle to represent the all possible structures. When we gave *a fully labeled example*, we know the single gold structure in the set (Figure 1.2(A)). *An unlabeled example* is shown in Figure 1.2(E), where we have no clues about which structure is correct. Interestingly, unlike binary classification problems, there exist *multiple levels of supervision signals for structured prediction problems* (Figure 1.2(B,C,D)). In this thesis, we study *several forms of supervision signals* including constraints and binary supervision signals, as they have different utilities in various scenarios. While these supervision signals cannot reveal the correct structures, they still convey information that we can learn from.

Constraints as indirect supervision signals We can view constraints as indirect supervision signals. For example, all possible meaning representations that can be generated from the human query in the semantic parsing example forms a very large set. If we take advantages of some prior knowledge and inject some constraints (for example, "New York" and "Maryland" need to represent similar things, the word "borders" should represent the `next_to (·)` predicate, and `next_to` predicate can only accept locations

as arguments), the set becomes much smaller. Note that constraints are only indirect signals, they make the set of possible structures smaller but cannot provide the correct structure. Examples are illustrated in Figure 1.2 (C,D).

Why do we want to view constraints as supervision signals? While constraints provide useful information for this particular example, but at test time, we might not be able to apply the constraints (e.g., we do not see the phase "New York", "Maryland" or "borders" in our testing example). Nevertheless, this does not mean that the constraints have no value. If we can develop machine learning frameworks that learn from knowledge carried by constraints (given that constraints have some information about the gold structure), constraints are supervision signals. In other word, we want to have machine learning frameworks that can *generalize* the information carried by constraints.

User's response as indirect supervision signals We can also treat user's response as an *indirect* supervision resource. For instance, in semantic parsing, we can assume that there exists a *teacher* that will provide feedback on whether the answer generated from our model is correct or not. In the above example, the feedback becomes: *Is the generated answer Pennsylvania?* While this supervision signal is binary, it has deep connections to the meaning representation — if we get the answer correct, it is very likely that the generated meaning representation is also correct. We demonstrate the idea by using Figure 1.2(B). Assume that there are four meaning representations that generate the answer "Pennsylvania", so the supervision signal is again indirect. However, the fact that the response drops most of the other structures indicates that it carries a lot of information. Again, the key is to have learning frameworks that can *generalize* the information carried by the user's responses.

Constraints versus Supervision Interestingly, we can also view the full supervision as constraints, since it constrains the structure that you can choose for this example. However, while this constraint works perfectly in this example, the key is to have learning frameworks can that *generalize* the knowledge. In this thesis, we want to have frameworks that can learn from indirect supervision signals, just like the traditional machine learning methods do for fully labeled examples.[1]

We refer to the traditional supervision resources, input-output pairs, as *direct supervision signals*, as they provide the correct target output for each example. We refer to supervision signals that do not directly provide the corresponding correct output of an example as *indirect supervision signals*. See the illustrations in Figure 1.2.

[1]One can also consider partial labels (for example, in the POS tagging task, only certain words are labeled) as indirect supervision signals [Do and Artières, 2009] (circle C in the Figure 1.2).

1.3 Goals of the Thesis

Our main goal is to **develop machine learning frameworks that can learn (and generalize) from various forms of indirect supervision signals such that we can reduce the cost of labeling for structured tasks**. In contrast to previous approaches which often design application specific algorithms, we develop principled and general purpose machine learning algorithms that can take advantage of *expressive high-level human knowledge* and *easy-to-get indirect supervision signals*. Unlike machines, which are good at performing large scale calculations and capturing statistical relations, humans can often easily contribute high-level knowledge that captures interdependence between decisions. While machines can eventually capture high-level information, they often require a significant amount of fully labeled examples. Therefore, it is important to have a learning framework that can both learn from labeled examples and make use of easy-to-get indirect supervision signals.

Two different notions of "structure" will be discussed in this thesis. First, many NLP problems can be considered as "structured output prediction" tasks where, given an instance, the goal is to output many interdependent output variables. For example, consider the task of extracting information from an apartment advertisement. We need to determine what type of information each word/phrase conveys (e.g., rent, features or neighborhood of the apartment). On the other hand, "structure" also exists as an intermediate level between input and output representations. Consider the task of paraphrase identification, a task of judging whether one sentence is a paraphrase of an other sentence or not. Unlike the first type of problem, now there is only one output decision: "yes" or "no." Nevertheless, there still exists a notion of *structure* for paraphrases: a sentence can be considered a paraphrase of another only if the syntactic/semantic structures of these two sentences can be properly aligned. Unfortunately, such alignments are typically not labeled in the training examples — the structure is *latent*. Given that we have two notions of structure, the question is whether we can develop a unified algorithmic paradigm for both that — most importantly — makes use of high-level human knowledge as indirect supervision to learn to predict these structures or improve the final task performance.

1.4 Overviews of the Solutions

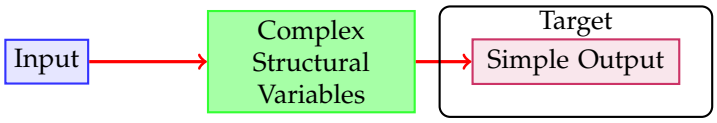
We first address the "structured output" task of incorporating prior human knowledge as *constraints* (In **Chapter 3**). Note that constraints belong to indirect supervision signals, given that they cannot reveal the full structures directly. For the task of extracting information from an advertisement post, one constraint may be that information of the same type should be grouped together. We propose a general framework

called *Constraint Driven Learning*, which is capable of combining statistical models and general expressive constraints in a principled way. We show that models combined with this type of high-level knowledge require significantly fewer numbers of labeled examples to extract information from an advertisement post compared to standard statistical models. We generalize the framework and propose the *Constrained Conditional Model (CCM)*, which provides a *direct* way to inject prior knowledge into a conditional model, in the form of **constraints**. Our design principle is to use constraints to *simplify* the modeling of a complex structured output problem. Instead of building a model from complex features, our model provide a way to combine "simple" learned models with a few "expressive" constraints. This idea is now broadly used in the NLP community.

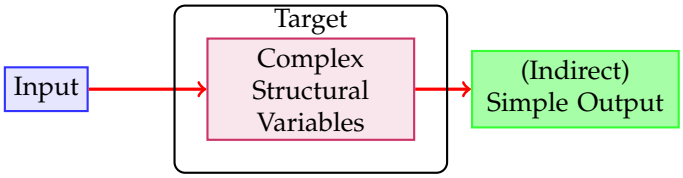
Next, we discuss the relationship between the Constraint Driven Learning framework and the Expectation Maximization (EM) algorithm [Dempster et al., 1977], when there are no labeled examples. Many unsupervised and semi-supervised learning frameworks are based on the EM algorithm. Due to different requirements and issues, many EM variations have been proposed. In this thesis, we propose a unified expectation maximization framework called UEM (**Chapter 4**), which can be considered as a generalization over the Constraint Driven Learning framework. UEM allows injecting constraints as indirect supervision, and covers many different EM variations. The UEM framework provides us a new platform to compare different EM variations and help us invent new learning algorithms.

We then move the focus to tasks with "latent structures" (**Chapter 5**). We develop an algorithmic solution that allows us to inject another type of indirect supervision: knowledge of the *latent* structure of a given task. For paraphrase identification, for example, while the target output is a simple binary decision, the latent structures can be defined as the alignments between the syntactic/semantic structures of two sentences. We show that using our new algorithm, we can use simply stated and easily available background knowledge within a large margin statistical framework and train systems for paraphrase identification and other tasks to get state-of-the-art results. Figure 1.3(a) illustrates the idea of our approach.

Interestingly, we can combine these two notions of "structure" in a unified and principled statistical learning framework (**Chapter 6**). We present a novel approach to structure prediction that is based on the following observation: structured output problems often have a companion learning problem – determining whether a given input possesses a "good" structure. For example, the companion problem for part-of-speech (POS) tagging is whether a given sequence of words has a corresponding sequence of POS tags that is "legitimate." While obtaining *direct supervision* for structures is difficult, it is often very easy to obtain this type of binary indirect supervision — labeled data for the companion binary decision problem. We develop a large margin learning framework that jointly learns both direct and indirect forms of



(a) (**Chapter 5**): Learning complex intermediate representations for binary classification problems.



(b) (**Chapter 6**): Supervising complex structures with easy-to-get binary labels as indirect supervision signals

Figure 1.3: Between Chapter 5 and 6, this thesis points out the relationship between frameworks for large margin structured output and frameworks for large margin binary output with latent structures (note the symmetry between (a) and (b), while their target tasks are different). This discovery makes joint learning from direct and indirect supervision signals possible.

supervision. Our experiments demonstrate the significant contribution of the easy-to-get indirect binary supervision on many important NLP tasks.

Figure 1.3(b) illustrates the idea of using indirect supervision signals. For a given input example, we would like to train a model that can annotate complex structures. However, while generating complex structures is our target, obtaining supervision signals at this level is too expensive. We would like to design principled machine learning methods that allow accepting indirect supervision signals, which can be driven by related tasks, invented by human or extracted from from existing prior knowledge.

1.5 Contributions of the Thesis

Due to the high supervision cost for the NLP tasks, people have investigated the possibilities of applying unsupervised and semi-supervised learning techniques to many important NLP tasks including part-of-speech tagging [Merialdo, 1994; Smith and Eisner, 2005; Haghighi and Klein, 2006; Johnson, 2007], information extraction [Thelen and Riloff, 2002; Grenager et al., 2005; Haghighi and Klein, 2006], named entity recognition [Collins and Singer, 1999; Cohen and Sarawagi, 2004], parsing [Klein and Manning, 2004; Smith and Eisner, 2006; Spitkovsky et al., 2010] and co-reference resolution [Haghighi and Klein, 2010]. However, many of these approaches only restrict themselves to limited forms of supervision signals while some supervision signals are available. While some approaches adopt other forms of prior knowledge to bias the model, most of them do not define the prior knowledge formally and do not provide a general platform to incorporate declarative constraints. Moreover, it is not clear how to integrate different types of supervision

signals and labeled examples together. Finally, the relations between various learning frameworks are often not studied.

This thesis addresses these issues, and make the following key contributions:

- **Principled and general machine learning frameworks for various forms of indirect supervision**

  Throughout this thesis we have developed several general purpose learning frameworks. In Chapter 3, we propose Constrained Conditional Model as a general platform for injecting declarative constraints as indirect supervision signals. In Chapter 5, we provide the first platform that allows incorporating expressive constraints for *latent structures*, and allows joint learning of the simple binary output and latent structures. In Chapter 6, we propose a learning framework that allows the use of both constraints and binary labeled data for learning structures. Regarding indirect supervision signals, we discuss various forms of constraints and extend simple dictionary look up constraints to more general and long distance constraints that are incorporated into and bias the model learned. We also discuss the possibility of *inventing* binary supervision signals or using existing binary supervision signals for learning structures.

- **Making use of different levels of supervision signals**

  We believe that it is necessary to integrate all types of supervision signals, including labeled examples. In both Chapter 3 and 6, our framework allows *integrating indirect supervision signals together with direct supervision signals*. Compared to unsupervised learning frameworks, frameworks that are able to combine both labeled examples and prior knowledge have been much less studied.

- **Understanding the relations between different learning frameworks**

  In Chapter 4, we analyze different EM variations and propose a unified constrained EM framework. The framework helps us find new variations that could be potentially useful. Moreover, between Chapter 5 and 6, this thesis points out the relationship between frameworks for large margin structured output and frameworks for large margin binary output with latent structures (note the symmetry between Figure 1.3(a) and 1.3(b)). This discovery makes joint learning from direct and indirect supervision signals possible.

- **Advancing state-of-the-art NLP task results**

  Many of our proposed frameworks advance start-of-the-art results. At the time we published the papers, several of the systems, including the IE system (Chapter 3) and the Paraphrasing system (Chapter 5) achieved new state-of-the-art results.

In the next chapter, we provide a gentle introduction for the necessary background knowledge for structured output prediction tasks; the details of our indirect supervision approach will be introduced in the following chapters.

We have not included all our related work in this document. For example, in [Clarke et al., 2010], we show how to use world's response as indirect supervision for improving the task of semantic parsing (the example showed in Figure 1.1). We will discuss these works and more recent works on using indirect supervision signals in Chapter 7.

Chapter 2

Background

The necessary background knowledge for structured output prediction learning frameworks and our notations are provided in this chapter. Understanding learning algorithms for structural prediction tasks is very important in this thesis, since our goal is to improve these frameworks to allow injecting high-level human knowledge and easy-to-get indirect supervision signals.

In this chapter, we mainly focus on general knowledge for structure learning, and some of the other advanced background knowledge will be reviewed in each chapter.

Besides reviewing the related work, we also describe an algorithmic contribution of this thesis and justify the choice of our design in this chapter. Moreover, we summarize different optimization methods for structural support vector machine [Taskar et al., 2004; Tsochantaridis et al., 2005] in Section 2.4.

2.1 Notations

Many NLP tasks can be considered as structured output prediction tasks. For example, the problem of Part-of-Speech (POS) tagging (Figure 2.1(a)) is a structured prediction task, where an input example is a sentence and the output is the corresponding Part-of-Speech sequence. Another example of a structured prediction task is dependency parsing. The output of dependency parsing is a tree which exhibits the syntactic relationship between words in the input sentence (Figure 2.1(b)). Note that the output space of both problems consists of many interdependent decisions, and hence separates them from standard binary/multiclass classification tasks.

(a) Part-Of-Speech Tagging (b) Dependency Parsing

Figure 2.1: Example structured output prediction NLP tasks.

We first summarize the notations that will be used throughout this thesis in Table 2.1. In order to sep-

arate the notation of the binary output problem and the structured output problem, we use the symbol $z \in \{-1, 1\}$ to represent binary output, $\mathbf{y}$ to represent the output structure and $\mathbf{h}$ to represent latent structure. One exception is in the Chapter 6, where the symbol $\mathbf{h}$ has two different meanings.

Sometimes it is convenient to treat a structure $\mathbf{y}$ (or $\mathbf{h}$) as a vector, given that each structure consists of multiple decisions. For example, we can write

$$\mathbf{y} = (y^1, y^2, \ldots, y^T),$$

where y^i is a variable representing the i-th decision of $\mathbf{y}$, and T is the total number of decisions. For example, given a sentence, we can define that y^1 represents the POS tag of the first word, y^2 represents the POS tag of the second word, and so on.

The feature vector $\Phi(\mathbf{x}, \mathbf{y})$ is a function defined over an input-output pair $(\mathbf{x}, \mathbf{y})$. When discussing the relationships between structured output models and binary classification models, we also use the notation $\Phi(\mathbf{x})$ to match the standard definition of the feature vector used in the "non-structural" models. See the discussions in Section 2.2 for more details.

When we mention the word "constraint" in this thesis, without special notice, we refer to the constraint defined over a structure. A constraint is denoted as Ψ. Ψ can be described by first order logic. For example, example constraints can be: "there must be at least one verb in a sentence", or "the verbs cannot appear consecutively in a sentence". In the thesis, some of the proposed frameworks only accept linear constraints. Fortunately, we can convert Ψ by "propositionalize" it into a finite set of linear inequalities. In [Rizzolo and Roth, 2007], the author states an efficient way of performing such conversions. In this thesis, we write inequality constraints behind Ψ as $u(\mathbf{x}, \mathbf{y}) \leq b$, where $u(\mathbf{x}, \mathbf{y})$ is a linear function over $\mathbf{y}$ (it can be a non-linear function over $\mathbf{x}$).

The loss function $\ell : \mathbb{R} \to \mathbb{R}$ can be instantiated by many commonly used loss functions such as hinge loss, with $\ell(a) = \max(0, a)$ or squared-hinge loss, with $\ell(a) = \max(0, a)^2$.

In the following, we review the basics of structured output prediction and its relationships to other models in Section 2.2. We review several optimization techniques in Section 2.3 and compare the properties of different algorithms in Section 2.4. Finally, we mention several semi-supervised learning algorithms in Section 2.5.

$\mathbf{w}$	a weight vector
$\Phi(\cdot)$	a feature vector generation function
$\mathbf{x}$	an input example
$\mathbf{y}$	an output structure
$\mathbf{h}$	a latent structure
Ψ	a (first-order like) constraint (See the text for more details)
z	an output binary variable, where $z \in \{-1, 1\}$
$\mathcal{Y}(\mathbf{x})$	all possible output structures of $\mathbf{x}$
$\mathcal{H}(\mathbf{x})$	all possible latent structures of $\mathbf{x}$
$\Delta(\mathbf{y}, \mathbf{y}')$	distance between structures $\mathbf{y}$ and $\mathbf{y}'$
$\ell(\cdot) : \mathbb{R} \rightarrow \mathbb{R}$	a loss function (hinge loss, square hinge loss, etc.)

Table 2.1: Notations that are used in this thesis.

2.2 Structured Output Prediction: A Gentle Introduction

The supervised training data for structured prediction tasks can be defined as follows: let $\mathcal{S} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{l}$ denote l labeled examples, where $\mathbf{x}_i$ represents the i-th example and $\mathbf{y}_i$ represents the corresponding correct structure. In the POS tagging example, $\mathbf{x}$ represents a sentence and $\mathbf{y}$ represents the gold POS sequence. In the dependency parsing examples, $\mathbf{y}$ represents the gold standard dependency tree corresponding to the input sentence $\mathbf{x}$.

In this thesis, we focus on using linear models for structural prediction tasks, where we can represent the model as $\mathbf{w}$ and make predictions by solving the following optimization problem:

$$\arg\max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x}_i)} \mathbf{w}^T \Phi(\mathbf{x}_i, \mathbf{y}). \tag{2.1}$$

This problem is often referred as the *decoding* problem. The set $\mathcal{Y}(\mathbf{x}_i)$ represents all possible structures that can be generated from the example $\mathbf{x}_i$. The unique feature of structured prediction models is the use of Eq. (2.1) to choose the best structure (according to $\mathbf{w}$) among possibly exponentially many structures in $\mathcal{Y}(\mathbf{x}_i)$.

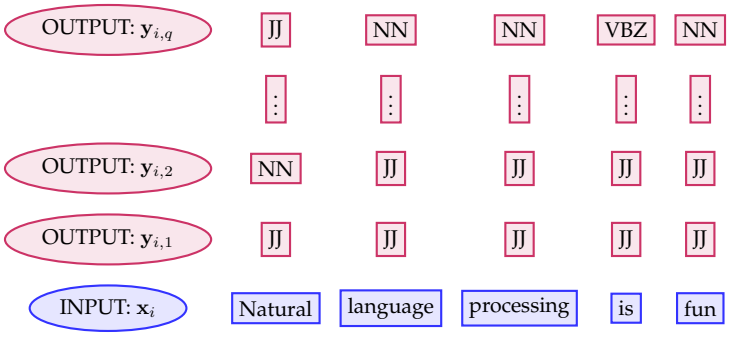


Figure 2.2: POS Tagging Example: $\mathcal{Y}(\mathbf{x}_i)$ for a sentence $\mathbf{x}_i$. See the text for more details.

Detailed Example: POS tagging Consider the POS tagging task as an example, for a given input $\mathbf{x}_i$, we would like to find the structure $\mathbf{y}_{i,*}$ among all possible structures $\mathcal{Y}(\mathbf{x}_i)$. Figure 2.2 displays an example showing that the contain of $\mathcal{Y}(\mathbf{x}_i)$. Assume $\mathbf{x}$ is the sentence "Natural language processing is fun." Note that this sentence contains 5 tokens. Assume that there are 45 different POS tags. The size of $\mathcal{Y}(\mathbf{x}_i)$ is $q = |\mathcal{Y}(\mathbf{x}_i)| = 45^5$. Each element $\mathbf{y}_{i,j}$ in $\mathcal{Y}(\mathbf{x}_i)$ represents a possible output for $\mathbf{x}_i$. Among these q sequences, one is the correct output $\mathbf{y}_{i,*}$ (In Figure 2.2, it is also $\mathbf{y}_{i,q}$).

Note that the feature vector of structured output prediction could be a function over both input $\mathbf{x}$ and output $\mathbf{y}$. For example, the feature vector $\Phi(\mathbf{x}, \mathbf{y})$ can contain features representing the conjunctions between a token and its tag (according to $\mathbf{y}$). It can also contain features such as the conjunctions between the j-th tag and $(j+1)$-th tag. According to this feature definition, $\Phi(\mathbf{x}_i, \mathbf{y}_{i,q})$ in Figure 2.2 has the following active features:

$$\{\text{Natural} - \text{JJ}, \text{language} - \text{NN}, \text{processing} - \text{NN}, \text{is} - \text{VBZ}, \text{fun} - \text{NN}, \text{JJ} - \text{NN}, \text{NN} - \text{NN}, \text{NN} - \text{VBZ}, \text{VBZ} - \text{NN}\}.$$

Note that size of $\Phi(\mathbf{x}_i, \mathbf{y}_{i,q})$ is very large (it contains all possible features in the training data) with only 9 active features for this specific example.

The weight vector assigns score to each feature, and hence assign score to each $(\mathbf{x}, \mathbf{y})$ pair. The prediction of the model is going to be selected by using the decoding problem Eq. (2.1). This procedure might be expensive if the feature Φ contains the long distance relationship of $\mathbf{y}$.

Learning Algorithms A learning algorithm can be considered as a procedure that finds a $\mathbf{w}$ according to labeled examples $\mathcal{S}$. While we have not introduced any learning algorithm, all learning algorithms try to reduce the training error (and try not to overfit to the data). In order to reduce the training error, a good $\mathbf{w}$ should try to satisfy the following constraint for every example $(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{S}$:

$$\mathbf{w}^T \Phi(\mathbf{x}_i, \mathbf{y}_i) \geq \max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x}_i)} \mathbf{w}^T \Phi(\mathbf{x}_i, \mathbf{y}). \tag{2.2}$$

The intuition behind this constraint is that a good weight vector should assign a higher score to the correct structure y_i than it assigns to any other structure.

In the following, we review commonly used models including binary classification models and multi-class classification models from the perspective of structure prediction models. We will then move on to sequential tagging models and finally to general structured output prediction methods. Note that we defer the discussion on the optimization algorithms to Section 2.3.

Algorithm 1 The Perceptron Learning Algorithm. Note that the feature function only depend on the input here.

Require: Learning rate η, Number of iteration N, Training Data $\mathcal{B} = \{(\mathbf{x}_i, z_i)\}_{i=1}^{l}$

1: $\mathbf{w} \leftarrow 0$
2: **for** $t = 1 \ldots N$ **do**
3: **for** $i = 1 \ldots l$ **do**
4: **if** $z_i \mathbf{w}^T \Phi(\mathbf{x}_i) \leq 0$ **then**
5: $\mathbf{w} \leftarrow \mathbf{w} + z_i \Phi(\mathbf{x}_i)$
6: **end if**
7: **end for**
8: **end for**

2.2.1 Binary and Multiclass Classification Models

While we focus on structured prediction problems, it is crucial to realize that structured output prediction models are more general than binary/multiclass classification models. In the following, we start by reviewing binary learning classification models from the perspective of predicting structures.

Discriminative Classifiers: Binary Linear Classifiers and Support Vector Machines Perceptron algorithm [Rosenblatt, 1958, 1962] is probably the earliest and the most important linear learning algorithm. As the computational power increased dramatically in the past few decades, more advanced linear classification models have been developed and used. Among the advanced linear classifiers, support vector machine (SVM) is the most influential model and has made significant impact both theoretically and practically.

The perceptron learning algorithm is described in Algorithm 1. The training algorithm is a mistake-driven algorithm, which only updates the weight vector (line 5) when the model makes a mistake (line 4). Unlike the perceptron algorithm, the SVM algorithm obtains the weight vector $\mathbf{w}$ by solving the following optimization problem:

$$\min_{\mathbf{w}} \frac{\|\mathbf{w}\|^2}{2} + C \sum_{i \in \mathcal{B}} \max(0, 1 - z_i \mathbf{w}^T \Phi(\mathbf{x}_i)), \tag{2.3}$$

where $\mathcal{B} = \{(\mathbf{x}_i, z_i)\}_{i=1}^{l}$, and C is a parameter that prevents overfitting. That is, C controls the balance between the regularization term $\frac{\|\mathbf{w}\|^2}{2}$ and the loss term. Note that throughout this thesis, for brevity, we write $i \in \mathcal{B}$ to indicate $(\mathbf{x}_i, z_i) \in \mathcal{B}$.

While the perceptron algorithm and SVM are different learning algorithms, they do share the same prediction function: $\operatorname{sgn}(\mathbf{w}^T \Phi(\mathbf{x}_i))$. That is, the model will predict the example as a positive example if and only if $\mathbf{w}^T \Phi(\mathbf{x}_i) \geq 0$. At a first glance, it seems that the structured prediction function (Eq. (2.1)) is different from the one used in perceptron and SVM. However, the decision function used for SVM is a special case of Eq. (2.1).

Note that the Boolean version of the perceptron algorithm and SVM uses a feature function that only

depends on the input $\mathbf{x}$, while the feature vector in Eq. (2.1) contains both input and the output variable $\Phi(\mathbf{x}, \mathbf{y})$. If we make

$$\Phi(\mathbf{x}, \mathbf{y} = 1) = \Phi(\mathbf{x}), \Phi(\mathbf{x}, \mathbf{y} = -1) = 0,$$

the decision function of structured output variables reduces to the decision function for the standard perceptron and SVM. More precisely,

$$\begin{aligned} \arg\max_{\mathbf{y} \in \{1,-1\}} \mathbf{w}^T \Phi(\mathbf{x}, \mathbf{y}) &= \arg\max_{\mathbf{y} \in \{1,-1\}} \begin{cases} \mathbf{w}^T \Phi(\mathbf{x}) & \text{if } \mathbf{y} = 1 \\ 0 & \text{if } \mathbf{y} = -1 \end{cases} \\ &= \begin{cases} 1 & \text{if } \mathbf{w}^T \Phi(\mathbf{x}) \geq 0 \\ -1 & \text{Otherwise.} \end{cases} \end{aligned}$$

We call this trick of reducing structured output prediction to binary output prediction "asymmetric feature reduction", as we only assign features to the positive class ($z = 1$). This trick is in fact very important when there are latent variables, and we will use this trick again in Chapter 5 and 6. Also see the discussion in Section 6.6 about this trick when there are latent variables.

Generative Model: naive Bayes The structured prediction function (2.1) also covers the prediction functions of many generative models. This implies that many generative models can be expressed as a linear classifier. The relationships between discriminative and generative linear classifiers have been discussed in many different scenarios [Domingos and Pazzani, 1997; Roth, 1999; Ng and Jordan, 2002]. In the following, we show how to reduce the decision function of a naive Bayes classifier to Eq. (2.1).

For the sake of simplicity, we assume that there are only two classes and discrete binary features when training a naive Bayes model. Therefore, all of the features used here can be represented as $[\mathbf{x}^j = k]$, an indicator function that has the value 1 if and only if the j-th feature of $\mathbf{x}$ has value k. A naive Bayes classifier models the probability distribution $P(\mathbf{x}, z)$ directly such that

$$\log P(\mathbf{x}, z) = \log P(\mathbf{x}|z)P(z) = \sum_j \sum_{k=0}^{1} \log P(\mathbf{x}^j|z)^{[\mathbf{x}^j = k]} + \log P(z),$$

because the naive Bayes model adopts the conditional independence assumption. The training procedure of a naive Bayes model simply tries to find the most likely value of $P(\mathbf{x}^j|z)$ and $P(z)$ — if you maximize the likelihood of $P(\mathbf{x}^j|z)$ and $P(z)$, you maximize that of $\log P(\mathbf{x}, z)$. The decision function for an example $\mathbf{x}$ is simply $\arg\max_z P(\mathbf{x}, z)$.

Define

$$\Phi(\mathbf{x}) = \left([\mathbf{x}^1 = 0], [\mathbf{x}^1 = 1], [\mathbf{x}^2 = 0], [\mathbf{x}^2 = 1], \ldots [\mathbf{x}^m = 0], [\mathbf{x}^m = 1], 1\right),$$

where m represents the number of input variables for a given example, and one addition dummy feature is active for all examples. Define $\mathbf{w}_1$ as a weight vector that has the same size of $\Phi(\mathbf{x})$, where for the feature $[\mathbf{x}^j = k]$, the corresponding weight in $\mathbf{w}_1$ has the value: $\log P(\mathbf{x}^j = k|z = 1)$. The corresponding weight value for the extra dummy feature is $\log P(z = 1)$. Similarly, we can define another weight vector $\mathbf{w}_{-1}$ for $z = -1$. Let

$$\Phi(\mathbf{x}, z = 1)^T = \begin{bmatrix} \Phi(\mathbf{x})^T & \mathbf{0}^T \end{bmatrix}, \Phi(\mathbf{x}, z = -1)^T = \begin{bmatrix} \mathbf{0}^T & \Phi(\mathbf{x})^T \end{bmatrix}, \mathbf{w}^T = \begin{bmatrix} \mathbf{w}_1^T & \mathbf{w}_{-1}^T \end{bmatrix},$$

then we get

$$\arg\max_{z \in \{1,-1\}} \mathbf{w}^T \Phi(\mathbf{x}_i, z) = \arg\max_{z \in \{1,-1\}} \log P(\mathbf{x}, z).$$

Therefore, the decision function of a naive Bayes model can be written in the form of Eq. (2.1).

Multi-Class Classification Models Many important information retrieval problems such as document classification tasks can be casted into multiclass classification problems. One-versus-all and all-versus-all are the commonly used procedures to build a multiclass classification classifiers based on binary classification classifiers [Allwein et al., 2000; Har-Peled et al., 2002]. We will focus on the one-versus-all procedure here. For more information about the comparisons between different multi-class classification models, the readers can refer to [Hsu and Lin, 2002].

Assume that there are M different classes and let the training data be $\mathcal{S} = \{(\mathbf{x}, \mathbf{y})\}$, where $\mathbf{y} \in \{1, 2, \ldots, M\}$. The one-versus-all approach is a general approach that builds M classifiers with any binary classification training algorithm, where the j-th classifier $\mathbf{w}_j$ is trained based on the training data $\mathcal{B}_j$. The definition of $\mathcal{B}_j$ is

$$\mathcal{B}_j = \{(\mathbf{x}, +1)|(\mathbf{x}, \mathbf{y}) \in \mathcal{S}, \mathbf{y} = j\} \cup \{(\mathbf{x}, -1)|(\mathbf{x}, \mathbf{y}) \in \mathcal{S}, \mathbf{y} \neq j\}.$$

At test time, the prediction of the one-versus-all prediction function becomes:

$$\arg\max_{i \in \{1,2,\ldots,M\}} \mathbf{w}_i \Phi(\mathbf{x}). \tag{2.4}$$

By rewriting

$$\mathbf{w} = \begin{bmatrix} \mathbf{w}_1 & \mathbf{w}_2 & \dots & \mathbf{w}_M \end{bmatrix}, \Phi(\mathbf{x}, i) = \begin{bmatrix} \overbrace{\mathbf{0} \dots \mathbf{0}}^{(i-1) \text{ blocks}} & \underbrace{\Phi(\mathbf{x})}_{\text{the } i\text{-th block}} & \overbrace{\mathbf{0} \dots \mathbf{0}}^{(M-i) \text{ blocks}} \end{bmatrix},$$

the decision function (Eq. (2.1)) would be the same prediction function as Eq. (2.4).

Note that in the one-versus-all approach, the training phase is done by training M classifiers independently. However, the prediction function is done by considering all of the M classifiers together, as shown in [Har-Peled et al., 2002], the right approach is to also consider all M classifiers in the training time. Hence, several techniques have been proposed to overcome this issue [Nilsson, 1965; Har-Peled et al., 2002]. Crammer and Singer [Crammer and Singer, 2002] proposed a joint model and wrote down M classifiers in one global objective function. This technique was later extended to the case of general structured output prediction problems [Taskar et al., 2004].

2.2.2 Sequence Tagging Models and General Structure Prediction Models

We now move the focus to the example tasks with more structures. We consider sequential tagging models first and then move to general structural models.

Generative Model: Hidden Markov Model Generative models specify a joint probability distribution over observations and the corresponding output structures. Many generative models have been proposed for structured prediction tasks [Rabiner and Juang, 1986; Eisner, 1996]. In the following, we review a very popular sequential generative model: the Hidden Markov Model (HMM). A (first-order) HMM is a generative model which models the joint probability of a series of tokens $\mathbf{x}$ and a sequence assignment $\mathbf{y}$. HMMs make an independence assumption that allows one to write the joint probability of $(\mathbf{x}, \mathbf{y})$ as follows:

$$P(\mathbf{x}, \mathbf{y}) = P(y_1) \prod_{i=2}^{T} P(y_i | y_{i-1}) \prod_{i=1}^{T} P(x^i | y^i), \tag{2.5}$$

where x^i is the i-th token in the input sequence, y^i is the i-th token in the output sequence, T is the number of tokens in this sequence, $P(y^1)$ represents the prior probabilities, $P(y^i | y^{i-1})$ represents the transition probabilities and $P(x^i | y^i)$ represents the emission probabilities.

Past works have shown that the prediction problem in HMMs can be viewed as a linear model over

Algorithm 2 Structured Perceptron

Require: Number of iteration N, Training Data $\mathcal{S} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{l}$
1: $\mathbf{w} \leftarrow 0, \mathbf{w}_{avg} \leftarrow 0$
2: **for** $t = 1 \dots N$ **do**
3: **for** $i = 1 \dots l$ **do**
4: $\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x}_i)} \mathbf{w}^T \Phi(\mathbf{x}_i, \mathbf{y})$.
5: $\mathbf{w} \leftarrow \mathbf{w} + \Phi_{\mathbf{y}_i, \hat{\mathbf{y}}(\mathbf{x}_i)}$
6: $\mathbf{w}_{avg} \leftarrow \mathbf{w}_{avg} + \mathbf{w}$
7: **end for**
8: **end for**
9: **return** $\mathbf{w}_{avg}/(Tl)$

"local" features [Roth, 1999; Collins, 2002]. That is, one can show that

$$\arg\max_{y} P(\mathbf{y}|\mathbf{x}) = \arg\max_{\mathbf{y}} \log P(\mathbf{x}, \mathbf{y}) = \arg\max_{\mathbf{y}} \mathbf{w}^T \Phi(\mathbf{x}, \mathbf{y}), \tag{2.6}$$

where $\mathbf{w}$ is a weight vector and Φ represents a feature function. Therefore, we can convert the probability tables of an HMM into a linear function represented by $\mathbf{w}$ with appropriate feature functions. In this representation, the feature function $\Phi(\mathbf{x}, \mathbf{y})$ is expressed as a set of features which contain "prior features", $\Phi_p(y^1)$, "transition features" , $\Phi_t(y^i, y^{i-1})$, and "emission features", $\Phi_e(x^i, y^i)$ [Roth, 1999]. In other words, there exists a one-to-one mapping between the active features and the associated probability representation, which can be rewritten in the form of a linear function.

Structured Perceptron The structured perceptron (SP) was first introduced by [Collins, 2002]. The algorithm (Algorithm 2) extends the mistake-driven idea of the Perceptron algorithm (mentioned in Algorithm 1) to the structured output case. In line 4, it finds the best structure for an example using the current weight vector. Then the weight vector is updated with the difference between the feature vectors of the true label and the prediction. Notice that this is a mistake-driven algorithm, which means that if the current weight vector successfully finds the correct output, the weights will not change. Inspired by the results of [Freund and Schapire, 1999], the algorithm maintains an averaged weight vector (lines 6 and 9), which is the final output. This technique has been shown to improve the generalization ability of the final model[Freund and Schapire, 1999]. However, while structured perceptron algorithm is simple and easy-to-implement, it does not capture the concept of margin and there is no easy method to select N, the number of iterations. In structured perceptron, the prediction function is Eq. (2.1).

Conditional Random Field Models Conditional Random Field (CRF) [Lafferty et al., 2001] can be viewed as a probabilistic discriminative model. CRF models the conditional probability by:

$$P(\mathbf{y}_i|\mathbf{x}_i, \mathbf{w}) = \frac{\exp^{\mathbf{w}^T \Phi(\mathbf{x}_i, \mathbf{y}_i)}}{\sum_{\mathbf{y} \in \mathcal{Y}(\mathbf{x}_i)} \exp^{\mathbf{w}^T \Phi(\mathbf{x}_i, \mathbf{y})}}. \tag{2.7}$$

The training of a CRF is done by maximizing the conditional log likelihood of the labeled examples. The objective function of a CRF can be written as follows:

$$\min_{\mathbf{w}} \frac{\|\mathbf{w}\|^2}{2} + C \sum_{i=1}^{l} \log \frac{\exp^{\mathbf{w}^T \Phi(\mathbf{x}_i, \mathbf{y}_i)}}{\sum_{\mathbf{y} \in \mathcal{Y}(\mathbf{x}_i)} \exp^{\mathbf{w}^T \Phi(\mathbf{x}_i, \mathbf{y})}}. \tag{2.8}$$

The denominator of (2.7) is a summation over all possible structures for this example. Often this is the main computation bottleneck for training a CRF model, given that it contains exponential number of structures. Fortunately, this function can be calculated efficiently if we introduce some restrictions on $\mathcal{Y}$ and Φ. For example, in order to calculate the gradient of the weight vector $\mathbf{w}$, one needs to calculate the term

$$E_{P(\mathbf{y}_i|\mathbf{x}_i, \mathbf{w})}[\Phi(\mathbf{x}_i, \mathbf{y}_i)].$$

If we adopt the Markov assumption as in a HMM model, this term can be computed by the standard forward-backward procedure [Rabiner, 1989]. However, such restrictions often make it impossible for CRF to capture long distance relationships. Several works have tried to improve the CRF models by capturing the long distance relationships in the supervised setting *at test time* [Roth and Yih, 2005; Finkel et al., 2005]. In a CRF, the prediction function can be expressed as Eq. (2.1) by rewriting the prediction function as follows:

$$\arg\max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x}_i)} P(\mathbf{y}|\mathbf{x}_i, \mathbf{w}) = \arg\max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x}_i)} \frac{\exp^{\mathbf{w}^T \Phi(\mathbf{x}_i, \mathbf{y}_i)}}{\sum_{\mathbf{y}' \in \mathcal{Y}(\mathbf{x}_i)} \exp^{\mathbf{w}^T \Phi(\mathbf{x}_i, \mathbf{y}')}} = \arg\max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x}_i)} \exp^{\mathbf{w}^T \Phi(\mathbf{x}_i, \mathbf{y}_i)} = \arg\max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x}_i)} \mathbf{w}^T \Phi(\mathbf{x}_i, \mathbf{y}).$$

Max Margin Models: Structural Support Vector Machines The idea of max margin models comes from improving the constraints (2.2), similar to what has been done in [Har-Peled et al., 2002]. The constraints can be improved by adding the notion of margin so that it requires the score of the gold structure to be better than that of the runner up by at least one. Formally, we define $\Delta(\mathbf{y}_i, \mathbf{y})$ to be the binary distance function which is 0 if and only if two structures $\mathbf{y}_i, \mathbf{y}$ are identical and 1 otherwise. In order words, Δ can be considered as a function that measures the binary distance between these two structures. The constraint

(2.2) is modified as follows:

$$\mathbf{w}^T\Phi(\mathbf{x}_i,\mathbf{y}_i) \geq \max_{\mathbf{y}\in\mathcal{Y}(\mathbf{x}_i)} \left[\mathbf{w}^T\Phi(\mathbf{x}_i,\mathbf{y}) + \Delta(\mathbf{y}_i,\mathbf{y})\right]. \tag{2.9}$$

The constraint can consider all possible $\mathbf{y} \in \mathcal{Y}(\mathbf{x}_i)$ including $\mathbf{y}_i$ in the right hand side because of the definition of the distance function. For brevity, we sometimes rewrite the above constraint as

$$\max_{\mathbf{y}\in\mathcal{Y}(\mathbf{x}_i)} \left[\Delta(\mathbf{y}_i,\mathbf{y}) - \mathbf{w}^T\Phi_{\mathbf{y}_i,\mathbf{y}}(\mathbf{x}_i)\right] \leq 0, \tag{2.10}$$

where $\Phi_{\mathbf{y}_i,\mathbf{y}}(\mathbf{x}_i) = \Phi(\mathbf{x}_i,\mathbf{y}_i) - \Phi(\mathbf{x}_i,\mathbf{y})$. Later [Taskar et al., 2004] extends the definition of distance function $\Delta(\mathbf{y}_i,\mathbf{y})$ to reveal more information on the differences of the structures. For example, for the task of part of speech tagging, using Hamming distance seems to be more reasonable because the model can express finer distinction between structures $\mathbf{y}_i$ and $\mathbf{y}$.

It is very important to note that

$$\max_{\mathbf{y}\in\mathcal{Y}(\mathbf{x}_i)} \left[\mathbf{w}^T\Phi(\mathbf{x}_i,\mathbf{y}) + \Delta(\mathbf{y}_i,\mathbf{y})\right], \tag{2.11}$$

is a *different* inference problem from the decoding problem Eq. (2.1). While Eq. (2.1) tries to find the best structure, the goal of Eq. (2.11) is to find structure that is the "worst"[1] when compared to the gold structure. To see this, assume that the weight vector is a zero vector and hamming distance is our distance measurement, the returned structure of Eq. (2.11) is a random structure with maximal hamming distance with respect to $\mathbf{y}_i$, while the Eq (2.1) will return a random structure given that all the weights are zero. Therefore, the structures revealed by these two procedure can be very different. Therefore, for some algorithms we discussed in this chapter, we need to implement two inference procedures: one for (2.1) and the other for (2.11).

The framework which captures this idea is often referred as Max-margin Markov Network [Taskar et al., 2004] or Structural Support Vector Machine (SSVM) [Tsochantaridis et al., 2005]. Training a SSVM entails solving the following global optimization problem:

$$\min_{\mathbf{w}} \frac{\|\mathbf{w}\|^2}{2} + C\sum_{i=1}^{l} L_S(\mathbf{x}_i,\mathbf{y}_i,\mathbf{w}), \tag{2.12}$$

where l is the number of labeled examples and $L_S(\mathbf{x}_i,\mathbf{y}_i,\mathbf{w})$ represents the loss function for the structured

[1]The word "worst" is respect to the violation of the constraint Eq. (2.9).

labeled examples. The function L_S can be written as

$$L_S(\mathbf{x}_i, \mathbf{y}_i, \mathbf{w}) = \ell(\max_{\mathbf{y}} \left[\Delta(\mathbf{y}, \mathbf{y}_i) - \mathbf{w}^T \Phi_{\mathbf{y}_i, \mathbf{y}}(\mathbf{x}_i)\right]),$$

where the function $\ell : \mathbb{R} \to \mathbb{R}$ can be instantiated by many commonly used loss functions such as hinge loss, with $\ell(a) = \max(0, a)$ or squared-hinge loss, with $\ell(a) = \max(0, a)^2$. Throughout this chapter, we write the loss function for SSVM as

$$\ell(a) = \max(0, a)^d,$$

and use the symbol d to indicate the use of the hinge loss function or the square hinge loss function. We refer to the formula as L_1**-loss SSVM** and L_2**-loss SSVM** for $d = 1$ and 2, respectively. The loss function L_S is closely related to the margin constraint expressed by Eq. (2.10) and it essentially measures the violation of the constraints for each example. In the maximal margin model, the prediction function is also Eq. (2.1).

MIRA MIRA could be considered as an approximation online learning algorithm for maximal margin models.

In contrast to the structured perceptron algorithm, the *Margin Infused Relaxed Algorithm (MIRA)*, which was introduced by [Crammer et al., 2005], explicitly uses the notion of margin to learn the weight vector. The algorithm is listed as Algorithm 3. Notice that the only change in this algorithm, compared to the structured perceptron, is in lines 5 and 6. In line 5, MIRA finds the K-best structures using current weight vector. Formally, we define $\Delta(\mathbf{y}_i, \mathbf{y})$ to be the binary distance function which is 0 if and only if two structures $\mathbf{y}_i, \mathbf{y}$ are identical and 1 otherwise. In order words, Δ can be considered as a function that measures the binary distance between these two structures. Then, it updates the weight minimally in order to satisfy the margin constraint (approximately, because it considers only the top K structures instead of all structures.) This quadratic programming (QP) problem (line 6) can be solved using Hildreth's algorithm [Censor and Zenios, 1997] or Platt's SMO algorithm [Platt, 1999].

It is worthwhile to mention the special case when $k = 1$. In this case, we can solve the QP problem analytically. Let $\mathbf{y}^*$ be the structure that maximizes the score with respect to the current weight vector $\mathbf{w}$ (line 5) and suppose $\mathbf{y}^*$ does not equal to the gold label $\mathbf{y}_i$. Then, the new weight vector can be written down in the following way: $\mathbf{w} \leftarrow \mathbf{w} + \eta \Phi_{\mathbf{y}_i, \mathbf{y}^*}(\mathbf{x}_i)$, where

$$\eta = \frac{\Delta(\mathbf{y}_i, \mathbf{y}^*) - \mathbf{w}^T \Phi_{\mathbf{y}_i, \mathbf{y}^*}(\mathbf{x}_i)}{\|\Phi_{\mathbf{y}_i, \mathbf{y}^*}(\mathbf{x}_i)\|^2}.$$

This reduces to the to Passive Aggressive Algorithm [Crammer et al., 2006], which has many similarities to

Algorithm 3 Margin Infused Relaxed Algorithm (MIRA). Note that best$(k, \mathbf{w}, \mathbf{x}_i)$ returns the k-best structures of the current example $\mathbf{x}_i$ according to $\mathbf{w}$.

Require: Number of iteration N, Training Data $\mathcal{S} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{l}$, Number of constraints K

$\mathbf{w} \leftarrow 0, \mathbf{w}_{avg} \leftarrow 0$
for $t = 1 \dots N$ **do**
  for $i = 1 \dots l$ **do**
    $\mathbf{w}_0 \leftarrow \mathbf{w}$
    $\mathcal{H}_k \leftarrow$ best$(k, \mathbf{w}, \mathbf{x}_i)$
    Obtain $\mathbf{w}$ by solving

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w} - \mathbf{w}_0\|^2$$
$$\text{S.T. } \mathbf{w}^T \Phi_{\mathbf{y}_i,\mathbf{y}}(\mathbf{x}_i) \geq \Delta(\mathbf{y}, \mathbf{y}_i), \forall \mathbf{y} \in \mathcal{H}_k$$

    $\mathbf{w}_{avg} \leftarrow \mathbf{w}_{avg} + \mathbf{w}$
  end for
end for
return $\mathbf{w}_{avg}/(Tl)$

the structured perceptron. The key difference, in this specific case, is that the learning rate η for MIRA is "self-tuned" – that is, different examples are scaled differently when the update is made. Note that in the structured perceptron algorithm, unlike the binary case, tuning a fixed learning rate has no effect because it lacks the concept of margin and the learning rate merely scales the function that needs to be maximized while performing inference. Like the structured perceptron, the choice of the number of iterations (N) depends on the domain and the application logic. With MIRA, the prediction function is still Eq. (2.1).

2.3 Optimization Algorithms for Large Margin Structured Output Prediction Models

A significant part of this thesis focuses on large margin structured margin linear models. Specifically, we focus on the objective function in Eq (2.12),

$$\min_{\mathbf{w}} \frac{\|\mathbf{w}\|^2}{2} + C \sum_{i=1}^{l} L_S(\mathbf{x}_i, \mathbf{y}_i, \mathbf{w}),$$

where l is the number of labeled examples and

$$L_S(\mathbf{x}_i, \mathbf{y}_i, \mathbf{w}) = \ell(\max_{\mathbf{y}} \left[\Delta(\mathbf{y}, \mathbf{y}_i) - \mathbf{w}^T \Phi_{\mathbf{y}_i,\mathbf{y}}(\mathbf{x}_i)\right]),$$

where the function $\ell(a) = \max(0, a)^d$. Recall that we refer the model to L_1**-loss SSVM** and L_2**-loss SSVM** when $d = 1$ and 2, respectively.

Fortunately, this is a convex problem, so there are many algorithms that we can apply to optimize this function. In Section 2.2, we mentioned that many learning problems can be casted into structured prediction problems. Interestingly, many techniques that are used to optimize the binary models can also be applied for the structured case.

In this following, we discuss three optimization algorithms for this objective function. Note that we only focus on the linear case without using non-linear kernels, given that recent advances in optimizing linear SVM have led to significant reductions in the training time. For example, for binary classification SVM, the speed of some algorithms [Shalev-Shwartz et al., 2007; Hsieh et al., 2008] is already competitive the speed of online learning algorithms such as perceptron. While we cannot enumerate all possible optimization procedures, we describe the following three that are easy to implement: stochastic gradient descent (SGD) [Kiefer and Wolfowitz, 1952; Bishop, 1996; Le Cun et al., 1998; Zhang, 2004; Bottou, 2004], exponentiated gradient (EG) descent [Kivinen and Warmuth, 1995; Collins et al., 2008] and cutting plane methods (CP) [Joachims et al., 2009; Chang et al., 2010a].

It is important to realize that some of the optimization methods can also be applied to other models. For example, the SGD and EG methods can also be applied to optimizing the CRF model (Eq. (2.8)), but we will mainly focus on solving SSVM. The properties of different optimization algorithms and online learning algorithms will be discussed in Section 2.4.

2.3.1 Stochastic Gradient Descent

The simplest algorithm, in terms of ease of implementation, is probably the stochastic gradient descent (SGD) method, [Kiefer and Wolfowitz, 1952; Bishop, 1996; Le Cun et al., 1998; Zhang, 2004; Bottou, 2004]– a general optimization technique which is widely used across the machine learning community. Broadly, the method proceeds as follows: at each step, it gets a estimation of the gradient direction of the objective function from a subset of labeled examples. The weight vector is then updated by the estimated gradient direction as in standard gradient descent. Compared to gradient methods, SGD updates the weight much more frequently and it has been shown that in practice SGD is faster than the standard gradient descent methods. In this thesis, we focus on the most commonly used version of SGD, where only one example is used to generate the estimated gradient.

To understand the SGD method, we first review the standard gradient descent (GD) method for optimizing a function $f(\mathbf{w})$. The algorithm iteratively goes over two steps:

- Calculate the gradient $\nabla f(\mathbf{w}_k)$ with respect to the current point $\mathbf{w}_k$
- Perform a line search and find a step size $\eta_k \geq 0$ and perform update:

$$\mathbf{w}_{k+1} \leftarrow \mathbf{w}_k - \eta_k \nabla f(\mathbf{w}_k).$$

  A valid step size η_k needs to ensure sufficient decrease in the value of objective function. For unconstrained smooth optimization, Wolfe conditions [Wolfe, 1969] are commonly used to judge if there is sufficient progress.

While GD methods work fine for general optimization problem, calculating gradient for Eq. (2.12) can be expensive because it involves all training examples. Recent advances in on-line learning algorithms [Bottou, 2004; Shalev-Shwartz et al., 2007; Hsieh et al., 2008] have shown that algorithms update the weight vector more frequently are generally faster. The SGD method performs the following two steps iteratively.

- Calculate a noisy gradient $\tilde{\mathbf{g}}_k$, where $E[\tilde{\mathbf{g}}_k] = \nabla f(\mathbf{w}_k)$.
- With a step size $\eta_k \geq 0$

$$\mathbf{w}_{k+1} \leftarrow \mathbf{w}_k - \eta_k \tilde{\mathbf{g}}_k.$$

The SGD algorithm can be extended for optimizing non-smooth convex functions by using subgradient [Boyd and Mutapcic, 2008]. With the right selection of a step size that satisfies the following conditions:

$$\eta_k \geq 0, \qquad \sum_{k=1}^{\infty} \eta_k^2 \leq \infty, \qquad \sum_{k=1}^{\infty} \eta_k = \infty,$$

the SGD procedure converges in expectation [Boyd and Mutapcic, 2008]. A similar proof for smooth functions appeared in some much earlier publications [Kiefer and Wolfowitz, 1952; Bishop, 1996].

In order to generate $\tilde{\mathbf{g}}$, an example $\mathbf{x}_j$ is picked randomly to estimate the gradient direction. The estimate gradient is generated from approximating the loss term generated by the loss term using one example. More precisely, we replace $C \sum_{i=1}^{l} L(\mathbf{x}_i, \mathbf{y}_i, \mathbf{w})$ by $lCL(\mathbf{x}_j, \mathbf{y}_j, \mathbf{w})$ in the objective function Eq. (2.12). The derivative of this new function gives us the estimated gradient direction $\tilde{\mathbf{g}}$.

$$\tilde{\mathbf{g}} = \nabla(\frac{\|\mathbf{w}\|^2}{2} + lCL(\mathbf{x}_j, \mathbf{y}_j, \mathbf{w})). \tag{2.13}$$

This way, $E[\tilde{\mathbf{g}}]$ (over the choice of the example) equals the gradient of the SSVM objective function.

The full SGD Algorithm for Eq. 2.12 is depicted in Algorithm 4. We follow the common implementation

Algorithm 4 SGD for structural SVM. Note that d determines whether if the hinge loss or the square hinge loss are used here. Note that even if the model does not make a mistake, we still perform an update of $\mathbf{w}$ by shirking its size. Unlike structural perceptron, we now are allowed to use a parameter C to balance between regularization and training error.

Require: Number of iteration N, Training Data $\mathcal{S} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{l}$, Balance C, Degree of Loss function d where $\ell(a) = \max(0, a)^d$, step sizes $\{\eta_k\}$

1: $\mathbf{w} \leftarrow 0$
2: **for** $t = 1 \dots N$ **do**
3: **for** $i = 1 \dots l$ **do**
4: $\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x}_i)} \left[\mathbf{w}^T \Phi(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}_i, \mathbf{y})\right]$
5: **if** $(\Delta(\mathbf{y}_i, \hat{\mathbf{y}}) \neq 0)$ **then**
6: **if** $d = 1$ **then**
7: $\gamma = 1$
8: **else if** $d = 2$ **then**
9: $\gamma = 2(\Delta(\mathbf{y}_i, \hat{\mathbf{y}}) - \mathbf{w}^T \Phi(\mathbf{x}_i, \hat{\mathbf{y}}))$
10: **end if**
11: $\tilde{\mathbf{g}} = \mathbf{w} - \gamma l C \Phi_{\mathbf{y}_i, \hat{\mathbf{y}}}(\mathbf{x}_i)$
12: **else**
13: $\tilde{\mathbf{g}} = \mathbf{w}$
14: **end if**
15: $\mathbf{w} \leftarrow \mathbf{w} - \eta \tilde{\mathbf{g}}$
16: update the step size η
17: **end for**
18: **end for**
19: **return** $\mathbf{w}$

scheme, where we do not sample from the training data to create $\tilde{\mathbf{g}}$ but pick the target example in a certain ordering. In line 4, we perform the *loss augmented inference* to find the structure that most violates the constraint in inequality (2.10). Notice that the loss augmented inference solves a different problem from the structure inference problem Eq. (2.1). Lines 5 to 13 calculate the estimated gradient. Finally, the weight vector is then updated in line 15.

While SGD is very easy to implement, several problems exist. First, users still need to specify the step size and this choice may impact the model convergent speed significantly. Second, the stopping criterion is not well defined. Also note that SGD does *not* ensure the decrease of the objective function. Therefore, it is commonly for people to apply the averaging trick here to ensure the quality of the final weight vector.

2.3.2 Exponentiated Gradient Algorithm for Structural SVM

The SGD algorithm solves the primal form of the objective function (Eq. (2.12)). It is also possible to solve the dual form of Eq. (2.12). Let the loss function $\ell(a)$ be the hinge loss function $\max(0, a)$ and let $\alpha_{i,\mathbf{y}}$ denote the dual variable corresponding to the output $\mathbf{y}$ of example x_i. The dual of the SVM objective function,

Algorithm 5 EG for structural SVM.

Require: Training Data $\mathcal{S} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{l}$, Learning rate η, Balance C

1: pick an initial feasible α randomly
2: **for** $t = 1 \dots N$ **do**
3: pick a random example $\mathbf{x}_i$
4: **for** every $\mathbf{y} \in \mathcal{Y}(x_i)$ **do**
5: $\nabla_{i,\mathbf{y}} = \frac{\partial Q(\alpha)}{\partial \alpha_{i,\mathbf{y}}}$
6: **end for**
7: **for** every $\mathbf{y} \in \mathcal{Y}(x_i)$ **do**
8: update $\alpha_{i,\mathbf{y}}$ to $\alpha_{i,\mathbf{y}} e^{-\eta \nabla_{i,\mathbf{y}}}$
9: **end for**
10: normalize αs so that $\sum_{\mathbf{y} \in \mathcal{Y}(\mathbf{x}_i)} \alpha_{i,\mathbf{y}} = C$
11: **end for**
12: **return** $\mathbf{w}(\alpha)$

denoted by $Q(\alpha)$, is as follows:

$$
\begin{aligned}
Q(\alpha): \quad & \min_{\alpha} \frac{1}{2}\|\mathbf{w}(\alpha)\|^2 - \sum_i \sum_{\mathbf{y} \in \mathcal{Y}(\mathbf{x}_i)} \Delta(\mathbf{y}, \mathbf{y}_i)\alpha_{i,\mathbf{y}} \\
\text{S.T.} \quad & \alpha_{i,\mathbf{y}} \geq 0, \forall i, \mathbf{y} \in \mathcal{Y}(\mathbf{x}_i), \\
& \sum_{\mathbf{y} \in \mathcal{Y}(\mathbf{x}_i)} \alpha_{i,\mathbf{y}} = C, \forall i
\end{aligned}
$$

The weight vector $\mathbf{w}(\alpha)$ can be reconstructed from the dual variables as

$$
\mathbf{w}(\alpha) = \sum_i \sum_{\mathbf{y} \in \mathcal{Y}(x_i)} \alpha_{i,\mathbf{y}} \Phi_{\mathbf{y}_i,\mathbf{y}}.
$$

Note that if we divide α by C, then new αs form a simplex.

Exponentiated gradient (EG) algorithm takes advantage of the fact that the constraints on the dual variables can be written down as a simplex set and uses the EG update rule introduced in [Kivinen and Warmuth, 1997] to optimize the dual variable [Collins et al., 2008]. We summarize the EG algorithm in Algorithm 5. For each randomly selected example $\mathbf{x}_i$, the algorithm first computes the gradient of the dual variables corresponding to this example (lines 4 and 5). The new values of the dual variables can be obtained by using the gradient on the exponents and then normalizing the values such that it remains feasible (lines 7-10). In practice, Algorithm 5 is often not feasible, but for certain structure, it is possible to overcome this difficulty through marginalizing the dot product values over parts (the local decisions of a structure). For more details, please refer to [Collins et al., 2008].

2.3.3 Cutting Plane Methods

The cutting plane (CP) methods use a different idea to solve the Structural SVM problem [Tsochantaridis et al., 2005]. Note that we can rewrite the SSVM objective

$$\min_{\mathbf{w}} \quad \frac{\|\mathbf{w}\|^2}{2} + C\sum_{i=1}^{l}\xi_i^d \tag{2.14}$$

$$\text{S.T.} \quad \forall i, \mathbf{y} \in \mathcal{Y}(\mathbf{x}_i), \Delta(\mathbf{y}, \mathbf{y}_i) - \mathbf{w}^T\Phi_{\mathbf{y}_i,\mathbf{y}}(\mathbf{x}_i) \leq \xi_i, \tag{2.15}$$

$$\forall i, \xi_i \geq 0$$

where d can be 1 or 2, indicating whether ℓ is the hinge loss or the square hinge loss function. While it seems that we have only two constraints for each example $\mathbf{x}_i$, in fact (2.15) implicitly carries $|\mathcal{Y}(\mathbf{x}_i)|$ number of linear inequalities (one for each structure $\mathbf{y}$). However, at the optimal solution, only a few of inequalities will be "active" (that is, an inequality becomes an equality). Intuitively, these are the difficult structures given that if the weight vector classified them correctly, Eq. (2.15) will be satisfied.

The cutting plane methods try to keep a set of "difficult" structures $\mathcal{W}_i$ (we called it a working set) for each example $\mathbf{x}_i$. Usually the size of the working set $\mathcal{W}_i$ is extremely small compared to $\mathcal{Y}(\mathbf{x}_i)$. Therefore, it means that obtaining the weight vector with respect to the current working sets is a lot easier than solving the original structural SVM problem (2.12). We can write down the objective function respect to the current working set:

$$\min_{\mathbf{w}} \quad \frac{\|\mathbf{w}\|^2}{2} + C\sum_{i=1}^{l}\xi_i^d \tag{2.16}$$

$$\text{S.T.} \quad \forall i, \mathbf{y} \in \mathcal{W}_i, \Delta(\mathbf{y}, \mathbf{y}_i) - \mathbf{w}^T\Phi_{\mathbf{y}_i,\mathbf{y}}(\mathbf{x}_i) \leq \xi_i,$$

$$\forall i, \xi_i \geq 0$$

where d is 1 or 2 for the hinge loss and square hinge loss functions, respectively.

The cutting plane algorithm (Algorithm 6) updates the working set in the following way:

- For an example $\mathbf{x}_i$, find structure $\hat{\mathbf{y}}_i$ that violates the constraint (2.15) the most, using the current weight vector $\mathbf{w}$ (line 5).
- Add $\hat{\mathbf{y}}_i$ into the working set $\mathcal{W}_i$ (line 7).
- Obtain a new $\mathbf{w}$ with respect to the current working set by solving Eq. (2.16) (line 8) in Algorithm 6.

The duality gap parameter ϵ controls the precision of the final solution. Note that the CP strategy still needs

Algorithm 6 Cutting Plane Methods. Note that the duality gap parameter ϵ controls the precision of the final solution.

Require: Training Data $\mathcal{S} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{l}$, Duality gap ϵ, Balance C

$\mathbf{w} \leftarrow 0$
$\mathcal{W}_i \leftarrow \emptyset, \forall i = 1 \dots l$
repeat
  for $i = 1 \dots l$ **do**
    $\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x}_i)} \left[\mathbf{w}^T \Phi(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}_i, \mathbf{y})\right]$
    if $\Delta(\mathbf{y}_i, \hat{\mathbf{y}}) - \mathbf{w}^T \Phi_{\mathbf{y}_i, \hat{\mathbf{y}}}(\mathbf{x}_i) > \xi_i + \epsilon$ **then**
      $\mathcal{W}_i \leftarrow \mathcal{W}_i \cup \{\hat{\mathbf{y}}\}$
      update $\mathbf{w}$ by solving Eq. (2.16)
    end if
  end for
until no new element is added to any $\mathcal{W}_i$
return $\mathbf{w}$

a solver to solve the subproblem (Eq. (2.16)). For example, one can apply SGD to solve the subproblem (Eq. (2.16)). In [Joachims et al., 2009], they apply a custom-designed solver based on SVM-light to solve the sub-problem. We will refer to the approach used in [Joachims et al., 2009] as CP+QP.

2.4 Summary of Discriminative Structured Learning Algorithms: A User Point of View

In this section, we first summarize the learning algorithms for structured models that are commonly used in the NLP community. At the end of this section, we justify the choice of the optimization algorithm procedure used in this thesis.

In Table 2.2, we summarize the different models and their corresponding optimization algorithms. Note that we call the procedure used in [Joachims et al., 2009] CP+QP and the procedure proposed in this thesis CP+DCD (more details will be mentioned later). We would like to compare these algorithms from a user point of view. For example, the user might want to know if he needs to adjust the step size by himself and what inference procedures he needs to implement.

Procedures required in the training phase We first review three different inference procedures we have mentioned.

- Argmax Inference (Argmax): The inference procedure used in the decoding phase. (Eq. (2.1))
- Loss-Augmented Inference (LArgmax): The inference procedure used in the margin constraint (Eq. (2.11)).

Model	Objective	(Optimization) Algorithms	Self-Adjusted Step size?	Inference in Training	Inference in Testing
CRF	Eq. (2.8)	SGD	N	Marginal	Argmax
		BFGS	Y	Marginal	Argmax
		EG	N	Marginal	Argmax
L_1**-loss SSVM**	Eq. (2.14)	SGD	N	LArgmax	Argmax
		CP+QP	Y	LArgmax	Argmax
		EG	N	Marginal	Argmax
L_2**-loss SSVM**	Eq. (2.14)	SGD	N	LArgmax	Argmax
		CP+DCD (**Contribution**)	Y	LArgmax	Argmax
SP	None	Algorithm 2	N	Argmax	Argmax
MIRA	None	Algorithm 3	N	Argmax	Argmax

Table 2.2: Summary of all algorithms for discriminative structured learning algorithms. Note that we use CP to represents general algorithms which use cutting plane procedures and CP+DCD to represent the procedure proposed in this thesis, where we use dual coordinate descent methods to optimize the subproblem in the CP procedure. See the text for more details.

- Marginal Inference (Marginal): The inference procedure that calculates the summation of certain values on all possible structures. For example, in a linear chain CRF, we use forward and backward algorithm to calculate the expectation of the feature functions.

While all learning algorithms use the inference problem (Eq. (2.1)) to make prediction results in the testing phase, only SP and MIRA use Eq. (2.1) in the training phase. Therefore, a user only needs to implement one inference procedure for SP and MIRA, and this probably is the reason why SP and MIRA are very popular in the NLP community. Both SGD and CP (including CP+QP and CP+DCD) run the Loss-Augmented Inference in the training phase, but make decisions differently. It is interesting to notice that the EG, like CRF, requires to calculate the marginal values of all structures; especially, in particular, EG can be used to solve the structured SVM objective function.

Online Learning algorithm and SGD In the algorithms we mentioned, only structured perceptron and MIRA are mistake-driven algorithms. It means that they do not update their weight vector if the current weight vector does not make a mistake for a given example. On the other hand, SGD for SSVM updates its weight vector for every example encountered. Batch algorithms like CP are not mistake-driven algorithms as well.

The structured perceptron (SP) algorithm *seems* to be very similar to a stochastic gradient descent algorithm if the regularization term is omitted in the objective in L_1**-loss SSVM**. More specifically, assume that the objective function for SGD is as follows

$$\sum_{i=1}^{l} \left(\max_{\mathbf{y}} \left[\mathbf{w}^T \Phi(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) \right] - \mathbf{w}^T \Phi(\mathbf{x}_i, \mathbf{y}_i) \right),$$

we refer to the SGD algorithm for this objective function as SGD-Loss-Only. However, there are several important differences between the SGD-Loss-Only and SP. First, the step size for SGD-Loss-Only may be

different for every example and for different $\mathbf{w}$, while the step size for SP is fixed. Second, note that SGD-Loss-Only and SP solve *different* inference problems in their training algorithms. SP solves the structure prediction problem during training, while SGD performs loss augmented inference. This means that the user should implement two (very related) inference algorithms for SGD. Therefore, SGD-Loss-Only and SP are still different even when there is no regularization term. In this sense, the SGD is not "mistake" driven, given that it did not use the actual inference procedure used in the testing time.

Adjusting the step size We use the cutting plane optimization procedure in this thesis with significant modifications. In [Joachims et al., 2009], the authors use the hinge loss function ($d = 1$), and solve Eq. (2.16) with custom design QP solver. In the Chapter 5 and 6, we first point out and realize that there are many advantages of choosing the square hinge loss function ($d = 2$), given that the optimization procedure for Eq. (2.16) can be easily implemented by the dual coordinate descent methods (DCD) [Hsieh et al., 2008], which does not require to tune the step size η and is very easy to implement. We refer to our algorithm as (CP+DCD). The full optimization details are in Section 6.2. To the best our knowledge, this is the first work which apples CP+DCD to the structured output prediction problems.

Tuning the step size (e.g. the learning rate) is necessary for SP, SGD and EG, but it is not necessary for MIRA and CP+DCD. In the CP method, we need an optimization algorithm to solve the subproblem (Eq. (2.16)). If we use the SGD algorithm, adjusting the step size would become necessary. In CP+DCD, the algorithm can find the optimal step size by itself.

The ability of choosing the step size automatically is one of the major reason that we develop and use CP+DCD. In addition, CP based method only require solving loss-augmented inference and has better control of the stopping condition. Hence, we believe that the CP+DCD method is an ideal optimization method for this thesis. Finally, the CP based methods have a natural way to parallelize the inference procedure, which is very important for the speed concern.

Stopping Condition Another advantage of CP over SGD and EG is the stopping condition. The usual stopping condition for SGD and EG is based on the number of iterations, which can be difficult to tune. Following [Joachims et al., 2009], our stopping condition for SSVM is the duality gap, which is more stable and easier to control compared to the number of iterations.

2.5 Unsupervised and Semi-Supervised Structured Output Prediction Models

Structured output prediction models are designed to capture the interdependence between assignments to individual variables. Unfortunately, obtaining labeled examples for structures is often difficult, given that many decisions need to be made for one input example. Hence, several approaches have been proposed to address the issue of lack of supervision.

2.5.1 Expectation Maximization and Self-training Algorithms

Probably one of the oldest algorithm that is used to address the issue of lacking supervision is the Expectation Maximization (EM) algorithms [Dempster et al., 1977]. EM algorithms are mainly designed for generative models, where the objective function aims to maximize the log-likelihood data by marginalizing all of the unlabeled latent variables. More formally, consider the following unlabeled dataset $\mathcal{U} = \{\mathbf{x}_i\}$. Let us represent the model parameters as θ. The EM algorithm maximizes the following objective function

$$\mathcal{L}(\theta) = \sum_{i=1}^{|\mathcal{U}|} \log P(\mathbf{x}_i|\theta) = \sum_{i=1}^{|\mathcal{U}|} \log \left(\sum_{\mathbf{h} \in \mathcal{H}(\mathbf{x}_i)} P(\mathbf{x}_i, \mathbf{h}|\theta) \right). \tag{2.17}$$

This is a non-convex objective function.

The Standard (Original, Soft) EM algorithm The EM algorithm can be considered as an optimization procedure for this objective function. The EM algorithm iteratively goes over two steps:

- "E-step": Estimate the probability distribution $P(\mathbf{h}|\mathbf{x}_i, \theta^{t-1})$ using the parameters from previous iteration θ^{t-1}.
- "M-step": Find the next parameters θ^t by solving the following objective function

$$\sum_{i=1}^{|\mathcal{U}|} \sum_{\mathbf{h} \in \mathcal{H}(\mathbf{x}_i)} q(\mathbf{x}_i, \mathbf{h}) \log(P(\mathbf{x}_i, \mathbf{h}|\theta)),$$

  where $q(\mathbf{x}_i, \mathbf{h}) = P(\mathbf{h}|\mathbf{x}_i, \theta^{t-1})$.

The EM procedure is guarantee to converge to the local maximal point of the original objective function. We can apply EM algorithm to generative models that are designed for structured output prediction tasks. Unlike the supervised learning approaches we mentioned above, the EM procedure take advantages of the

unlabeled data to improved the model. Unfortunately, previous work has shown that the straight forward applications of EM procedure can drift away from the right model even with a good starting point because of lack of guidance [Merialdo, 1994].

The Hard (Truncated, Viterbi) EM algorithm In the EM algorithm, the "E-step" of the EM algorithm needs to estimate the full posterior distribution $P(\mathbf{h}|\mathbf{x}_i, \theta^{t-1})$, which can be expensive to compute in certain scenarios. Therefore, in practice, a commonly used variation of the EM algorithm called *the hard (truncated) EM* algorithm has been proposed. The algorithm was first proposed by [Neal and Hinton, 1998], where they refer to this algorithm as "sparse" EM. Instead of calculating full posterior distribution, the hard EM algorithm only finds a single best structure according to the current model. The hard EM works as follows:

- "E-step": Find the best assignment $\hat{\mathbf{h}}_i = \arg\max_{\mathbf{h}} P(\mathbf{h}|\mathbf{x}_i, \theta^{t-1})$ using the parameters from previous iteration θ^{t-1}.
- "M-step": Find the next parameters θ^t by solving the following objective function

$$\sum_{i=1}^{|\mathcal{U}|} \sum_{\mathbf{h} \in \mathcal{H}(\mathbf{x}_i)} \log(P(\mathbf{x}_i, \hat{\mathbf{h}}_i|\theta)).$$

The EM algorithm is in fact very important in this thesis. In Chapter 3, our algorithm can be considered as an extension of the hard EM algorithm. In Chapter 4, we put various EM frameworks under the same framework. For more discussions and comparisons between EM and hard EM, please refer to Chapter 4.

Self-training Algorithms While EM is widely used in the community, it is mainly designed for probabilistic generative models. For general discriminative/generative models, which do not need to be probabilistic models, *self-training* is a commonly used algorithm for training semi-supervised models in the natural language community [McClosky et al., 2006; Daumé III, 2008]. The self-training algorithm can be decompose into two steps:

- "Prediction step": Find the best assignment $\hat{\mathbf{h}}_i = \arg\max_{\mathbf{h}} \mathbf{w}^T \Phi(\mathbf{x}_i, \hat{\mathbf{h}})$ using the weight vector from previous iteration $\mathbf{w}$.
- "Retraining step": Find the next weight vector $\mathbf{w}$ by training the model on the pseudo labeled data $\{(\mathbf{x}_i, \hat{\mathbf{h}}_i)\}$.

The self-training algorithm is indeed closely related to the hard-EM algorithm. When the model is a generative model, the self-training algorithm reduce to the hard EM algorithm.

2.5.2 Approaches Related to Domain Specific Knowledge

Due to the problem of lack of guidance in EM, some works have tried to inject knowledge into EM algorithms. One is using the prior knowledge to accurately tailor the generative model so that it better captures the domain structure. For example, [Grenager et al., 2005] propose *Diagonal Transition Models* for sequential labeling tasks where neighboring words tend to have the same labels. This is done by constraining the HMM transition matrix, which can be done also for other models, such as CRF. However [Roth and Yih, 2005] showed that reasoning with more expressive, non-sequential constraints can improve the performance for the supervised protocol.

A second approach has been to use a small high-accuracy set of labeled tokens as a way to seed and bootstrap the semi-supervised learning. This was used, for example, by [Collins and Singer, 1999] in information extraction. [Haghighi and Klein, 2006] extends the dictionary-based approach to sequential labeling tasks by propagating the information given in the seeds with contextual word similarity. This follows a conceptually similar approach by [Cohen and Sarawagi, 2004] that uses a large named-entity dictionary, where the similarity between the candidate named-entity and its matching prototype in the dictionary is encoded as a feature in a supervised classifier.

2.5.3 Semi-supervised Structural SVM

There has been several large margin frameworks [Zien et al., 2007; Brefeld and Scheffer, 2006] that try to address the issue of lacking supervision for structural prediction tasks. [Brefeld and Scheffer, 2006] extend the idea of co-training [Blum and Mitchell, 1998] to structural prediction tasks. They proposed a joint objective function that learns over labeled data, maximizing the agreement between two views of the unlabeled data. [Zien et al., 2007] use the idea of transductive learning and apply it to structural prediction tasks.

2.5.4 Latent Structural SVM

Latent Structural SVM [Yu and Joachims, 2009] is a framework that allows using latent structures in the structural SVM framework (Eq. (2.12)). While Latent Structural SVM is not designed for semi-supervised and supervised learning, it is still worth mentioning here because its relationship to our proposed frameworks. The goal of Latent Structural SVM is to use latent structures $\mathbf{h}$ as an intermediate step to improve the target task (predicting $\mathbf{y}$) performance.

In the Latent Structural SVM, the feature vector is defined over both the latent structure $\mathbf{h}$ and the output

structure $\mathbf{y}$. Its objective function can be written down as follows:

$$\min_{\mathbf{w}} \frac{\|\mathbf{w}\|^2}{2} + C \sum_i \left(\max_{\mathbf{y},\mathbf{h}} \left(\Delta(\mathbf{y}_i, \mathbf{y}, \mathbf{h}) + \mathbf{w}^T \Phi(\mathbf{x}_i, \mathbf{y}, \mathbf{h}) \right) - \max_{\mathbf{h}} \left(\mathbf{w}^T \Phi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}) \right) \right), \tag{2.18}$$

where $\mathbf{y}$ represents the *output* structures and $\mathbf{h}$ represents the *latent* structures. Unlike Structural SVM, the objective function for Latent Structural SVM is non-convex. For more details please refer to [Yu and Joachims, 2009]. For more discussions on Latent Structural SVM and our proposed frameworks, please see Section 6.14.

Chapter 3

Constraints as Indirect Supervision

The goal of this thesis is to design structural learning frameworks that can take advantage of high level human knowledge. We first address "structured output" tasks by incorporating prior human knowledge as *constraints* [Chang et al., 2007, 2008a,b].

Natural Language Processing (NLP) systems typically require large amounts of labeled data to achieve good performance. Acquiring labeled data is a difficult and expensive task. Therefore, increasing attention has been recently given to semi-supervised learning, where large amounts of unlabeled data are used to improve the models learned from a small training set. The hope is that semi-supervised or even unsupervised approaches, when given enough knowledge about the *structure* of the problem, will be competitive with supervised models trained on large training sets. However, in the general case, semi-supervised approaches give mixed results, and sometimes even degrade the model performance [Nigam et al., 2000]. In many cases, improving semi-supervised models is done by *seeding* these models with domain information taken from dictionaries [Yarowsky, 1995; Collins and Singer, 1999; Haghighi and Klein, 2006].

In many problems, dependencies among output variables have non-local nature, and incorporating them into the model as if they were probabilistic phenomena can undo a great deal of the benefit gained by factorization, as well as making the model more difficult to design and understand. For example, consider an information extraction task where two particular types of entities cannot appear together in the same document. Modeling mutual exclusion in the scenario where n random variables can be assigned mutually exclusive values introduces n^2 pairwise edges in the graphical model, with obvious impact on training and inference. Obviously, this is very expensive given that a lot of parameters are being wasted in order to learn something the model designer already knows. For example, in order to capture such constraints by higher order HMMs or CRFs, we need to build a T-order model which can consider all connections if there are T tokens in $\mathbf{x}$. This requires a significant increase in the number of parameters even though we actually know that all the weights on the links between y_i and y_j should be $-\infty$ if y_i equals to y_j. In short, HMMs and CRFs do not have a way to encode the knowledge *directly* but only *indirectly*, by adding more features or increasing the order of the models [Roth and Yih, 2005]. However, the inference problems in high-order

models are very expensive and learning a more complex model requires more labeled examples to have a good performance. Therefore, high order models will have a huge disadvantage when the number of examples is limited. In short, non-local and first-order relationships can be very difficult to model using only *local* features and might require a lot of training examples to achieve good results.

In this chapter, we address the need of having a general framework that allows to encode the knowledge *directly* and develop a general learning framework to address this issue. The contributions of the chapter are as follows:

1. We propose the **Constrained Conditional Model (CCM)**, which provides a *direct* way to inject prior knowledge into a statistical model, in the form of **constraints**.

   One advantage of CCMs is that it allows combining simple models with *declarative and expressive* constraints. This is an effective approach to making probabilistic models expressive. Therefore, CCMs can be considered as a nice *interface* for incorporating knowledge into off-the-shelf statistical models without designing a task-specific model. Note that adding constraints to CCMs does not enlarge the feature space but rather augments the simple linear models. Along with appropriate training approaches that we discuss later, we need to learn simpler model than standard high order probabilistic models but can still make decisions with expressive models. Since within CCMs we combine declarative constraints, possibly written as first order logic expressions [Rizzolo and Roth, 2007], with learned probabilistic models, we can treat CCMs as a way to combine or bridge logical expressions and learning statistical models. We also discuss how to solve inference problems with expressive constraints efficiently in Section 3.1.2.

2. Based on the principle introduced by CCMs, we introduce **HMM$^{\text{CCM}}$**, a constraint-infused Hidden Markov Model. We demonstrate how to train and test HMM$^{\text{CCM}}$ in a principled way and show that adding little knowledge can improve the model significantly.

   Note that by modeling the constraints directly, the inference problem in Eq. (2.1), becomes harder to solve, compared to the one used by low order HMMs/CRFs. As we show later, such a sacrifice is usually very rewarding in terms of final performance. In this chapter, we use soft constraints rather than hard constraints in the constraint driven learning framework. Our definition of the soft constraints allows us to use beam search to solve the inference problem approximately. Moreover, by treating constraints and models separately, the constraints do not add any overhead to our *learning* algorithm of HMM$^{\text{CCM}}$.

3. We show that prior knowledge plays a crucial role when the amount of labeled data is limited. We empirically show that incorporating high-level knowledge via CCMs significantly improves the results of both *supervised learning* and *semi-supervised learning*.

   Note that semi-supervised learning results are especially interesting, since we can consider constraints as a supervision resource to guide the semi-supervised learning procedure.

Our generalized procedure for semi-supervised learning is called COnstraint-Driven Learning **(CODL)** [Chang et al., 2007]. As is often the case in semi-supervised learning, the algorithm can be viewed as a process that improves the model by generating feedback through *labeling* unlabeled examples. Our algorithm pushes this intuition further, in that the use of constraints allows us to better exploit domain information as a way to label, along with the current learned model, unlabeled examples. Given a small amount of labeled data and a large unlabeled pool, our framework initializes the model with the labeled data and then repeatedly, as shown in Figure 3.1. This way, we can generate *better* "training" examples during the

Generalized COnstraint-Driven Learning

1. Uses *constraints* and the learned model to label the instances in the pool
2. Updates the model using newly labeled data/distributions

Figure 3.1: Generalized constrained driven learning algorithm

semi-supervised learning process. While not only is the procedure is intuitively appealing, it can be justified as an optimization procedure for an objective function. Note this general algorithm can be considered as an extension to the hard EM algorithm and the self-training algorithm mentioned in Section 2.5.1.

This chapter formally defines CCMs so that it is easier to apply constraints to statistical models in both supervised and semi-supervised settings. Moreover, we provide a principled justification for the algorithms proposed in [Chang et al., 2007] (with modifications) and obtain better empirical results. Finally, this chapter includes a wide set of experiments that show the properties of the HMM$^{\text{CCM}}$ algorithm and compares it to other algorithms.

Note that we are not the first to point out the importance of long distance relationships and other approximated supervised training algorithms have been proposed (See Section 3.5 for more details). However, we want to stress that in CCM, the notion of constraint is different and much more general. For example, the CCM framework offers the possibility to separate models (features) and constraints. Therefore, it is possible to apply constraints to a trained model *directly* without re-training the model. Moreover,

such separation is the key of the success of our semi-supervised learning algorithm, which uses constraints as a form of supervision. We clarify this point later in text.

The rest of the chapter is organized as follows: Section 3.1 formally defines the constrained conditional models. We introduce an instance of CCMs based on a Hidden Markov Model in Section 3.2. In Section 3.3 we introduce the tasks and the data on which the algorithms will be tested. The experimental results are presented in Section 3.4. We discuss related work in Section 3.5 and summarize this chapter in Section 3.6.

3.1 Constrained Conditional Model

CCMs target structured prediction problems, where given a point $\mathbf{x}$ in an input space $\mathcal{X}$, the goal is to find a label assignment $\mathbf{y}$ in the set of all possible output structures for $\mathbf{x}$, $\mathcal{Y}(\mathbf{x})$. For example, in part-of-speech (POS) tagging, $\mathcal{Y}(\mathbf{x})$ is the set of all possible POS tags for a given input sentence $\mathbf{x}$.

Given a set of feature functions $\Phi = \{\phi_i(\cdot)\}_{i=1}^n$, $\phi_i : \mathcal{X} \times \mathcal{Y} \rightarrow R$, which typically encode the *local* properties of a pair $(\mathbf{x}, \mathbf{y})$ (often, the image of ϕ_i is $\{0, 1\}$), the "score" of a structure $\mathbf{y}$ of a linear model can be represented as

$$f(\mathbf{x}, \mathbf{y}) = \mathbf{w}^T \Phi(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} w_i \phi_i(\mathbf{x}, \mathbf{y}).$$

The prediction function of this linear model is $\arg\max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} f(\mathbf{x}, \mathbf{y})$.

Constrained Conditional Models provide a general **interface** that allows users to easily combine domain knowledge (which is provided by humans) and statistical models (which are learned from the data). In this chapter, we represent domain knowledge as a (usually small) set of constraints $\Psi = \{\Psi_k\}_{k=1}^m$. For each constraint, we are also provided a function $d_{\Psi_k} : \mathcal{X} \times \mathcal{Y} \rightarrow R$ that measures the degree to which the constraint Ψ_k is violated in a pair $(\mathbf{x}, \mathbf{y})$. While there are different ways to estimate d_{Ψ_k}, in this chapter, we define the "violation function" as follows. Let

$$\mathbf{y}_{[1\ldots i]} = (y^1, y^2, \ldots, y^i).$$

be a partial assignment of $\mathbf{y}$. Then

$$d_{\Psi_k}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{T} \hat{\Psi}_k(\mathbf{x}; \mathbf{y}_{[1\ldots i]}), \tag{3.1}$$

where $\hat{\Psi}_k(\mathbf{x}; \mathbf{y}_{[1\ldots i]})$ is a binary function which indicates whether y_i violates the constraint Ψ_k with respect to a partial assignment $\mathbf{y}_{[1\ldots i-1]}$. Note that for some constraints, the violation cannot be calculated with

partial assignments. In these cases, $\hat{\Psi}_k$ will return 0 to indicate the constraints i is not violated according to the current partial assignment.

A **Constrained Conditional Model** can be represented using two weight vectors: the feature weight vector $\mathbf{w}$ and the constraint penalty vector ρ. The score of an assignment $\mathbf{y} \in \mathcal{Y}$ for an instance $\mathbf{x} \in \mathcal{X}$ can then be obtained by[1]

$$f_{\Phi,\Psi}(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{n} w_i \phi_i(\mathbf{x},\mathbf{y}) - \sum_{k=1}^{m} \rho_k d_{\Psi_k}(\mathbf{x},\mathbf{y}). \tag{3.2}$$

A CCM then selects the best structure using the inference problem

$$\mathbf{y}^* = \arg\max_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} f_{\Phi,\Psi}(\mathbf{x},\mathbf{y}), \tag{3.3}$$

as its prediction.

Note that Eq. (3.2) allows using both "hard constraints" (constraints that should not be violated) and "soft constraints" (constraints that can occasionally be violated). Assume that the constraint set can be partitioned into a soft constraint set $\mathcal{S}$ and a hard constraint set $\mathcal{H}$ ($\mathcal{H} \cap \mathcal{S} = \emptyset$ and $\mathcal{H} \cup \mathcal{S} = \ominus$). The set of "feasible" structures for a given input $\mathbf{x}$ is then reduced to

$$\bar{\mathcal{Y}}(\mathbf{x}) = \{\mathbf{v} \mid \mathbf{v} \in \mathcal{Y}(\mathbf{x}), \Psi_k(\mathbf{x},\mathbf{v}) = 0, \forall \Psi_k \in \mathcal{H}\}$$

Eq. (3.3) can be rewritten as

$$\arg\max_{\mathbf{y}\in\bar{\mathcal{Y}}(\mathbf{x})} \sum_{i=1}^{n} w_i \phi_i(\mathbf{x},\mathbf{y}) - \sum_{k:\Psi_k\in S} \rho_k d_{\Psi_k}(\mathbf{x},\mathbf{y}). \tag{3.4}$$

Note that a CCM is not restricted to be trained with any particular learning algorithm. The key goal of a CCM is to allow combining constraints and models in the testing phase. Similarly to other linear models, specialized algorithms may need to be developed to train CCMs. Notice also that the left component in Eq. (3.2) may stand for multiple linear models, trained separately. Unlike standard linear models, we assume the availability of some prior knowledge, encoded in the form of *constraints*. When there is no prior knowledge, there is no difference between CCMs and other linear models.

[1] Recall that n is the number of features and is typically very large, and m is the number of constraints, typically small.

3.1.1 Benefits of Separating Constraints and Features

In Eq. (3.2), the constraints term (the second term) appears to be similar to the features term (the first time). In fact, using constraints or features to express long distance relationships sometimes can be a design choice. However, it is important to note that both in this work and in many other recent publications [Roth and Yih, 2004, 2005; Chang et al., 2007; Graca et al., 2007; Bellare et al., 2009; Carlson et al., 2010; Ganchev et al., 2010], people have demonstrated the importance of separating features and constraints. In this section we discuss this issue in details.

- Hard constraints vs. features

  While we simplified our notation in Eq. (3.2), the constraint term is different from the feature term because it can be used to enforce hard constraints. Hence, it is necessary to separate constraints and features.

- Reuse and improving existing models with expressive constraints

  It is often expensive to retrain a complicated NLP system. While sometime choosing features or constraints to express long distance relationships can be a design choice, adding more features often require expensive retraining. Moreover, in [Roth and Yih, 2004], they propose use constraints to *combine* two independent trained models. Note that if we model the long distance constraints with features, we need to train these two models *jointly*, which can be much more expensive compared to training them separately by separating constraints from the features.

- Implications on learning algorithms

  Separating expressive constraints from models also impacts the learning performance. Many recent works have show many benefits of keeping the existing model and treating the expressive constraints as a supervision resource [Chang et al., 2007; Graca et al., 2007; Bellare et al., 2009; Carlson et al., 2010; Ganchev et al., 2010]. As we show in this work, using constraints as a supervision resource can be very effective when there are few labeled examples in the semi-supervised setting.

  In the supervised setting, we separate the constraints from features in Eq. (3.2) because the constraints should be trusted most of the time. Therefore, the penalties ρ can be fixed or handled separately. For example, if we are confident about our knowledge, rather than learning the $\{\rho_j\}$, we can directly set them to ∞, thus enforcing the chosen assignment $\mathbf{y}$ to satisfy the constraints. There issues are discussed in details later in the chapter.

- Efficiency

Another difference between ρ and $\mathbf{w}$ is that ρ should always be positive. The reason is that $d_{\Psi_i}(\mathbf{x}, \mathbf{y}) \geq 0$ and the assignments that violate the constraints should be punished (See Eq. (3.2)). This allows us to design an admissible heuristic and speed up exact inference using A^* search. We cannot have this nice result when we treat constraints as features. This is of particular importance, since the constraints could be non-local, therefore efficient dynamic programming algorithms are not applicable.

There are several advantages of using constraints. First, constraints provide a platform for encoding prior knowledge, possibly expressed as high level predicates. As we will show later, this is especially important when the number of labeled instances is small. Second, constraints can be more expressive than features used by the *existing* model so adding constraints can sometimes prevent us to redesign the model. Instead of building a model from complex features, CCMs provide a way to combine "simple" learned models with a small set of "expressive" constraints. Importantly, combining simple models with constraints often results in better performance. For example, the top-ranking system in the CoNLL 2005 shared task uses a CCM approach and outperforms many systems built using complex models [Punyakanok et al., 2005a].

3.1.2 Inference with Constraints

Adding expressive constraints comes with a price – the dynamic programming inference algorithms typically used in off-the-shelf statistical models can no longer be applied. In this section, we discuss three different types of inference algorithms that allow solving the inference problem in Eq. (3.3) with expressive constraints.

Integer Linear Programming

In the earlier related works that made use of constraints, the constraints were assumed to be binary functions; in most cases, a high level (first order logic) description of the constraints was compiled into a set of linear inequalities, and exact inference was done using an integer linear programming formulation (ILP) [Roth and Yih, 2004, 2007, 2005; Punyakanok et al., 2005a; Barzilay and Lapata, 2006; Clarke and Lapata, 2006]. Although ILP can be intractable for very large-scale problems, it has been shown to be quite successful in practice when applied to many practical NLP tasks [Roth and Yih, 2007].

A^* Search

Recall that the inference problem for CCMs is as follows (a copy of Eq.(3.3)):

$$\max_{\mathbf{y}} f_{\Phi,\Psi}(\mathbf{x},\mathbf{y}) = \max_{\mathbf{y}} \mathbf{w}^T\Phi(\mathbf{x},\mathbf{y}) - \sum_{k=1}^{m} \rho_k d_{\Psi_k(\mathbf{x},\mathbf{y})}$$

Assume that there exists an efficient dynamic programming algorithm to compute $\arg\max \mathbf{w}^T\Phi(\mathbf{x},\mathbf{y})$ without considering the constraints[2]. This implies that if we ignore the constraints, given a partial label assignment $\mathbf{y}_{[1\ldots i]}$, we can efficiently complete the label assignment $\mathbf{y}_{[(i+1)\ldots T]}$ without considering the constraint penalty, where T represents the total number of "parts" of the output structure. That is, we can solve the following optimization problem efficiently and exactly:

$$h(\mathbf{x},\mathbf{y}_{[1\ldots i]}) = \max_{\mathbf{y}_{[(i+1)\ldots T]}} \mathbf{w}^T\Phi(\mathbf{x},\mathbf{y}_{[(i+1)\ldots T]} \mid \mathbf{y}_{[1\ldots i]}) \tag{3.5}$$

Note that in the above equation, $\mathbf{y}_{[1\ldots i]}$ is fixed and we search over the rest of an assignment $\mathbf{y}_{[(i+1)\ldots T]}$ to complete $\mathbf{y} = \mathbf{y}_{[1\ldots i]} \cdot \mathbf{y}_{[(i+1)\ldots T]}$. The value $\mathbf{w}^T\Phi(\mathbf{x},\mathbf{y}_{[(i+1)\ldots T]} \mid \mathbf{y}_{[1\ldots i]})$ is the partial score for the $\mathbf{y}_{[(i+1)\ldots T]}$ with the given prefix and hence,

$$\mathbf{w}^T\Phi(\mathbf{x},\mathbf{y}_{[1\ldots i]} \cdot \mathbf{y}_{[(i+1)\ldots T]}) = \mathbf{w}^T\Phi(\mathbf{x},\mathbf{y}_{[1\ldots i]}) + \mathbf{w}^T\Phi(\mathbf{x},\mathbf{y}_{[(i+1)\ldots T]} \mid \mathbf{y}_{[1\ldots i]})$$

We can perform this factorization because of the assumption that the feature function can be decomposed.

We also define g as the function that returns the score (with constraint penalties) of the current partial assignment $\mathbf{y}_{[1\ldots i]}$:

$$g(\mathbf{x},\mathbf{y}_{[1\ldots i]}) = \mathbf{w}^T\Phi(\mathbf{x},\mathbf{y}_{[1\ldots i]}) - \sum_{j=1}^{m} \rho_j d_{\Psi_j}(\mathbf{x},\mathbf{y}_{[1\ldots i]}) \tag{3.6}$$

Next, we show that using g and h, the A^*algorithm can always return the optimal solution of the CCM inference problem.

Theorem 1. *Assume that $\rho_k \geq 0$ for $k = 1\ldots m$ (that is, we always punish the assignment that violates the constraints), and in A^* algorithm, we use $h(\mathbf{x},\mathbf{y}_{[1\ldots i]})$ (Eq.* (3.5)*) as our heuristic function and use $g(\mathbf{x},\mathbf{y}_{[1\ldots i]})$ (Eq.* (3.6)*)to obtain the score of the current partial assignment (that is, we use $g(\mathbf{x},\mathbf{y}_{[1\ldots i]}) + h(\mathbf{x},\mathbf{y}_{[1\ldots i]})$ as an estimation of the final score). Then, the A^* algorithm will always return the optimal solution of Eq* (3.2)*, the CCMs*

[2]This is the case for virtually all off-the-shelf structured statistical models, since their feature function $\Phi(\mathbf{x},\mathbf{y})$ can be decomposed. For example, if the task is a sequential tagging task and the feature function only captures the relationship of consecutive tokens, there exists an efficient Viterbi algorithm that can return the optimal sequence.

inference problem.

Proof. We prove this by showing that h is an admissible heuristic function. Since it is known that $\rho_k \geq 0$, for $k = 1 \dots m$ and the definition of Eq (3.2),

$$\max_{\mathbf{v}, \mathbf{v}_{[1\dots i]}=\mathbf{y}_{[1\dots i]}} f_{\Phi,\Psi}(\mathbf{x}, \mathbf{v}) \leq g(\mathbf{x}, \mathbf{y}_{[1\dots i]}) + h(\mathbf{x}, \mathbf{y}_{[1\dots i]})$$

Hence, we never underestimate the final score given the current partial assignment $\mathbf{y}_{[1\dots i]}$. Given that we are solving a maximization problem, h is a admissible heuristic function for the A^*algorithm. □

Approximated Search

While the A^* algorithm is technically sound, in this chapter, we use beam search to approximate the solution for the inference problem in Eq. (3.3). The advantage of using this procedure is that the memory usage of beam search is fixed while the memory usage of the A^* algorithm can be potentially big. We found that the approximated inference procedure performs very well in our experiments. The comparison of the three proposed inference algorithms on other domains is an interesting issue to address in future research.

3.2 Learning Constrained Conditional Models Based on HMM

In this section, we demonstrate how to apply the idea of CCMs to a commonly used Hidden Markov Model (HMM) and propose **HMM$^{\text{CCM}}$**. The new model naturally incorporates the constraints into HMM and makes it a very powerful model. In Section 3.1.2, we show that while the constraints introduce some overhead to the inference problem, we can still solve it efficiently in practice. *Interestingly, constraints do not add any overhead to our learning algorithm of HMM^{CCM}.*

For HMM, the readers can go to Section 2.2.2 to review the details and its relationships to linear classifiers. The rest of this section is organized as follows: we first review show how to derive the supervised training algorithm for HMM$^{\text{CCM}}$. In the second part of this section, we describe an instantiation of CoDL, a semi-supervised learning algorithm for CCMs and apply it to HMM$^{\text{CCM}}$.

3.2.1 HMM$^{\text{CCM}}$: Supervised Training

Assume that we have m constraints $\Psi_1, \Psi_2, \dots, \Psi_m$, and define $P(\Psi)$ as the probability that constraint Ψ is *violated*. In HMM$^{\text{CCM}}$, in order to combine statistical models and constraints (Eq. (3.2)), we adopt the idea of "product of experts" [Hinton, 1999], where the HMM is the expert that predicts the probability of the

label assignment, and the constraints component downgrades solutions that violate the constraints. This defines a new scoring function:

$$\begin{aligned}\Omega(\mathbf{x}_j, \mathbf{y}_j) &= \text{HMM Probability} \times \text{ Constraint Violation Score} \\ &= P_\Theta(\mathbf{x}_j, \mathbf{y}_j) \prod_{k=1}^{m} \prod_{i=1}^{T_j} P(\Psi_k)^{c_j^{k,i}} P(\neg\Psi_k)^{1-c_j^{k,i}},\end{aligned} \tag{3.7}$$

where Θ are the parameters of the HMM, $c_j^{k,i}$ is a binary variable equal to 1 if the label assignment to y_j^i violates the constraint Ψ_k with respect to partial assignment $\mathbf{y}_{j,[1\ldots i-1]}$, and Ψ_k indicates the event that the constraint Ψ_k is violated. It is important to notice that the constraint violation score captures the "degree of violation" by counting the penalty multiple times.

The new scoring function $\Omega(\mathbf{x}_j, \mathbf{y}_j)$ augments the original HMM with the constraints we have. It is important to notice that Eq. (3.7) is a CCM. $\log \Omega(\mathbf{x}_j, \mathbf{y}_j)$ can be rewritten in the form of (3.2) as follows:

$$\begin{aligned}\log \Omega(\mathbf{x}_j, \mathbf{y}_j) &\equiv \hat{f}_{w,\rho}(\mathbf{x}_j, \mathbf{y}_j) \\ &= \mathbf{w}^T \Phi(\mathbf{x}_j, \mathbf{y}_j) + \sum_{k=1}^{m} \log \frac{P(\Psi_k)}{P(\neg\Psi_k)} \sum_{i}^{T_j} c_j^{k,i} + c \\ &= \mathbf{w}^T \Phi(\mathbf{x}_j, \mathbf{y}_j) - \sum_{k=1}^{m} \rho_k d_{\Psi_k}(\mathbf{x}_j, \mathbf{y}_j) + c\end{aligned} \tag{3.8}$$

where $\rho_k = -\log \frac{P(\Psi_k)}{P(\neg\Psi_k)}$, $d_{\Psi_k}(\mathbf{x}_j, \mathbf{y}_j) = \sum_i^{T_j} c_j^{k,i}$ and c is a constant which does not affect the inference results. Note that the definition of the $d_{\Psi_k}(\mathbf{x}_j, \mathbf{y}_j)$ matches the one we defined earlier in Eq. (3.1).

To train HMM$^{\text{CCM}}$, we need to find $\mathbf{w}$ and ρ that maximize the new scoring function

$$\sum_{j=1}^{l} \log \Omega(\mathbf{x}_j, \mathbf{y}_j) = \sum_{j=1}^{l} \hat{f}_{w,\rho}(\mathbf{x}_j, \mathbf{y}_j) \tag{3.9}$$

It is worth noting several things. First, despite the fact that we use probabilities extensively in the scoring function, the function in Eq. (3.9) itself does *not* represent log likelihood of the dataset, since the augmented model does not have likelihood interpretation. Nevertheless, it is still a smooth concave function and its optimal value can be determined by setting the gradient to zero. Algorithm 7 describes the training procedure in detail. Interestingly, the solution resembles the standard HMM model. In fact, we can estimate the prior probability, transition probability and emission probability in exactly the same way as in HMM.

For the constraint violation part, a simple derivation shows that the optimal value for $P(\Psi_k)$ is obtained by

$$P(\Psi_k) = \frac{\sum_{j=1}^{l} \sum_{i}^{T_j} c_j^{k,i}}{\sum_{j=1}^{l} T_j}. \tag{3.10}$$

Note that the training procedure is "inference-free" in the sense that it is only based on partial counting. We do not need to solve any inference problems during the training but apply the constraints only at the test phase. In Section 3.1.2 we discuss several alternatives for efficient approximate and exact solutions to the inference problem. This completes the machinery for supervised training and inference in HMM$^{\text{CCM}}$.

Algorithm 7 Supervised Learning HMM$^{\text{CCM}}$. The algorithm optimizes the objective function $\sum_{j=1}^{l} \log \hat{f}_{w,\rho}(\mathbf{x}_j, \mathbf{y}_j)$ defined in Eq. (3.9).

Require: $\mathbf{L}$: labeled training set, $\{\Psi_k\}_{k=1}^{m}$: a set of constraints
1: Calculate Θ, the parameters of the HMM model with traditional HMM training.
2: Obtain $\mathbf{w}$ by applying the transformation on Θ described in [Roth, 1999; Collins, 2002]
3: **for** $k = 1 \dots m$ (constraint index) **do**
4: **for** $j = 1 \dots |\mathbf{L}|$ (training instance index) **do**
5: **for** $i = 1 \dots T_j$ (token position) **do**
6: $c_j^{k,i} \leftarrow \hat{\Psi}_k(\mathbf{x}_j; y_1^j, \dots, y_i^j)$
7: **end for**
8: **end for**
9: **end for**
10: $P(\Psi_k) = \frac{\sum_{j=1}^{l} \sum_{i}^{T_j} c_j^{k,i}}{\sum_{j=1}^{l} T_j}$.
11: $\rho_k = -\log \frac{P(\Psi_k)}{P(\neg\Psi_k)}$.
12: **return** $\mathbf{w}$, ρ

3.2.2 HMM$^{\text{CCM}}$: Semi-Supervised Learning

Acquiring labeled data is a difficult and expensive task. Therefore, an increasing attention has been recently given to semi-supervised learning, where large amounts of unlabeled data are used to improve models learned from a small training set [Yarowsky, 1995; Blum and Mitchell, 1998; Collins and Singer, 1999; Thelen and Riloff, 2002; Haghighi and Klein, 2006].

Before we discuss unsupervised and semi-supervised training in HMM$^{\text{CCM}}$, it is useful to introduce some new notation. Throughout this section, we assume there is only one unlabeled observed input example, $\mathbf{x}_U$, with associated unobserved output sequence $\mathbf{h}$ for the sake of simplicity. When we use some model or oracle to assign values to $\mathbf{h}$, we call the pair $(\mathbf{x}_U, \mathbf{h})$ pseudo-labeled data. Also, to avoid notation overload, we assume that we have one labeled and one unlabeled instance. This allows us to drop the sums of the form $\sum_j P(\mathbf{x}_j, \mathbf{y}_j)$ and write instead $P(\mathbf{x}, \mathbf{y})$. We note that this is done without loss of generality and for notational convenience only.

Traditionally, unsupervised and semi-supervised learning are done with the Expectation Maximization (EM) algorithm [Dempster et al., 1977; Borman, 2004]. Given only the unlabeled data $\mathbf{x}_U$, the EM algorithm is an iterative method for finding the θ which maximizes the objective function[3]

$$\theta^* = \arg\max_{\Theta} \log P_\Theta(\mathbf{x}_U) = \arg\max_{\Theta} \log \sum_h P_\Theta(\mathbf{x}_U|\mathbf{h}) P_\Theta(\mathbf{h})$$

Unfortunately, while it is possible to estimate the full distribution $P(\mathbf{h}|\mathbf{x}_U)$ when the model only captures "local decisions", it is very difficult to estimate this distribution when the long distance, expressive constraints are used [4].

In order to alleviate the difficulty of estimating the full distribution in the presence of constraints, we maximize the function $\log \Omega(\mathbf{x}_U, \mathbf{h})$ over both **the model parameters** $(\mathbf{w}, \rho)$ and **the label assignment** $\mathbf{h}$, which is equivalent to solving the problem:

$$(\mathbf{w}^*, \rho^*, \mathbf{h}^*) = \arg\max_{\mathbf{w},\rho,\mathbf{h}} \log \Omega(\mathbf{x}_U, \mathbf{h}) = \arg\max_{\mathbf{w},\rho,\mathbf{h}} \hat{f}_{w,\rho}(\mathbf{x}_U, \mathbf{h})$$

In contrast to EM, which only maximizes the likelihood of the unlabeled data by marginalizing hidden variables, we search for the best pseudo-label $\mathbf{h}$ and the model parameters $(\mathbf{w}, \rho)$ at the same time.

Our objective function can be optimized as follows (with initial $\mathbf{w}$ and ρ):

1. (Inference) Fix $\mathbf{w}$ and ρ, and optimize $\mathbf{h}$.

   The solution for $\mathbf{h}$ with fixed $\mathbf{w}$ and ρ is coming from Eq. (3.8) and can be found using the algorithms described in section 3.1.2. In other words, $\mathbf{h}$ is the solution of the following optimization problem:

$$\mathbf{h} \leftarrow \arg\max_{\mathbf{h}} \hat{f}_{w,\rho}(\mathbf{x}_U, \mathbf{h}) = \arg\max_{\mathbf{h}} \mathbf{w}^T \Phi(\mathbf{x}_U, \mathbf{h}) - \sum_{k=1}^{m} \rho_k d_{\Psi_k}(\mathbf{x}_U, \mathbf{h}),$$

2. (Learning) Fix $\mathbf{h}$, and optimize $\mathbf{w}$ and ρ.

   The solution of optimizing $\mathbf{w}$ and ρ can be obtained by applying Algorithm 7 on the pseudo-labeled data $(\mathbf{x}_U, \mathbf{h})$.

By the definition of Eq. (3.8), both steps are guaranteed to increase the objective function. Again, note that this procedure has an advantage over EM: it does not need to compute the conditional probability distribution, but only to get the best assignment $\mathbf{h}$ for the example $\mathbf{x}_U$.

[3] Recall that θ can be rewritten in the form of CCMs using $\mathbf{w}$ and ρ.

[4] [Ganchev et al., 2010] proposed to use expectation constraints to resolve this issue. See Section 3.5 for more discussions.

In HMM^{CCM}, the weight vector and the penalty vector resemble the probability distributions defined in section 3.2.2 so they can be estimated easily. As for EM, the objective function is not convex. Therefore, it is essential to have a good starting point.

Since a good starting point is necessary, we move our focus to "semi-supervised learning" and use a small amount of labeled examples to initialize the weight vector. One key difference between semi-supervised learning and unsupervised learning is that we need to balance labeled training data and unlabeled training data in order to have the best results. It is known that traditional semi-supervised training can degrade the learned model's performance [Nigam et al., 2000; Cozman et al., 2003]. [Nigam et al., 2000] has suggested balancing the contribution of labeled and unlabeled data to the parameters. In our algorithm, we use a similar intuition, but instead of weighting data instances, we introduce a smoothing parameter β which controls the convex combination of the *models* induced by the labeled and unlabeled data.

Algorithm 8 provides a pseudocode for our semi-supervised algorithm called **CoDL** (COnstraint-Driven Learning) in [Chang et al., 2007]. We note that CoDL is a general procedure, and as such, can and will be applied to models other than HMM^{CCM}in later sections.

As is often the case in semi-supervised learning, the algorithm can be viewed as a process that improves the model by generating feedback through *labeling* unlabeled examples. Our algorithm pushes this intuition further, in that the use of constraints allows us to better exploit domain information as a way to label, along with the current learned model, unlabeled examples. Given a small amount of labeled data and a large unlabeled pool, our framework initializes the model with the labeled data and then repeatedly:

1. Uses *constraints* and the learned model to label the instances in the pool (line 5)
2. Updates the model by newly labeled data (line 8).

This way, we can generate *better* "training" examples during the semi-supervised learning process. Note that line 8 also performs the linear combinations among models with the parameter β.

CoDL uses constraints as prior knowledge in the semi-supervised setting. We later show that prior knowledge plays a crucial role when the amount of labeled data is limited. CoDL makes use of CCMs, which provides a good platform to combine the learned models and prior knowledge. It is very important to note that $CoDL$ can naturally extend as a general purpose semi-supervised learning algorithm for any CCM model.

Relations to EM algorithms Recall that we review EM and hard EM algorithms in Section 2.5.1. It is interesting to note that in the absence of constraints, CoDL reduces to "hard-EM", which only finds the best assignment in every step. To further illustrate the difference between CoDL, "hard-EM" and (soft) EM,

Algorithm 8 Constraint driven learning algorithm, which uses constraints to guide semi-supervised learning. Note that this version is a specification of the general CoDL algorithm (Figure 3.1).

Require: **L**: labeled training set, **U**: unlabeled dataset N: learning cycles
β : balancing parameter with the supervised model,
$\{C\}$: a set of constraints,
$learn(.)$: a supervised learning algorithm

Initialize $(\mathbf{w}, \rho) = (\mathbf{w}_0, \rho_\mathbf{0}) = \mathbf{learn}(\mathbf{L})$.
for N iterations **do**
  $\mathbf{T} = \emptyset$
  for $\mathbf{x} \in \mathbf{U}$ **do**
    $\hat{\mathbf{h}} \leftarrow \arg\max_\mathbf{y} \mathbf{w}^T \Phi(\mathbf{x}, \mathbf{y}) - \sum_{k=1}^{m} \rho_k d_{\Psi_k}(\mathbf{x}, \mathbf{y})$
    $\mathbf{T} = \mathbf{T} \cup \{(\mathbf{x}, \hat{\mathbf{h}})\}$
  end for
  $(\mathbf{w}, \rho) = \beta(\mathbf{w}_0, \rho_\mathbf{0}) + (1 - \beta)\mathbf{learn}(\mathbf{T})$
end for

consider the problem of unsupervised part-of-speech tagging. In (soft) EM, we do not find the most likely label assignment given the data as part of the training procedure. On the other hand, when estimating the model parameters, we smoothed over all possible label assignments weighted by their likelihood. When we run "hard-EM", we get the most likely label assignment as part of the procedure. Like "hard-EM", CoDL also finds only the best assignment during the learning processing. However, unlike "hard-EM", CoDL makes use of constraints to guide the learning process. More comparisons between CoDL, "hard-EM" and EM will be discussed in Section 3.4.3.

3.2.3 HMM$^{\text{CCM}}$ versus HMM$^{\text{CCM}}_{\infty}$

We would like to stress again that HMM$^{\text{CCM}}$ is just one algorithm of applying CCM models on Hidden Markov Models. One simple variation is to use "hard constraints" in CCM (called HMM$^{\text{CCM}}_{\infty}$, given that the penalty is infinity). The advantage of using hard constraints in CCM is that we do not need to learn the penalty vector ρ, and the learning algorithm for supervised setting is exactly the same as HMM. The semi-supervised learning algorithm for HMM$^{\text{CCM}}$(Algorithm 8) can be directly applied to HMM$^{\text{CCM}}_{\infty}$. The disadvantage of HMM$^{\text{CCM}}_{\infty}$ is that it always enforces the constraints, which can in fact be violated in the gold data. See Section 3.4.5 for more comparisons between these two CCM approaches.

3.3 Tasks and Data

In this section we introduce two information extraction problems which we used to evaluate the models and ideas presented in this chapter. In both problems, given input text, a set of pre-defined fields is to be identified. Since the fields are typically related and interdependent, these kinds of applications provide

(a) [*AUTHOR* Lars Ole Andersen .] [*TITLE* Program analysis and specialization for the C programming language .] [*TECH-REPORT* PhD thesis ,] [*INSTITUTION* DIKU , University of Copenhagen ,] [*DATE* May 1994 .]

(b) [*AUTHOR* Lars Ole Andersen . Program analysis and] [*TITLE* specialization for the] [*EDITOR* C] [*BOOKTITLE* Programming language] [*TECH-REPORT* . PhD thesis ,] [*INSTITUTION* DIKU , University of Copenhagen , May] [*DATE* 1994 .]

Figure 3.2: Error analysis of a HMM model. The labels are underlined to the right of each open bracket. The correct assignment is shown in (a). The predicted assignment (b) violates some constraints, most obviously, the punctuation marks.

Citations	
Start	The citation can only start with author or editor.
AppearsOnce	Each field must be a consecutive list of words, and can appear at most once in a citation.
Punctuation	State transitions must occur on punctuation marks.
BookJournal	The words *proc, journal, proceedings, ACM* are *JOURNAL* or *BOOKTITLE*.
Date	Four digits starting with 20xx and 19xx are *DATE*.
Editors	The words *ed, editors* correspond to *EDITOR*.
Journal	The word *journal* are *JOURNAL*.
Note	The words *note, submitted, appear* are *NOTE*.
Pages	The words *pp., pages* correspond to *PAGE*.
TechReport	The words *tech, technical* are *TECH_REPORT*.
Title	Quotations can appear only in titles.
Location	The words *CA, Australia, NY* are *LOCATION*.

Table 3.1: The list of constraints used in the citations domain. Some constraints are relatively difficult to represents in traditional models.

a good test case for an approach like ours (the data for both problems is available at: `http://L2R.cs.uiuc.edu/~cogcomp/Data/IE.tgz`.[5]).

The first task is to identify fields from citations [McCallum et al., 2000]. The data originally included 500 labeled references, and was later extended with 5,000 unannotated citations collected from papers found on the Internet [Grenager et al., 2005]. Given a citation, the task is to extract the fields that appear in the given reference. There are 13 possible fields including author, title, location, etc.

To gain an insight into how the constraints can improve the model accuracy and guide semi-supervised learning, assume that the sentence shown in Figure 3.2 appears in the unlabeled data pool. Part (a) of the figure shows the correct labeled assignment and part (b) shows the assignment labeled by a HMM trained on 30 labeled samples. However, if we apply the constraint that state transition can occur only on punctuation marks, the same HMM will result in the correct labeling (a). Therefore, by adding the improved labeled assignment we can generate better training samples during semi-supervised learning. In fact, the requirements on punctuation marks are only some of the constraints that can be applied to this

[5]Note that we used different training-testing split in our experiments than [Grenager et al., 2005].

Advertisements	
FieldLength	Each field must be at least 3 words long.
Punctuation	State transitions can occur only on punctuation marks or the newline symbol.
Address	The words *address, carlmont, st, cross* are *ADDRESS*.
Available	The words *immediately, begin, cheaper* are *AVAILABLE*.
Contact	The words **Phone*, *Email** are *CONTACT*.
Features	The words *laundry, kitchen, parking* are *FEATURES*.
Neighborhood	The words *close, near, shopping* are *NEIGHBORHOOD*.
Photos	The words *http, image, link* are *PHOTOS*.
Rent	The words *$, *Money** are *RENT*.
Restrictions	The words *smoking, dogs, cats* are *RESTRICTIONS*.
Roomates	The words *roommates, respectful, drama* are *ROOMMATES*.
Size	The words *sq, ft, bdrm* are *SIZE*.
Utilities	The words *utilities, pays, electricity* are *UTILITIES*.

Table 3.2: The list of constraints used in the advertisements domain. Some constraints are relatively difficult to represents in traditional models. *Phone*, *Email* and *Money* are tokens corresponding to phone numbers, email addresses and monetary units, which were identified in text using regular expressions. This preprocessing was done before applying any training algorithms.

problem. The set of constraints we used in our experiments appears in Table 3.1. Note that some of the constraints are non-local and are very intuitive for people, yet it is very difficult to inject this knowledge into most models.

The second problem we consider is extracting fields from advertisements [Grenager et al., 2005]. The dataset consists of 8,767 advertisements for apartment rentals in the San Francisco Bay Area downloaded in June 2004 from the Craigslist website. In the dataset, only 302 entries have been labeled with 12 fields, including *size, rent, neighborhood, features*, and so on. The data was preprocessed using regular expressions for phone numbers, email addresses and URLs. The list of the constraints for this domain is given in Table 3.2. We implement some global constraints and include unary constraints which were largely imported from the list of seed words used in [Haghighi and Klein, 2006]. We slightly modified the seed words due to differences in pre-processing.

3.4 Experimental Results

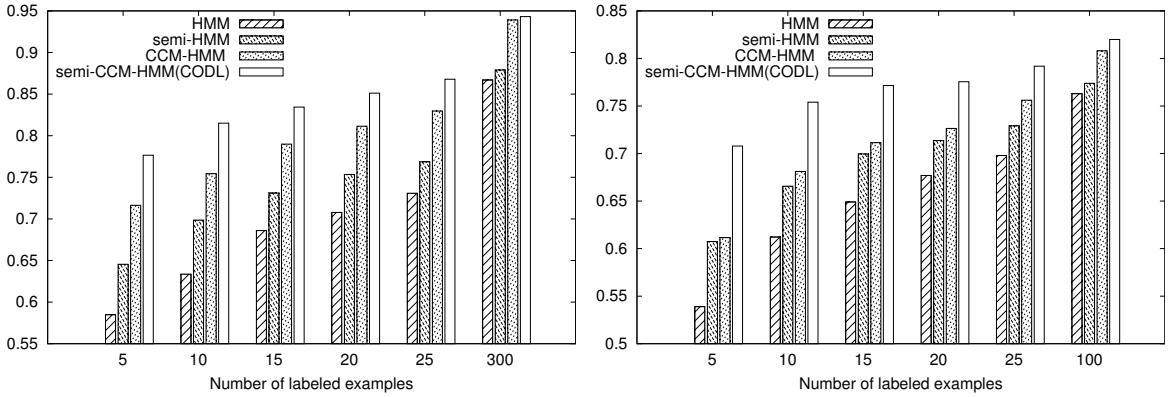
We empirically verify the effectiveness of combining constraints and statistical models in this section. The experiments are designed to answer the following series of research questions.

(1) How important is it to add knowledge into statistical models? More specifically:

- How does HMM^{CCM} perform compared to the original HMM?

Citations				
# Labeled	Supervised		Semi-Supervised	
Samples	HMM	HMM^{CCM}	HMM	HMM^{CCM}
5	58.48	71.64 (31.69 %)	64.55	77.65 (36.96 %)
10	63.37	75.44 (32.94 %)	69.86	81.51 (38.67 %)
20	70.78	81.15 (35.49 %)	75.35	85.11 (39.61 %)
300	86.69	93.92 (54.29 %)	87.89	94.32 (53.07 %)
Advertisements				
5	53.90	61.16 (15.74 %)	60.75	70.79 (25.58 %)
10	61.21	68.12 (17.80 %)	66.56	75.40 (26.42 %)
20	67.69	72.64 (15.32 %)	71.36	77.56 (21.63 %)
100	76.29	80.80 (19.02 %)	77.38	82.00 (20.40 %)

Table 3.3: The impact of using constraints for supervised and semi-supervised learning (generative HMM). Note that while semi-supervised HMMs performs much better than supervised HMMs, using constraints still improves the semi-supervised HMMs significantly. The numbers in the brackets denote error reduction over similar algorithm without constraints.



(a) Citations (b) Advertisements

Figure 3.3: The utility of constraints in semi-supervised setting.

- How efficient is it to use constraints as a supervision resource?

Note that these two questions address different aspects of using constraints in CCMs. The first question addresses the amount of improvement obtained by adding constraints. The second question, on the other hand, addresses the issue of using constraints as a *supervision resource* compared to labeling examples.

(2) How do our CCM training algorithms compare against other algorithms?

- Comparisons between EM, hard EM and the CoDL algorithm.
- How is the CCM approach compared to other approaches?

First, we ask the question about what are the benefits of using CoDL as opposed to the standard EM and hard EM algorithms. We then compare CoDL to other approaches of encoding long distance relationships. Note that we can often design a heavily engineered and tailored model for a specific task. However, this process is challenging and time consuming, and must be repeated for every new task. On the other hand, CCMs provide an easy to use model-specification language that works for all tasks. Therefore, we compare HMM^{CCM} to several tailored models to see if our general purpose model can match the performance of a specifically designed model. It is natural to expect that a tailored model will perform at least as well as CoDL, however if CoDL matches the performance of a tailored model, we consider it a success. We also compared the results to the recent approaches of using expectation constraints [Bellare et al., 2009].

(3) What are the properties of CCMs and CoDL? These questions include:

- In HMM^{CCM}, do we need to learn the penalty vector ρ?
- What is the utility of each constraint in our tasks?
- In CoDL, how important it is to tune β?

Among all of the research questions, the most important one is to verify whether adding constraints can improve the models or not. Again, while CCMs are not the only way to incorporate constraints, they provide a nice interface so that users do not need to invent a tailored model for every task.

The results reported in this and the following sections are token-level accuracies, which were averaged on 5 randomly generated training sets (each round with different labeled set). We tested on a fixed test set and a fixed development set, both containing 100 labeled samples. When semi-supervised learning algorithms are used, we use 1000 held-out unlabeled examples as part of our training data in both domains. This setting was first used by [Grenager et al., 2005; Chang et al., 2007; Haghighi and Klein, 2006] and then

used by many other works. In the semi-supervised setting we ran 5 iterations of CoDL. The reason that we choose to run only 5 iterations is that our semi-supervised learning procedure usually converges very fast (See Section 3.4.3 for more details).

3.4.1 How does HMM^{CCM} perform compared to the original HMM?

To see the impact of using constraints, we compare HMM and HMM^{CCM}in Table 3.3. For our HMM implementation, we use the standard partial counting approach to train the supervised model and use the hard EM algorithm to train the semi-supervised model. For more discussions between hard EM, EM and HMM^{CCM}, please refer to Section 3.4.3. For both HMM and HMM^{CCM}, the parameter β was set to 0.1 for both approaches. We also ran 5 iterations for the hard EM algorithm. The effect of applying constraints is significant: for example, when there are only 5 labeled examples, the constraints push the accuracy from 58% to 71% in citation domain and from 53% to 61% in advertisement domain. The results for more data points are shown in Figure 3.3.

In the semi-supervised setting, adding constraints improves the HMM models more dramatically. One interesting result (see Table 3.3) is that with small amount of labeled data, the benefit of applying constraints is greater in the semi-supervised setting than that in the supervised setting. That is, with 5 labeled samples, in the advertisements domain, applying constraints in the supervised setting reduces the error rate by 15.47% while applying constraints in the semi-supervised setting reduces the error rate by 25.58%. Similarly, on the citations domain, applying the constraints reduces the error rate by 31.69% in the supervised setting, while in the semi-supervised setting, the error rate decreases by 36.96%. This result highlights the utility of using constraints in semi-supervised setting.

While with small amounts of labeled data, the majority of improvement comes from guiding semi-supervised learning with constraints, the situation is reversed when more labeled data is available. In this scenario, the parameters of the basic model are learned fairly well and semi-supervised learning cannot improve them further. In this case, most of the improvement comes from applying the constraints, while the utility of semi-supervised learning is limited. Nevertheless, for the advertisements domain, semi-supervised learning with constraints outperforms the supervised protocol with constraints by 1.2% (82.00 versus 80.80) even when 100 labeled samples are available.

3.4.2 How efficient is it to use constraints as a supervision resource?

We would like to view the results in the previous section from a different perspective: we can acquire knowledge either by adding constraints or adding more labeled samples. Here we view the "constraints"

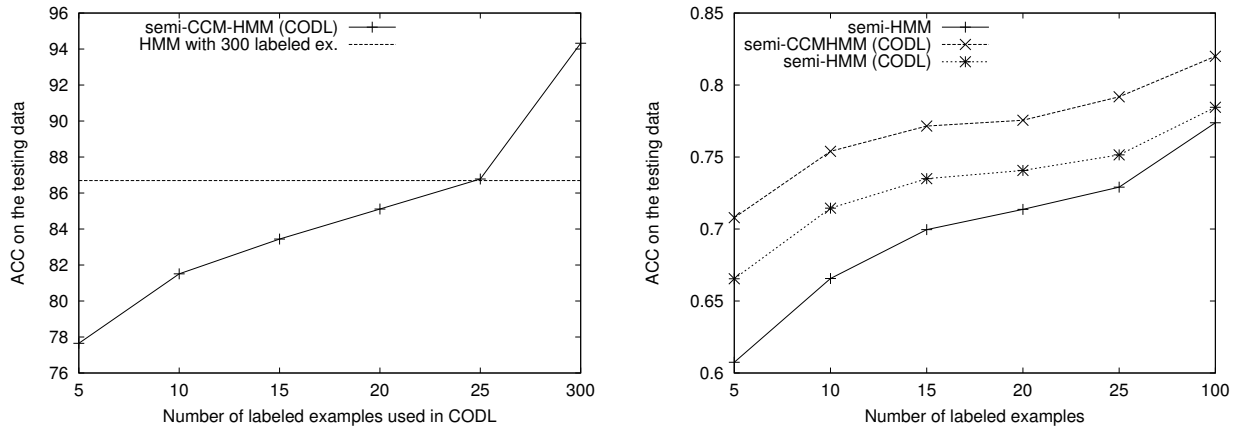


Figure 3.4: **Using constraints as supervision resource. Left:** In *citations* domain, with 25 labeled citations, our semi-supervised algorithm performs competitively to the supervised version trained on 300 samples. **Right:** In *ads* domain. Note that in semi-HMM(CoDL), we train the HMM model with CoDL, but *do not* apply the constraints in the testing phase. The goal of this experiment is to see how well can CoDL guide the statistical component of the CCMs (in this case, the statistical component is HMM). The superior performance of semi-HMM(CoDL) shows that CoDL indeed can successfully guide the HMM.

as a supervision resource rather than a part of the models and examine the utility of adding constraint as opposed to adding more labeled data.

The results in Table 3.3 clearly suggest that adding constraints is more efficient than adding labeled samples. Note that the model driven by constraints and 20 labeled samples outperforms the traditional HMM trained with 100 labeled samples on the advertisements domain and is only slightly worse compared to the traditional HMM trained with 300 labeled samples on the citations domain.

Figure 3.4 strengthens the claim of using constraints as a supervision resource. The left figure shows that in the citations domain, the semi-supervised HMM^{CCM}achieves with 25 labeled samples similar performance to the supervised version without constraints with 300 labeled samples. The right figure distinguishes the impact constraints made in the training phase and in the testing phase. Semi-HMM(CoDL) represents the results where we train the HMM model with CoDL but do *not* apply the constraints in the testing phase. The goal of this experiment is to see how well can CoDL guide the statistical component of the CCMs (in this case, the statistical component is HMM). Semi-HMM(CoDL) and HMM have the same expressivity, but the former is trained with constraints (using CoDL) while the latter is trained without using constraints. The superior performance of semi-HMM(CoDL) shows that CoDL indeed can successfully guide the HMM. This demonstrates the value of constraints as an additional supervision resource.

In other words, injecting constraints into the model requires design effort, but we believe that the increased expressivity of the model is well worth the effort. For example, applying constraints to the basic

HMM trained on 300 labeled samples, improves the accuracy from 86.66% to 94.03%. We wanted to get a rough estimate on the number of additional labeled samples that are needed to achieve similar performance with the traditional HMM. Since the performance of the semi-supervised model on the citations domain is 94.51%, we assume that the labels assigned to the unlabeled examples are fairly accurate. Therefore, we used our final model to label the unlabeled data and appended it to the training set. This way, we had 1300 labeled samples, which we used to train an HMM without constraints. The resulting accuracy was 88.2%, still far from 94.51%.

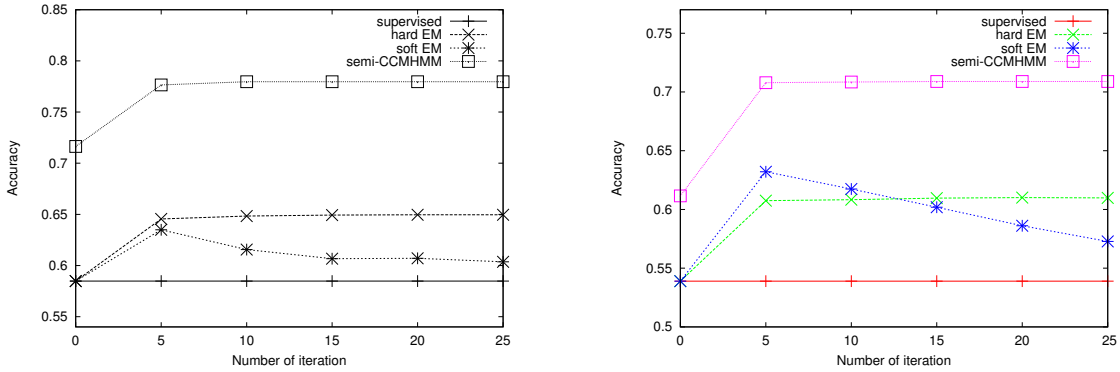
Moreover, when we trained the HMM on the training *and the test set* (400 labeled samples altogether), the resulting accuracy was 95.63%. That is, even after seeing the test samples, the HMM does not have the expressivity to learn the true concept. On the other hand, when the constraints are applied, the accuracy goes up to 99.22%. Therefore, we speculate that the basic HMM is simply unable to capture the expressive declarative aspects of the problem, no matter how much labeled data is available.

3.4.3 Comparisons between EM, hard EM and the CoDL algorithm

Expectation Maximization is the standard semi-supervised learning algorithm for generative models. In Section 3.2.2 we showed that our semi-supervised learning algorithm has an objective function which is different from that of EM, and very similar in spirit to hard-EM, which only find the best assignment instead of finding the full posterior distribution. In fact, when constraints are not used, our learning procedure is identical to hard-EM. The difference between EM and hard-EM is that the former requires to predict a full posterior distribution according to the parameters, while the latter one only requires finding the best assignment. It is important to note that when hard constraints are used, it is very difficult to calculate the distribution $P(\mathbf{y}|\mathbf{x})$ because of the long distance relationships. Note that one can relax constraints by transforming them into expectation constraints and make calculating posterior tractable [Ganchev et al., 2010]. Please see the discussion in Section 3.5 for more details. In Section 3.4.4, we compare our algorithm to published results of a framework called alternating projections (AP), which uses expectation constraints in an EM-like algorithm.

In this section, we compare three approaches: **EM**, **hard-EM** and **semi-HMM$^{\text{CCM}}$**. Note that the difference between hard-EM and semi-HMM$^{\text{CCM}}$ is the use of the constraints. The experimental results of using 5 labeled examples are in Figure 3.5. For all of the approaches, we put more weight on the labeled data and less weight on the unlabeled data ($\beta = 0.9$). Putting more weights on the supervised model helps all three approaches.

First, in our experiments, we find that the accuracy of the EM approach degrades as the number of



(a) Citations, 5 training samples

(b) Ads, 5 training samples

Figure 3.5: The testing accuracy vs. number of iterations of semi-supervised learning in the citations and ads domain with 5 labeled examples. Note that the difference between hard-EM and semi-HMM$^{\text{CCM}}$ is the usage of the constraints. See the text for more discussion.

iterations grows in Figure 3.5. While EM tries to maximize the log likelihood of the observed variables, it does not necessary mean that the model will get better performance as the number of the iterations grows [Liang and Klein, 2008]. This observation is consistent with [Merialdo, 1994; Liang and Klein, 2008; Collins-thompson, 2009]. Therefore, in our experiments, we find that while the EM approach can be better than the hard-EM approach (see Figure 3.5(b), number of iterations equals to 5.), the hard-EM approach is generally more stable as the number of iterations grows. In fact, we hardly see any change for the hard-EM approach after 5 iterations, and this is the reason why we choose to run only 5 iterations for the semi-supervised learning algorithms.

Recall that Algorithm 8 is similar to the hard-EM procedure but allows using constraints. Figure 3.5 shows that the **semi-HMM$^{\text{CCM}}$** approach is significantly better than both **EM** and **hard-EM**. Figure 3.5 also demonstrates that it is important to use constraints in the semi-supervised learning algorithms when the size of the labeled data is small.

3.4.4 How is the CCM approach compared to other approaches?

In this section, we compare the proposed approach to other existing approaches. First, we compare to a "tailored model" with a semi-CRF model [Sarawagi and Cohen, 2004], which integrates the long distance relationships and local relationships together in one model. Second, we compared CCM to alternating projection (AP) framework [Bellare et al., 2009], which can be considered as a discriminative special case for the Posterior Regularization framework [Ganchev et al., 2010]. Note that the AP framework is not a "tailored

model" given that the AP framework also keeps their baseline model and the constraints separately.

Compared to CRF and Semi-CRF We have seen that constraints allow us to capture the properties of the problem which HMM cannot. However, it can be argued that we can modify a HMM model with certain amount of work. For example, if we segment the text on punctuation marks, and use a multinomial emission model for each state, we can capture the "transition on punctuation marks" constraint. The question is whether such a tailored model will perform significantly better than an off-the-shelf HMM with our constraints. Before we go into further discussion, we note that an HMM cannot be tailored to capture all the constraints, for example the constraint "each field can appear only once" cannot be injected into an HMM model by tailoring segmentation, emission and transition components. Also, we argue that it is significantly more time consuming to engineer and implement a tailored model (particularly with semi-supervised training) than to take an off-the-shelf model and downweight the output space with constraint violation penalties. Moreover, the tailored model we consider in this section can also be augmented with additional declarative constraints. In fact, the tailored model can be considered as an instance of CCMs, but with a tailored way to inject the constraints. Therefore, if the more general way of injecting constraints which we propose in this chapter works competitively to a tailored model, we consider this a success for CCMs.

We choose Semi-Markov CRF (semi-CRF) [Sarawagi and Cohen, 2004] as our tailored model competitor. Semi-CRF operates on a segment level rather than on a token level. That is, we define a segmentation to be $s = (s_1, \ldots, s_T)$ where each s_i ($1 \leq i \leq T$) is a triple (t_i, u_i, l_i) with t_i denoting the segment beginning, u_i the segment end, and l_i - the assigned label. Training a semi-CRF involves finding the weights, and inference involves finding the segmentation which optimizes the following function:

$$P(s|\mathbf{x}, W) = \frac{1}{Z(\mathbf{x})} \exp^{\mathbf{w}^T \Phi(\mathbf{x}, s)}$$

where $\mathbf{x}$ is the input sequence, s is the segmentation, $\Phi(\mathbf{x}, s)$ are the features extracted from the segmentation s of $\mathbf{x}$, and $Z(\mathbf{x}) = \sum_{s'} \exp^{\mathbf{w}^T \Phi(\mathbf{x}, s')}$.

This allows the model to extract segment-level features, such as string edit distance to a multi-token dictionary of entities, and an average field length. Semi-CRFs exploit the fact that in many applications, adjacent tokens take the same label, an assumption that indeed holds on our data as well. For our problems, semi-CRFs have an attractive quality– they allow to inject segment-level features like "segment length" and "segment ends on a punctuation mark." Another attractive property of semi-CRFs is that the computational penalty paid for adding the segment-level expressivity when compared to first-order CRF is linear in L, the

Training Instances	HMM	HMM^{CCM}	CRF	semi-CRF	semi-CRF^{+}
5	58.48	63.68	51.43	50.69	60.14
10	63.37	66.88	54.61	50.38	62.51
20	70.78	77.52	63.92	62.96	72.22
300	86.69	93.35	89.09	92.46	94.60

Table 3.4: Comparison between HMM^{CCM}and tailored models citations domain. Also note that semi-CRF is a *supervised* learning algorithm and semi-CRF^{+} use extra features such as the segmentation length. Also note that in this table, both HMM^{CCM} and semi-CRF only use the "Punctuation" constraint and all the other models do not use any constraint. We only show the results for the citation domain, because we could not tune semi-CRF to perform competitively on the advertisements domain using the same features.

maximal segment length. We stress that there are important differences between semi-CRF (which tailors the training and inference to accommodate segment-level features), between order-L CRF and between CRF with constrained output space. The comparison of the models is outside the scope of this chapter, the interested reader is referred to [Sarawagi and Cohen, 2004] and [Roth and Yih, 2005].

Semi-CRFs were originally proposed for the problem of named entity recognition [Cohen, 2004; Sarawagi and Cohen, 2004] with significant performance gains due to the ability of the model to capture inexact segment-level string matching to gazetteers. The computational penalty is high – the maximal length of a named entity was assumed to be 4, so the inference for semi-CRF is 4 times slower than for token-level first order CRF. In our problems, however, the maximum field length for citations was 100 tokens, and the maximum field length for the advertisements was 200 tokens, making the training and the inference of the model prohibitively slow.

Therefore, we compared the behavior of the competing models with a single constraint - "transition on punctuation marks." This constraint is readily injected into the semi-CRF by adding the feature indicating whether a segment ends with punctuation mark. We compared the following models: HMM, HMM^{CCM}, CRF and semi-CRF. The HMM and the CRF models come without constraints. HMM^{CCM}and semi-CRF use a single constraint- "transition on punctuation marks." The default implementation of semi-CRF makes use of multiple additional features, including token normalization, token prefixes, suffixes, whether the token contains only digits, and also, most importantly - the segment length. To make a fair comparison, we removed most of the features, and used the same token-level features as in HMM. However, we were curious to see how much the segment length feature can improve the performance, particularly since it comes built in with the tailored model design. Therefore, we have 2 flavors of semi-CRFs: semi-CRF and semi-CRF^{+}, one with and one without the segment length feature.

The results are summarized in Table 3.4. We note that CRF is a discriminative model, therefore, as it is often the case, it performs worse than the generative model (HMM) when there is little training data

and outperforms the HMM when a lot of training data is available [Ng and Jordan, 2001]. Furthermore, semi-supervised training in discriminative models is substantially harder. We also note that the injection of constraints in the generic way proposed in the CCM framework improves the HMM performance from 86.66 to 93.35 with 300 labeled samples. Injecting the constraint "state transitions can occur only on punctuation marks" by tailoring CRF improves the performance from 89.09 for CRF to 92.46 for semi-CRF, and including the additional feature of segment length in semi-CRF$^+$ further improves the performance to 94.60 on the citation domain with 300 labeled samples. Therefore, we see that although the tailored model has some potential, injecting constraints in the CCM framework actually brings bigger performance gains.

It is important to note that while semi-CRF performed very well on the citations domain, we failed to tune it to perform competitively on the advertisements domain. We suspect that the reason from the fact that we use very simple features in our CRF model given that he advertisements domain is a lot more difficult than the citation domains.

Compared to Approaches with Expectation Constraints The Posterior Regularization Framework (PR) [Ganchev et al., 2010] observes that while calculating posterior with hard constraints can be difficult, calculating posterior distribution with expectation constraints can be tractable with careful designs. Please see Section 3.5 for more discussions.

The Alternating Projections framework [Bellare et al., 2009] can be considered as a special case of PR with discriminative models. In [Bellare et al., 2009], the authors perform experiments on the citation dataset in a very similar setting as the one we used in this chapter, although the training-testing data split is a bit different. Table 3.5 is created by getting our results in Table 3.3 and citing their reported results in [Bellare et al., 2009].

In Table 3.5, there are two AP approaches: AP-T uses the testing dataset as the unlabeled dataset while AP-I uses another unlabeled dataset to bootstrap the results. Note that the performances of AP are quite similar to the CCM models. However, the baseline model used by AP is a much stronger CRF model compared to both the CRF or the HMM models we built in this chapter. Their baseline CRF model is built with many different additional features including token features (identity, token prefixes, token suffixes and character n-grams), lexicon features (the token is presence of a token in a lexicon of author names, journal names, etc.), regular expressions (common patterns for years and page numbers), and other bigram features.

# Labeled	AP-T	AP-I	semi-HMM^{CCM}
5	75.6	74.6	77.65
20	85.4	85.1	85.11
300	94.0	94.3	94.32

Table 3.5: Comparison to Alternating Projections [Bellare et al., 2009], a discriminative special case of Posterior Regularization [Ganchev et al., 2010]. The AP results are cited from [Bellare et al., 2009], while the CCM results are from Table 3.3.

3.4.5 In HMM^{CCM}, do we need to learn the penalty vector ρ?

Previous works ([Punyakanok et al., 2005b; Roth and Yih, 2005]) have used "hard" constraints to disallow any label assignments that violates them. In the problems considered in this work, several gold assignments in the training set violate the constraints. Therefore, it seems beneficial to learn a constraint penalty vector ρ. As we mentioned in Section 3.2.3, HMM^{CCM} is just one instance of a CCM model, and we can also have the CCM version of HMM using hard constraints HMM$^{CCM}_{\infty}$. Table 3.6 shows that with sufficient amount of labeled data, HMM^{CCM} (learning with soft constraints) outperforms HMM$^{CCM}_{\infty}$ in both the citations and the advertisements domain.

(a)-Citations

Training samples	5	10	20	300
semi-HMM^{CCM}	77.65	81.51	85.11	94.32
semi-HMM$^{CCM}_{\infty}$	78.18	81.11	85.16	92.80

(b)-Advertisement

Training samples	5	10	20	300
semi-HMM^{CCM}	70.79	75.40	77.56	82.00
semi-HMM$^{CCM}_{\infty}$	69.91	73.46	75.25	79.59

Table 3.6: Comparison of using hard constraints and soft constraints in semi-supervised learning.

3.4.6 What is the utility of each constraint in our tasks?

To highlight the impact of each constraint, in the following experiments, rather than learning the penalty of constraints violation from the data, we have enforced hard constraints. Tables 3.7 and 3.8 show the contribution of each constraint individually. Table 3.7 shows that the constraint *Start* (which requires the citations to start with either author or editor) actually hurts the performance in the semi-supervised setting. The constraint *AppearsOnce* hurts the performance in the supervised setting, but improves it significantly in the semi-supervised setting. Global constraints, such as *Punctuation*, improve the performance the most. Another interesting result is that while local constraints do not improve the performance significantly (even in the semi-supervised setting), when combined with the global constraints, they lead to significant performance improvements. While Tables 3.7 and 3.8 show the impact of using hard constraints, it is worthwhile

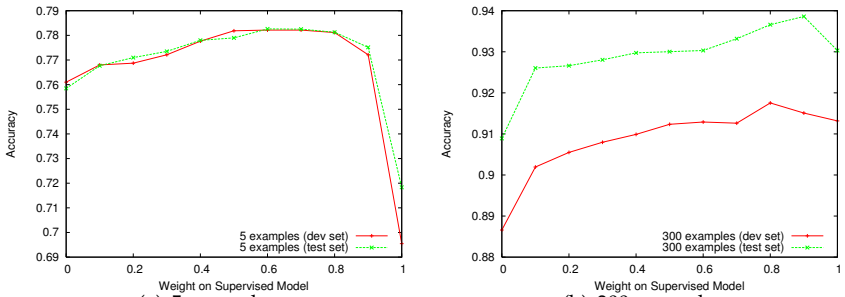
Constraint	Supervised	Semi-Supervised
None	**58.48**	**64.55**
Start	58.52	64.52
AppearsOnce	58.69	65.92
Punctuation	63.68	71.23
BookJournal	58.96	64.68
Date	61.50	66.76
Editor	58.70	64.77
Journal	58.66	64.73
Note	58.55	64.61
Pages	58.77	64.68
TechReport	58.73	64.43
Title	59.66	65.54
Location	58.81	64.97
ALL	**71.64**	**77.65**

Table 3.7: Utility of hard constraints on the citations domain; supervised and semi-supervised setting with 5 training examples.

to note that the soft constraints perform better (see Table 3.6).

3.4.7 In CoDL, how important it is to tune β?

It is well known (for example, [Cozman et al., 2003]) that semi-supervised learning can degrade the performance when the assumptions of the model do not hold on the data. One way to overcome this problem is to reweight labeled and unlabeled samples. Recall that in analogy to [Nigam et al., 2000], when performing semi-supervised learning, we use the weighted average of the models trained on labeled and unlabeled data (see Section 3.2.2).



(a) 5 examples (b) 300 examples

Figure 3.6: The performance of HMM semi-supervised algorithm with constraints on the citations domain. The x axis represents the weight of the supervised model. When weighting parameter is 0, there is no smoothing at all and the model is equivalent to pure unsupervised training. When weighting parameter is 1, the results will be equivalent to those of a purely supervised model.

Figure 3.6 summarizes the effect of weighting parameter β in line 8 of Algorithm 8. As expected, when the amount of the labeled data is increased, the model performs better with smaller values of β. Note that

Constraint	Supervised	Semi-Supervised
None	**53.90**	**60.75**
FieldLength	54.38	63.85
Address	53.95	60.85
Available	53.96	60.77
Contact	53.90	60.75
Features	54.20	60.84
Neighborhood	53.90	60.75
Photos	53.90	60.67
Rent	53.89	60.80
Restrictions	54.27	60.79
Roomates	53.90	60.78
Size	54.22	61.11
Punctuation	58.64	68.81
Utilities	54.05	60.89
All	**61.16**	**70.79**

Table 3.8: Utility of hard constraints on the advertisements domain; supervised and semi-supervised setting with 5 training examples.

in the experiments reported in this chapter, we do not adjust the weighting parameter for different size of labeled data. We are always using a fixed weighting parameter $\beta = 0.9$, which is not an optimal value for small training sets (for example, for 5 labeled examples).

3.5 Related Work

In this section we review selected publications related to the CCM framework. We first discuss several publications that can be considered as the special cases of the CCMs framework. Other related publications are grouped in three different categories and the corresponding discussions are also included in this section. Note that we defer the discussions of CoDL and several modern constraint-based EM algorithm in the next chapter.

Our work on CCMs builds on several works that can be considered as special cases of CCMs. In most cases, these works combine hard constraints with learning algorithms in the supervised setting. The first work in this line [Roth and Yih, 2004] (extended in [Roth and Yih, 2007]) suggests a formalism that combines constraints with linear models on information extraction tasks. They use linear inequalities and suggest Integer Linear Programming as the inference framework. Following [Roth and Yih, 2004, 2007], a series of works proposed and studied models that incorporate learned models with declarative constraints with successful applications in Natural Language Processing and Information Extraction, including semantic role labeling [Roth and Yih, 2005; Punyakanok et al., 2005a, 2008], summarization [Clarke and Lapata, 2006; Barzilay and Lapata, 2006], generation [Marciniak and Strube, 2005], co-reference resolution [Denis

and Baldridge, 2007], and parsing [Riedel and Clarke, 2006; Martins et al., 2009a].

Most of these works use only hard constraints with the factored approach and supervised classifiers. In contrast, we introduce soft constraints (modeled as the degree of violating a constraint), into the model and integrate constraints into semi-supervised learning, extending [Chang et al., 2007]. In addition, we investigate different training paradigms for CCMs, both for the probabilistic component, and for the constraints component and provide a more rigorous analysis of using constraints in structured prediction tasks. The Never-Ending-Language-Learner (NELL) project provides a web scale experiments [Carlson et al., 2010] for using constraints in semi-supervised learning algorithms. They highlight the importance of decoupling constraints from the model by showing that the constraints can bring significant impact on the performance in the semi-supervised setting.

3.5.1 Captures Long Distance Relationships

In order to make the inference procedure of finding the best assignment tractable, most structured output prediction models only capture local relationships. While CCMs are designed to solve this issue, several approaches (most of them only focus on the supervised learning algorithms) have been proposed in order to take care of the long distance relationships as well. For example, [Collins, 2000; Charniak and Johnson, 2005; Toutanova et al., 2005] propose to use a two stage approach to address this issue: in the first stage, a local model is used to produce the k-best solutions and, in the second stage, a global model which captures long distance relationships is used to rerank the k-best solutions generated in the first stage. Since the global model only focuses on k assignments, modeling long distance relationships becomes tractable. However, this approach suffers from the problem of error propagation. If the k-best solutions produced by the local model does not contain the correct parse tree, it is impossible for the global model to find the correct solution.

Recently, [Daumé and Marcu, 2005; Kazama and Torisawa, 2007; Huang, 2008] proposed to use approximate inference procedure and let the weights of long distance features guide the search procedure. This approach is similar to the beamsearch procedure we proposed in Section 3.1.2, in the sense that the search procedure is guided by the constraint penalties. Importantly, CCMs focus on injecting high level knowledge in the form of "first-order" like features. For example, in this chapter, we show that we can improve HMM very significantly with 10 extra constraint features. In contrast, lots of grounded features are used in [Daumé and Marcu, 2005; Kazama and Torisawa, 2007; Huang, 2008] to capture long distance relationships.

Another line of work that captures long distance relationships is [Finkel et al., 2005], which uses Gibbs sampling as an inference algorithm. The CCM framework is more general, since [Finkel et al., 2005] only

focuses on a specific type of long distance relationship.

3.5.2 Injecting Knowledge into Graphic Models

The combination of constraints and probabilistic graphical models has also been studied before from the probabilistic modeling perspective. For example, [Dechter and Mateescu, 2004] propose a combination of the Bayesian network model with a collection of deterministic constraints and call the resulting model a *mixed network*. They conclude that the deterministic constraints of a mixed network are handled more efficiently when maintained separately from the Bayes network and processed with special purpose algorithms. In addition, they find that the semantics of a mixed network are easier to work with and understand than an equivalent, "pure" Bayes network with deterministic constraints modeled probabilistically. Similar in spirit, CCMs differ from the mixed networks by allowing the probabilistic portion of the model to represent an arbitrary *conditional* distribution, instead of a joint distribution (in the form of a Bayes network).

Markov Logic Networks (MLN) [Richardson and Domingos, 2006] is a probabilistic logic framework which uses logic to provide a convenient way of specifying a Markov Random Field. MLN and CCMs are similar in that they both combine declarative logic into statistical models. The crucial difference between CCM and MLN is on the issue of model decomposition. MLN includes the expressive features as part of the probabilistic model, while we propose factoring the model into a simpler probabilistic model with additional constraints, which can also be specified using first order logic expressions [Rizzolo and Roth, 2007].

3.6 Summary

This chapter provides a unified view of a framework aimed to facilitate decision making with respect to multiple interdependent variables the values of which are determined by learned probabilistic models. We proposed CCMs, a framework that augments linear models with expressive declarative constraints as a way to support decisions in an expressive output space while maintaining modularity and tractability of training. Importantly, this framework provides a principled way to incorporate expressive background knowledge into the decision process. It also provides a way to combine conditional models, learned independently in different situations, along with declarative information to support coherent global decisions.

Chapter 4

Unified Constrained Expectation Maximization

Expectation Maximization (EM) [Dempster et al., 1977] is the most widely used algorithm for unsupervised and semi-supervised learning. Many successful applications of unsupervised and semi-supervised learning in NLP such as text classification [McCallum et al., 1998; Nigam et al., 2000], Part-of-speech tagging [Merialdo, 1994; Das and Petrov, 2011] and parsing [Klein and Manning, 2004] rely on using EM with various generative models.

However, the EM algorithm, as it was originally designed, can be difficult to apply or may lead to sub-optimal performance. Therefore, many variations of EM have been proposed addressing different aspects of EM. For example, hard EM [Neal and Hinton, 1998] addresses the case when calculating the full posterior distribution is expensive, as we have discussed in Section 2.5.1.

In Chapter 3, we proposed **CoDL** (COnstraint-Driven Learning), which uses the constraints as indirect supervision to guide semi-supervised learning. When there are *no labeled examples*, we also mentioned that CoDL is closely related to hard (truncated) EM (Section 3.2.2). Recently, many learning frameworks which take advantage of constraints have been proposed. Among them, the work most related to the CoDL algorithm is **Posterior Regularization** (PR) [Ganchev et al., 2010] [1]. While modeling the exact posterior distribution with hard constraints is expensive in general, PR extends the regular EM algorithm by incorporating expectation constraints. We will review the notation and the semantics of expectation constraints in Section 4.1.

Unfortunately, while there are many different variations of the EM algorithms, little study has been done to understand the relationships between these EM algorithms in the NLP community. For example, people often consider hard EM to be an approximation of EM that should only be used for computational reasons. However, hard EM has its own objective function, and some works have shown that hard EM can sometime be better than the regular EM algorithm [Spitkovsky et al., 2010]. Moreover, even though hard EM is often

[1] Note that there exist other approaches which aim to combine human knowledge with statistical models. For example, [Mann and McCallum, 2008; Bellare et al., 2009] propose **Generalized Expectation Criteria**, which uses a different distance function in the E-step of the EM algorithm. [Liang et al., 2009] proposes **Learning from Measurements**, which incorporates prior information and models posteriors from a Bayesian point of view. For more discussions and comparisons of these approaches, please refer to [Ganchev et al., 2010].

proposed as a worthwhile alternative to an often problematic [Liang and Klein, 2008; Spitkovsky et al., 2010] EM, it is still unclear when to use which algorithm, and if there are better alternatives.

In this chapter, we approach this important issue from a novel perspective. We believe that "EM or Hard-EM?" and "PR or CoDL?" are not the right questions to ask. Instead we present a unified framework for EM called Unified Constrained EM (UEM) that covers most EM variations. UEM allows us to compare and investigate the properties of EM in a systematic way and helps us find better alternatives. For example, UEM helps us find "Linear Programming EM", which has not been formally proposed before to the best of our knowledge.

Several works [Rose et al., 1990; Rose, 1998; Ueda and Nakano, 1998; Smith and Eisner, 2004] have addressed the scenario where EM gets stuck in local maxima and use a parameter controlling the "temperature" as a Deterministic Annealing (DA) technique [2]. Tempered EM (TEM) algorithm [Hofmann, 2001] is a very similar technique to address the same issue. The UEM is related to DA and TEM in the sense that they all use an additional temperature parameter. However, their goals are very different. On one hand, the main goal of UEM is to analyze the relations between different EM algorithms. On the other hand, TEM and DA mainly aim to maximize the EM objective function and use temperature as a technique to avoid local maximum problems. Moreover, these frameworks have largely ignored the use of constraints within EM. The discussions between the temperature-like parameter and the constraints are at the heart of our contributions in this chapter.

The contributions of this chapter are as follows:

- We propose a general framework called Unified Expectation Maximization (UEM) that generalizes a large number of the EM frameworks including constraints-based EM, which allows us to investigate the properties of different EM algorithms in a systematic way and finds better alternative (Section 4.1).
- We discuss the interactions between the temperature-like parameter γ and the constraints. This discussions allows us to put PR and CoDL (the hard constraint version) under the same framework, and also uncovers a potentially important EM-variation, linear programming EM (Section 4.1).
- Through our analysis, we show that it is important to tune γ in order to choose the most appropriate EM algorithm. We show that the choice of γ has a deep connection to the quality of the initialization parameters. Surprisingly, we found that standard EM is often not the best choice even if the initialization point was constructed with few of training examples.
- We show that on a large word-alignment dataset, the UEM framework outperforms the Posterior

[2] For more discussions about the definition and justification of using the temperature parameter, please refer to[Rose, 1998]

Regularization framework [Ganchev et al., 2010] and the (modified) CoDL framework significantly[3] [Chang et al., 2007] (Section 4.4).

4.1 A Unified Expectation Maximization Framework

In this chapter, we present a unified framework, Unified Expectation Maximization (UEM), for covering, uncovering, and relating different variations of the EM algorithm by harvesting the ideas from [Rose et al., 1990; Ueda and Nakano, 1998; Graca et al., 2007] and the CCM model proposed in Chapter 3. First, we review the standard EM algorithm once again. Probabilistic unsupervised learning often aims to estimate parameter θ of $P_\theta(\mathbf{x}, \mathbf{h})$ when only $\mathbf{x}$ is observed and $\mathbf{h}$ remains hidden. To obtain the parameter θ in the unsupervised setting, EM maximizes the log likelihood of the observed data:

$$\mathcal{L}(\theta) = \log P_\theta(\mathbf{x}) = \log \sum_{\mathbf{h} \in \mathcal{H}(\mathbf{x})} P_\theta(\mathbf{x}, \mathbf{h}). \tag{4.1}$$

UEM builds upon the EM framework. One important feature of UEM is the use of constraint. Therefore, before talking about the objective function of UEM, we first review the notation of expectation constraints and then propose the unified EM approach. We borrow the notation of expectation constraints from [Ganchev et al., 2010] whenever possible.

Expectation Constraints We mainly focus on using linear constraints in this thesis. Given m *linear* constraints, assume that the k-th constraint can be written in the following form:

$$u_k(\mathbf{x}, \mathbf{h}) \leq b_k,$$

where $u_k(\mathbf{x}, \mathbf{h})$ is a linear function over $\mathbf{h}$ (it can be a non-linear function over $\mathbf{x}$)[4]. Let us define a posterior distribution q over $\mathcal{H}(\mathbf{x})$, the set of all possible structures. The expectation constraints with respect to q can be expressed by

$$E_q[\mathbf{u}(\mathbf{x}, \mathbf{h})] \leq \mathbf{b}, \tag{4.2}$$

where $\mathbf{u}(\mathbf{x}, \mathbf{h})^T = \begin{bmatrix} u_1(\mathbf{x}, \mathbf{h}) & \dots & u_m(\mathbf{x}, \mathbf{h}) \end{bmatrix}^T$ and $\mathbf{b} = \begin{bmatrix} b_1 & \dots b_m \end{bmatrix}^T$. With these constraints, we introduce the notion of $\mathcal{Q}$ representing the set of feasible posterior distributions with respect to the expectation

[3] Note that in this chapter, we only focus on the case where there is no labeled example for the CoDL framework.

[4] It is beneficial for the readers to review our definition of constraints in Section 2.1. In this chapter, we only focus on linear constraints. That is, we assume that all first-order like constraints have been converted into linear inequalities.

constraints. More precisely,

$$\mathcal{Q} = \{q \mid E_q[\mathbf{u}(\mathbf{x}, \mathbf{h})] \leq \mathbf{b}, q \text{ is a valid probability distribution over } \mathcal{H}(\mathbf{x})\}.$$

Conceptually, $\mathcal{Q}$ is closely related to the set

$$\{\mathbf{h} \mid \mathbf{h} \in \mathcal{H}(\mathbf{x}), \mathbf{u}(\mathbf{x}, \mathbf{h}) \leq \mathbf{b}\},$$

but the former defines a set of all feasible posterior distributions while the latter one defines a set of all feasible structures [5].

UEM Objective Function UEM uses a tuning parameter γ and learns θ as

$$\max_{\theta} T(\theta; \gamma) = \max_{\theta} \left(\mathcal{L}(\theta) - \min_{q \in \mathcal{Q}} D(q(\mathbf{h}), P_\theta(\mathbf{h}|\mathbf{x}); \gamma) \right), \tag{4.3}$$

where $\mathcal{Q}$ is a space of feasible distributions, $D(q(\mathbf{h}), p(\mathbf{h}); \gamma)$ is a modified KL divergence defined by

$$D(q, p; \gamma) = H(q, p) - \gamma H(q), \tag{4.4}$$

and $H(q, p)$ is defined as the cross entropy between distribution q and p. Different values of γ leads to different treatments of the entropy of the posterior. We can consider that UEM allows us to obtain certain desirable results by playing with the entropy of the posterior.

The UEM Algorithm Define $F(\theta, q(\mathbf{h}); \gamma) = \mathcal{L}(\theta) - D(q(\mathbf{h}), P_\theta(\mathbf{h}|\mathbf{x}); \gamma)$. In order to maximize $F(\theta, q(\mathbf{h}); \gamma)$, we modify the version of the EM algorithm given by [Neal and Hinton, 1998] and [Ganchev et al., 2010] in a straightforward way to perform a block coordinate ascent as follows:

1. "E-step": $q^{t+1} = \arg\max_{q \in \mathcal{Q}} F(\theta^t, q; \gamma) = \arg\min_{q \in \mathcal{Q}} D(q, P_{\theta^t}(\mathbf{h}|\mathbf{x}); \gamma)$.
2. "M-step": $\theta^{t+1} = \arg\max_{\theta} F(\theta, q^{t+1}; \gamma) = \arg\max_{\theta} E_{q^{t+1}} [\log P_\theta(\mathbf{x}, \mathbf{h})]$.

[5] For a detailed discussion between the relationships of these two sets, readers can refer to [Sontag, 2010]

It is important to notice that the UEM algorithms contains constraints, given the definition of $\mathcal{Q}$. The M-step comes from the following derivations:

$$\begin{aligned} F(\theta, q(\mathbf{h}); \gamma) =& \mathcal{L}(\theta) - D(q(\mathbf{h}), P_\theta(\mathbf{h}|\mathbf{x}); \gamma) \\ =& \log P_\theta(\mathbf{x}) + \sum_{\mathbf{h} \in \mathcal{H}(\mathbf{x})} q(\mathbf{h}) \log P_\theta(\mathbf{h}|\mathbf{x}) - \gamma \sum_{\mathbf{h} \in \mathcal{H}(\mathbf{x})} q(\mathbf{h}) \log q(\mathbf{h}) \\ =& E_q \left[\log P_\theta(\mathbf{x}, \mathbf{h})\right] - \gamma \sum_{\mathbf{h} \in \mathcal{H}(\mathbf{x})} q(\mathbf{h}) \log q(\mathbf{h}). \end{aligned}$$

Note that γ plays a role only in the E-step, and the M-step is the same as that of the standard EM for all γ. We refer to the UEM algorithm with parameter γ as UEM$_\gamma$. The following theorem shows that the algorithm incrementally improves the objective value for any γ:

Theorem 2. *After the t^{th} iteration of UEM$_\gamma$, $T(\theta^{t+1}; \gamma) \geq T(\theta^t; \gamma)$ for all γ*

Proof. $T(\theta^{t+1}; \gamma) = F(\theta^{t+1}, q^{t+2}; \gamma) \geq F(\theta^{t+1}, q^{t+1}; \gamma) \geq F(\theta^t, q^{t+1}; \gamma) = T(\theta^t; \gamma)$. □

4.2 Relationship between UEM and Other EM Algorithms

The simple parameter γ allows UEM to cover many variations of EM frameworks. In Section 4.4, we show that choosing the right parameter impacts the results of the EM algorithm significantly.

Note that without the presence of the constraints, the relationship between UEM and other EM algorithms has been discussed before [Ueda and Nakano, 1998; Smith and Eisner, 2004]. However, for the sake of completeness, we still include that discussion here. *The novel contribution of this section is that we are the first to list and analyze the role of the temperature when the constraints are present*. In the following, we discuss the relationships between UEM and EM algorithms accordingly.

4.2.1 Without constraints

UEM and EM ($\gamma = 1$) When there is no constraint on $\mathcal{Q}$ (basically, $\mathcal{Q}$ represents all possible posterior distribution of $\mathcal{H}(\mathbf{x})$) and $\gamma = 1$, UEM reduces to EM. We can verify this by setting $\gamma = 1$, $T(\theta; \gamma) = \mathcal{L}(\theta) - \min_{q \in \mathcal{Q}} D(q(\mathbf{h}), P_\theta(\mathbf{h}|\mathbf{x}))$. Moreover, note that since there is no constraint in $\mathcal{Q}$, $P_\theta(\mathbf{h}|\mathbf{x}) \in \mathcal{Q}$ and $\min_{q \in \mathcal{Q}} D(q(\mathbf{h}), P_\theta(\mathbf{h}|\mathbf{x})) = 0$. Therefore, the objective function of UEM reduces to $\mathcal{L}(\theta)$.

UEM and Hard EM ($\gamma = -\infty$) When $\gamma \to -\infty$, the modified KL-divergence becomes a discrete maximization problem of finding the output $\mathbf{h}' \in \mathcal{H}(\mathbf{x})$ and the distribution becomes $\delta_{\mathbf{h}} \in \mathcal{Q}$ where $\delta_{\mathbf{h}'}$ is a

Kronecker-Delta function centered at $\mathbf{h}'$.

To see this, first, observe that for any $q(\mathbf{h})$ with non-zero entropy, $D(q(\mathbf{h}), P_\theta(\mathbf{h}|\mathbf{x}); \gamma) \to \infty$ as $\gamma \to -\infty$. Second, for any $\mathbf{h}'$, $D(\delta_{\mathbf{h}'}, P_\theta(\mathbf{h}|\mathbf{x}); \gamma) = -\log P_\theta(\mathbf{h}'|\mathbf{x})$. These imply that $D(q(\mathbf{h})P_\theta(\mathbf{h}|\mathbf{x}); \gamma)$ is minimized at $q = \delta_{\mathbf{h}=\hat{\mathbf{h}}}$ where $\hat{\mathbf{h}} = \arg\max_{\mathbf{h}\in\mathcal{H}(\mathbf{x})} P_\theta(\mathbf{y}|\mathbf{x})$. Similarly, hard EM algorithm replaces the distribution $q(\mathbf{h})$ in the E-step by just a single $\mathbf{h}$ which maximizes $P_\theta(\mathbf{h}|\mathbf{x})$.

UEM and DA, TEM ($\gamma \in (\infty, 1)$) When $\gamma \to \infty$, the E-step predicts uniform probability for all outputs. This property has been used in to avoid local maximum problems [Rose, 1998; Ueda and Nakano, 1998; Hofmann, 2001] of the standard EM where starting from ∞ (uniform), γ is reduced gradually to 1 (EM). The range of $\gamma \in [0, 1]$ is much less studied in the community.

4.2.2 With constraints

UEM and Posterior Regularization ($\gamma = 1$) Similar to EM, Posterior Regularization framework can be immediately recovered by injecting constraints in $\mathcal{Q}$ and setting γ to 1. However, note that when there are constraints present in $\mathcal{Q}$, the objective function of UEM is not the original log likelihood anymore. The reason is that the $P_\theta(\mathbf{h}|\mathbf{x})$ might not be in the feasible set $\mathcal{Q}$, so the minimal value of $D(q(\mathbf{h}), P_\theta(\mathbf{h}|\mathbf{x}))$ might not be zero.

UEM and CoDL ($\gamma = -\infty$) The UEM framework also covers the CoDL algorithm proposed in Chapter 3. More precisely, it covers the CoDL algorithm with the hard constraints. Recall that when $\gamma \to -\infty$, the posterior distribution needs to become a Kronecker-Delta distribution $q = \delta_{\mathbf{h}=\hat{\mathbf{h}}'}$ center at a structure $\mathbf{h}'$. Combining this fact with the expectation constraints: $E_q[\mathbf{u}(\mathbf{x}, \mathbf{h})] \leq \mathbf{b}$, **the expectation constraints become hard constraints** for $\mathbf{h}'$. The full E-step can then be written as follows:

$$\begin{aligned} \max_{\mathbf{h}'\in\mathcal{H}(\mathbf{x})} \quad & \log P_\theta(\mathbf{h}'|\mathbf{x}) \\ \text{S.T.} \quad & \mathbf{u}(\mathbf{x}, \mathbf{h}') \leq \mathbf{b}, \end{aligned} \tag{4.5}$$

which is an integer programming problem. Note that this equation is exactly the same as Eq. (3.4) with $\rho = -\infty$. Therefore, when there are constraints (written as expectation constraints) and $\gamma = -\infty$, UEM reduces to CoDL .

UEM and Linear Programming EM ($\gamma = 0$) Many publications [Kearns et al., 1998; Smith and Eisner, 2004; Hofmann, 2001] mention that TEM becomes hard EM when $\gamma = 0$. This point is often correct, but it

does not hold in general. When $\gamma = 0$, the entropy term of q vanished, and the E-step can be rewritten as:

$$\begin{aligned} \max_{q} \quad & \textstyle\sum_{\mathbf{h}' \in \mathcal{H}(\mathbf{x})} q_{\mathbf{h}'} \log P_\theta(\mathbf{h}'|\mathbf{x}) \\ \text{S.T.} \quad & E_q[\mathbf{u}(\mathbf{x}, \mathbf{h})] \leq \mathbf{b}, \\ & q_{\mathbf{h}'} \geq 0, \forall \mathbf{h}' \in \mathcal{H}(\mathbf{x}), \\ & \textstyle\sum_{\mathbf{h}' \in \mathcal{H}(\mathbf{x})} q_{\mathbf{h}'} = 1 \end{aligned} \tag{4.6}$$

There are several things worth mentioning in the above formulation. First, we represent the posterior distribution using real-valued variables and add simplex constraints to make q a valid distribution. Second, the $\log P_\theta(\mathbf{h}'|\mathbf{x})$ terms are constant in the E-step. Therefore, Eq. (4.6) is a **linear programming problem**. Importantly, when there are no additional constraints, the simplex constraints form an integral polyhedron. The solution set of the linear programming hence also contains integral solutions, so we can consider the UEM to be a version of hard EM when $\gamma = 0$ and there are no constraints. It is crucial to note that Eq. (4.6) is exactly the *linear relaxation* of Eq. (4.5). Therefore, if finding the integral solution is too expensive in CoDL, we can consider using linear programming with the EM algorithm. This algorithm can be justified by setting γ to 0 in UEM.

While it may seem that this linear programming requires exponential number of variables/constraints to describe $\mathcal{Q}$, for many common probability model, we can decompose the model and rewrite the linear programing with polynomial number of constraints [Sontag, 2010]. We refer to UEM$_{\gamma=0}$ as *LP-EM*. To the best of our knowledge, this is the first time LP-EM is formally introduced.

4.2.3 Summary

Table 4.1 contains a summary of different EM algorithms in the UEM family. We note that some approaches have not been formally proposed before. One such approach is Tempered Constrained EM, which combines expectation constraints and the idea of using temperature to avoid the problem of local maximum. Another example is LP-EM, which is an important member of the UEM family given the recent advances in fast inference algorithms for linear programming problems.

In order to understand this the UEM framework fully, there are two important notes that are worth mentioning, and we discuss them specifically in the following.

Interpretation of γ Beside treating γ as a temperature parameter as in [Ueda and Nakano, 1998; Rose, 1998; Smith and Eisner, 2004], another direct viewpoint is to view γ as a parameter that leads to different

Framework	$\gamma = \infty \rightarrow 1$	$\gamma = 1$	$\gamma = 0$	$\gamma = -\infty$
No constraints	Tempered EM [Hofmann, 2001], Deterministic Annealing EM [Rose et al., 1990]	Standard EM	(NEW) Linear programing EM: Truncated/Hard/Viterbi EM	Truncated/Hard/Viterbi EM
With constraints	(NEW) Tempered Constrained EM	Posterior Regularization [Ganchev et al., 2010]	(NEW) Constrained Linear programing EM	CoDL [Chang et al., 2007]

Table 4.1: The summary of different constrained Expectation Maximization algorithms. The entries marked with "(NEW)" are the frameworks have not been proposed before to the best of our knowledge. Note that Eq. (4.3) is the objective function for all the EM frameworks listed in this table. When there is no constraint and $\gamma = 0$, the linear programming EM can be considered as hard EM.

treatments of the entropy of the posterior. In order words, we can treat γ as an entropy *regularizer* that biases the posterior distribution.

It is both important to notice that in the original deterministic annealing EM framework [Ueda and Nakano, 1998; Smith and Eisner, 2004], people usually do *not* use γ but use the inverse temperature parameter $\beta = \frac{1}{\gamma}$ instead. In our analysis, we point out that using γ directly is often more appropriate, given that the meanings of $\gamma \rightarrow -\infty$ and $\gamma \rightarrow \infty$ are very different. One cannot reveal these differences by using the inverse temperature parameter, given that $\beta \rightarrow 0$ in both cases.

When there are no constraints and $\gamma = 0$ It might be confusing that when there are no constraints and $\gamma = 0$, we list both Hard-EM and Linear programming EM in Table 4.1. This is a result of the "totally unimodular" constraint matrix in the linear programming problem. In (4.6), when there are no user-defined constraints (there are still linear inequalities to make sure the output distribution is valid), one can trivially show that the feasible reason is an integral polyhedron and hence the optimal solution of the LP (when $\gamma = 0$) contains integral solutions [Veinott and Dantzig, 1968; Roth and Yih, 2005]. Hence, in this case, one may consider the LP-EM to be the Hard EM, given that every iteration the E-step can generate integral solution with the linear programming.

4.3 Solving the E-step for UEM for $\gamma \geq 0$

In Section 4.2, we discussed several specific values of γ for UEM, but other values of γ should be considered as different UEM variations. In this section, we discuss how to solve the E-step for general γ. Recall that the E-step solves the following problem:

$$\min_{q \in \mathcal{Q}} D(q(\mathbf{h}), P_{\theta^t}(\mathbf{h}|\mathbf{x}); \gamma). \tag{4.7}$$

When $\gamma < 0$, Eq. (4.7) is non-convex so it is difficult to find its optimal solution. One can apply integer linear programming when $\gamma = -\infty$. In the rest of the section, we focus on the case where $\gamma \geq 0$.

As pointed out by [Smith and Eisner, 2004], a modified version of forward-backward (inside-outside) algorithm can be used to perform EM with temperature. In [Ganchev et al., 2010], they propose a algorithm for solving the E-step in PR when constraints are present. *We present an efficient novel inference algorithm that combines these two techniques. Our algorithm works for all $\gamma > 0$ and is efficient for agreement constraints.*

When $\gamma \geq 0$, Eq. (4.7) is a convex problem. If $\gamma > 0$, the solution of the E-Step can be written as

$$q \propto P_{\theta^t}(\mathbf{h}|\mathbf{x})^{\frac{1}{\gamma}} \exp(-\lambda \cdot \mathbf{f}(\mathbf{x}, \mathbf{h})\mathbf{x}, \mathbf{h}))^{\frac{1}{\gamma}}, \tag{4.8}$$

where λ is a vector of the solution of the dual formulation of Eq. (4.7). The derivation of this formulation appears in Appendix A.1. The vector of λ can be obtained easily by solving the dual formulation of Eq. (4.7), if the constraints and $\mathbf{h}$ can be decomposed in the same way.

When we only have agreement constraints, the E-step can be solved even more efficiently. Agreement constraints are one natural resource of the constraints. For example, a full parsing model and a dependency parsing model should agree to each other [Rush et al., 2010]. Assume that we have two models θ_1 and θ_2 and the output of each model are $\mathbf{h}_1$ and $\mathbf{h}_2$. Let each output $\mathbf{h}_k$ be decomposed into a set of specific variables $\tilde{\mathbf{h}_k}$ and a set of shared variables V. Define $\mathbf{u}(\mathbf{x}, \mathbf{h})$ as a function that extract the shared variables V from $\mathbf{h}$. The E-step with agreement constraints can be written in the following way:

$$\arg\min_{q_1, q_2} \quad D(q_1(\mathbf{h}_1), P_{\theta_1}(\mathbf{h}_1|\mathbf{x}); \gamma) + D(q_2(\mathbf{h}_2), P_{\theta_2}(\mathbf{h}_2|\mathbf{x}); \gamma) \tag{4.9}$$

$$\text{S.T.} \qquad E_{q_1}[\mathbf{u}(\mathbf{x}, \mathbf{h}_1)] = E_{q_2}[\mathbf{u}(\mathbf{x}, \mathbf{h}_2)] \tag{4.10}$$

The Lagrange function of Eq. (4.9) can be written as

$$L(q_1, q_2, \lambda) = D(q_1(\mathbf{h}_1), P_{\theta_1}(\mathbf{h}_1|\mathbf{x}); \gamma) + D(q_2(\mathbf{h}_2), P_{\theta_2}(\mathbf{h}_2|\mathbf{x}); \gamma) - \lambda^T (E_{q_1}[\mathbf{u}(\mathbf{x}, \mathbf{h}_1)] - E_{q_2}[\mathbf{u}(\mathbf{x}, \mathbf{h}_2)])$$

Because the objective of Eq. (4.9) is a convex function, the optimal solution for the dual problem can be converted into a solution for the primal problem. The dual problem of Eq. (4.9) can be written down as

$$\max_{\lambda} \min_{q_1, q_2} L(q_1, q_2, \lambda).$$

Note L is convex with respect to q_1, q_2, and is concave with respect to λ.

Our strategy of solving Eq. (4.9) is to use the subgradient based Lagrange relaxation method, similar to the one used in [Rush et al., 2010]. The main idea of this approach is to treat $\min_{q_1,q_2} L(q_1, q_2, \lambda)$ as a function of λ. Note that Eq. (4.9) is a concave function (given that L is concave with respect to λ, and the minimal of concave functions is still a concave function). However, it is non-differential in general. The subgradient of this function with respect to the current λ_0 is

$$-E_{\hat{q}_1}[\mathbf{u}(\mathbf{x}, \mathbf{h}_1)] + E_{\hat{q}_2}[\mathbf{u}(\mathbf{x}, \mathbf{h}_2)],$$

where $\hat{q}_1$ and $\hat{q}_2$ are the optimal q_1 with the current λ_0, respectively. The value of $\hat{q}_1$ can be obtained by solving

$$\min_{q_1} D(q_1(\mathbf{h}_1), P_{\theta_1}(\mathbf{h}_1|\mathbf{x}); \gamma) - \lambda^T E_{q_1}[\mathbf{u}(\mathbf{x}, \mathbf{h}_1)], \text{ or}$$
$$\min_{q_1} D(q_1(\mathbf{h}_1), P_{\theta_1}(\mathbf{h}_1|\mathbf{x}) \exp^{-\lambda^T \mathbf{u}(\mathbf{x}, \mathbf{h}_1)}; \gamma).$$

Similarly, $\hat{q}_2$ can be obtained by solving

$$\min_{q_2} D(q_2(\mathbf{h}_2), P_{\theta_2}(\mathbf{h}_2|\mathbf{x}); \gamma) + \lambda^T E_{q_2}[\mathbf{u}(\mathbf{x}, \mathbf{h}_2)], \text{ or}$$
$$\min_{q_2} D(q_2(\mathbf{h}_2), P_{\theta_2}(\mathbf{h}_2|\mathbf{x}) \exp^{\lambda^T \mathbf{u}(\mathbf{x}, \mathbf{h}_2)}; \gamma).$$

After obtaining the gradient, the λ can be updated by moving toward to the gradient direction:

$$\lambda \leftarrow \lambda + \eta(-E_{\hat{q}_1}[\mathbf{u}(\mathbf{x}, \mathbf{h}_1)] + E_{\hat{q}_2}[\mathbf{u}(\mathbf{x}, \mathbf{h}_2)]),$$

where η is a step size chosen by the user.

Our final subgradient-based dual decomposition method is listed in Algorithm 9. It finds the optimal solution of Eq. (4.9) without solving the complex constrained optimization problem directly.

Algorithm 9 The Subgradient-based Dual Decomposition Algorithm for Eq. (4.9) for all $\gamma \geq 0$

1: **Input:** Two distributions P_{θ_1} and P_{θ_2}
2: **Output:** Output distributions q_1 and q_2 satisfying 4.10
3: $\lambda \leftarrow 0$
4: **for** N-th iterations **do**
5: $\quad q_1(\mathbf{h}) \leftarrow \arg\min_{q_1} D(q_1(\mathbf{h}), P_{\theta_1}(\mathbf{h}|\mathbf{x}) \exp(-\lambda \cdot f(\mathbf{x}, \mathbf{h})); \gamma)$
6: $\quad q_2(\mathbf{h}) \leftarrow \arg\min_{q_2} D(q_2(\mathbf{h}), P_{\theta_2}(\mathbf{h}|\mathbf{x}) \exp(+\lambda \cdot f(\mathbf{x}, \mathbf{h})); \gamma)$
7: $\quad \lambda \leftarrow \lambda + \eta_N(-E_{q_1}[\mathbf{u}(\mathbf{x}, \mathbf{h})] + E_{q_2}[\mathbf{u}(\mathbf{x}, \mathbf{h})])$
8: **end for**
9: return (q_1, q_2)

Note that this is a more general approach compared to the framework proposed in [Rush et al., 2010], which only focuses on linear programming problems. Assume that our base model is a HMM model. In steps 5 and 6, we perform the modified forward-backward procedure for $\gamma > 0$. For $\gamma \to 0$, it reduces to the case discussed by [Rush et al., 2010] by replacing the forward-backward algorithm by the Viterbi algorithm which finds the highest scoring alignment in steps 5 and 6. For example, when $\gamma > 0$, solution of step 5 is in the form of

$$P_{\theta_1}(\mathbf{h}_1|\mathbf{x})^{\frac{1}{\gamma}} \exp(\frac{\lambda \cdot f(\mathbf{x}, \mathbf{h}_1)}{\gamma}),$$

and when $\gamma = 0$, the solution of step 5 can be obtained by

$$\arg\max_{\mathbf{h}_1 \in \mathcal{H}(\mathbf{x})} P_{\theta_1}(\mathbf{h}_1|\mathbf{x}) \exp(\lambda \cdot f(\mathbf{x}, \mathbf{h}_1)).$$

4.4 Experiments

The main goal of UEM framework is to provide a platform to understand different characteristics of various EM algorithms, and to gain insight into when to choose which algorithm. By varying the value of γ from 1 to 0, we can obtain the EM algorithm (PR with constraints), the Hard EM algorithm (Constrained Linear Programming with constraints) and many other variations ($1 > \gamma > 0$). Comparing EM and hard-EM is an important issue. Recently, [Spitkovsky et al., 2010] showed that Viterbi EM can outperform the EM algorithm by a significant margin. The UEM framework allows us to analyze the EM algorithm in a "continuous" way. We did not explore $\gamma = \infty$ because recent papers [Roth and Yih, 2005; Martins et al., 2009b; Rush et al., 2010; Koo et al., 2010] show that the linear programming results are often quite satisfactory.

In contrast to many works that use TEM or DA, which focus on $\gamma \geq 1$. In this chapter, we mainly focus on the values of γ from 1 to 0. Moreover, we use γ as a tool to analyze and compare versions of EM algorithms, rather than to use it in the annealing setting. Our experiments are designed to answer the following two research questions:

1. What is the best γ to use? EM or Hard EM? What affects the value of the best γ?
2. Given an available development set, how critical it is to select the appropriate EM algorithm? In other words, how crucial it is to select γ in a large-scale experiment?

POS experiments In order to answer the first question, we perform an unsupervised POS learning experiment with the tagging dictionary assumption. We take the standard 24,115 token subset of Penn Treebank. The tagging dictionary is derived from the entire Penn Treebank. We use a first order (bigram) HMM

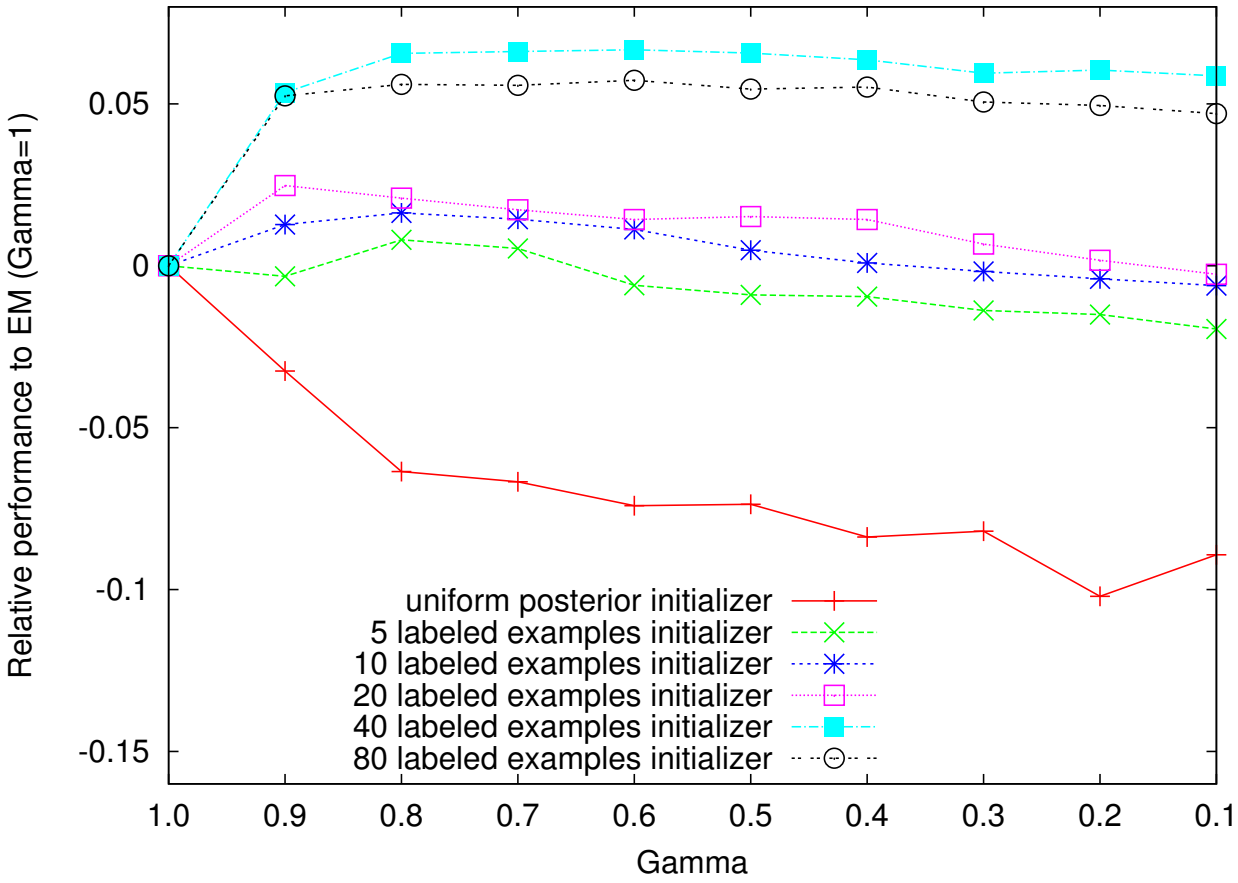


Figure 4.1: POS Experiments on the impact of initialization parameters to the best value of γ. We report the relative performance compared to EM in this figure (see Eq. (4.11)). The number of labeled examples indicates the size of the training set used to create the initial θ for different EM algorithms. The uniform initialization does not use any labeled example. The results show that the value of the best γ is sensitive to the initialization point. Furthermore, EM ($\gamma = 1$) is often not the best choice.

model. Our goal is to analyze the impact of different values of γ and its relationship to different initializations points.

Different initialization points are constructed as follows. The "uniform posterior" initialization is done by spreading the probability over all possible tags for each tokens. In order to construct better initialization points, we train a supervised HMM tagger on a hold-out labeled set. We construct 5 different labeled sets, where the sizes of them are 5, 10 ,20, 40 and 80, respectively. Those initialization points are then fed into the different EM algorithms. For a particular γ, we report its relative performance with respect to EM ($\gamma = 1.0$). More precisely, we report

$$rel(\gamma) = \frac{Acc(UEM_\gamma) - Acc(UEM_{\gamma=1})}{Acc(UEM_{\gamma=1})}, \tag{4.11}$$

Data Size	En-Fr			Fr-En		
	PR	LP-EM	$UEM_{0.3}$	PR	LP-EM	$UEM_{0.3}$
10k	13.78	15.01	11.02	13.22	15.70	10.92
50k	11.26	11.03	9.08	10.72	10.13	9.43
100k	10.59	9.19	8.22	10.33	9.68	8.76

Table 4.2: AER comparisons on Hansard Corpus for French-English alignment for various data sizes

where Acc represents the accuracy evaluated on the ambiguous words of the 24,115 set. Note that $rel(\gamma = 1) = 0$. When $rel(\gamma) > 0$, it means that UEM$_\gamma$ performs better than EM algorithm, and vice versa.

The results summarized are in Figure 4.1. Note that when we use the "uniform" initialization, EM is the clear winner by a significant margin. Surprisingly, with the initialization point is obtained by merely 5 or 10 examples, EM is not the best algorithm anymore. Interestingly, we find that setting $\gamma = 0.9$ works fairly well across all initialization points beside "uniform." Moreover, when the initialization point becomes better, smaller γ performs better. The results indicate that **the best value of γ is closely related to the quality of the initialization point**.

This experiment agrees with [Merialdo, 1994], where the authors show that EM stops working in the semi-supervised setting. We further extend this result by showing this issue can be overcome by just setting γ to a smaller value. In [Spitkovsky et al., 2010], the authors show that Hard EM (Viterbi EM) can be better than the standard EM. We suspect that in their setting, a γ that is a little bit smaller than 1 can already beat EM.

Word Alignment Experiments In order to answer the second question, we present experimental results on statistical word alignment which is a well known application of unsupervised learning and is a key step towards machine translation from a source language S to a target language T. We borrow the set-up from [Ganchev et al., 2008] (also [Graca et al., 2007; Ganchev et al., 2010][6]). Note that their setting includes two language experiments and each of them comes with a small development set. [Ganchev et al., 2008] shows significant improvements in alignment error rate (AER) by introducing constraints over posterior probabilities of the alignments to guide their EM algorithm. In particular, we consider the agreement constraints which direct alignment probabilities in one direction (P_{θ_1}: from S to T) to agree with the alignment in the other direction (P_{θ_2}: from T to S) and apply Algorithm 9. Following [Graca et al., 2007], we use the same testing method called posterior decoding for all of the models [Liang et al., 2006], and do not apply constraint at test time.

We test our approach on the Hansards corpus for French-English translation [Och and Ney, 2000] and Europarl corpus [Koehn, 2002] for Spanish-English translation with EPPS [Lambert et al., 2006] alignment

[6] http://www.seas.upenn.edu/~strctlrn/CAT/CAT.html

Data Size	En-Es			Es-En		
	PR	LP-EM	$UEM_{0.3}$	PR	LP-EM	$UEM_{0.3}$
10k	26.78	29.57	24.33	26.09	28.57	23.23
50k	24.11	24.65	22.18	23.23	23.35	21.01
100k	23.26	23.31	21.81	22.55	22.20	20.25

Table 4.3: AER comparisons on EPPS for Spanish-English alignment for various data sizes

annotation. We select the best $\gamma = 0.3$ on the development set and used it to build all of our models.

We report the AER for different algorithms and different sizes of unlabeled data. We focus on comparing $\gamma = 1.0$ (EM/PR) and $\gamma = 0.0$ (LP-EM) with $\gamma = 0.3$. Note that $\gamma = 1$ is the same as the PR approach of [Ganchev et al., 2008]. Note that in this setting LP-EM is fairly close to the CoDL because the dual decomposition design make Algorithm 9 always returns integral solutions[7]. $UEM_{0.3}$ shows significant improvement over both PR and LP-EM.

4.5 Summary

In this chapter, we present a unified framework for Expectation Maximization (EM) algorithms parametrized by a temperature-like parameter. While introducing a temperature parameter has been considered in the literature as a technique to avoid the local maximum problem, we further analyze the role of temperature with constraints. We demonstrate the importance of this unified framework by showing that in many cases, EM (or PR when constraints are used) is often not the best choice, and claim that the temperature parameter is sensitive to the initialization point. The proposed unified constrained EM framework shows significant improvement over PR and LP-EM on the word alignment problems.

[7] In some rare cases, the agreement constraints cannot be satisfied. See [Rush et al., 2010] for more discussions.

Chapter 5

Learning Constrained Intermediate Representations

Many NLP tasks can be phrased as decision problems over complex linguistic structures. Successful learning depends on correctly encoding these (often latent) structures as features for the learning system. Tasks such as transliteration discovery [Klementiev and Roth, 2008], recognizing textual entailment (RTE) [Dagan et al., 2006] and paraphrase identification [Dolan et al., 2004] are a few prototypical examples. However, the input to such problems does not specify the latent structures and the problem is defined in terms of surface forms only. Most current solutions transform the raw input into a meaningful intermediate representation[1], and then encode its structural properties as features for the learning algorithm.

Note that unlike the problems we discussed in Chapter 3 and 4, these problems are binary classification tasks rather than structured output prediction tasks. Nevertheless, there still exists a notion of *structure* for these binary tasks: the requirement of finding a "good" intermediate representation. Since the intermediate representations are not labeled in the training data, they can be considered as *latent structures* here.

Consider the RTE task of identifying whether the meaning of a short text snippet (called the *hypothesis*) can be inferred from that of another snippet (called the *text*). A common solution [MacCartney et al., 2008; Roth et al., 2009] is to begin by defining an alignment over the corresponding entities, predicates and their arguments as an intermediate representation. A classifier is then trained using features extracted from the intermediate representation. The idea of using an intermediate representation also occurs frequently in other NLP tasks [Bergsma and Kondrak, 2007; Qiu et al., 2006].

While the importance of finding a good intermediate representation is clear, emphasis is typically placed on the later stage of extracting features over this intermediate representation, thus separating learning into **two stages** – specifying the intermediate representation, and then extracting features for learning. The intermediate representation is obtained by an inference process using predefined models or well-designed heuristics. While these approaches often perform well, they ignore a useful resource when generating the latent structure – the labeled data for the final learning task. As we will show in this section, this results in degraded performance for the actual classification task at hand. Several works have considered this issue

[1] In this section, the phrases "intermediate representation" and "latent representation" are used interchangeably.

[McCallum et al., 2005; Goldwasser and Roth, 2008; Chang et al., 2009; Das and Smith, 2009]; however, they provide solutions that do not easily generalize to new tasks.

In [Chang et al., 2010a], we propose a unified solution to the problem of learning to make the classification decision **jointly** with determining the intermediate representation. Our *Learning Constrained Latent Representations (***LCLR***) framework* is guided by the intuition that there is *no* intrinsically good intermediate representation, but rather that a representation is good only to the extent to which it improves performance on the final classification task. In the rest of this section we discuss the properties of our framework and highlight its contributions.

Learning over Latent Representations This chapter formulates the problem of *learning over latent representations* and presents a novel and general solution applicable to a wide range of NLP applications. We analyze the properties of our learning solution, thus allowing new research to take advantage of a well understood learning and optimization framework rather than an ad-hoc solution. We show the generality of our framework by successfully applying it to three domains: transliteration, RTE and paraphrase identification.

Joint Learning Algorithm In contrast to most existing approaches that employ domain specific heuristics to construct intermediate representations to learn the final classifier, our algorithm learns to construct the optimal intermediate representation to support the learning problem. Learning to represent is a difficult structured learning problem however, unlike other works that use labeled data at the intermediate level, our algorithm only uses the binary supervision supplied for the final learning problem.

Flexible Inference Successful learning depends on constraining the intermediate representation with task-specific knowledge. Our framework uses the declarative Integer Linear Programming (ILP) inference formulation, which makes it easy to define the intermediate representation and to inject knowledge in the form of *constraints*. While ILP has been applied to structured output learning, to the best of our knowledge, this is the first work that makes use of ILP in formalizing the general problem of learning intermediate representations.

5.1 Preliminaries

We introduce notation using the Paraphrase Identification task as a running example. This is the binary classification task of identifying whether one sentence is a paraphrase of the other. A paraphrase pair from

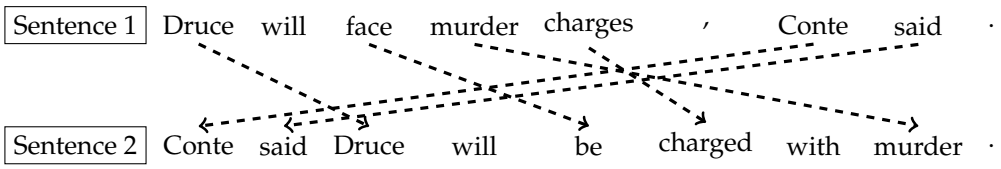


Figure 5.1: **A binary classification problem that needs latent structures: Paraphrase Identification problem.** The task is to identify whether one sentence is a paraphrase of the other. The dotted lines represent a possible intermediate representation for the paraphrase identification task. Since different representation choices will impact the binary identification decision directly, our approach chooses the representation that best facilitates the binary learning task.

the MSR Paraphrase corpus [Quirk et al., 2004] is shown in Figure 5.1. In order to identify that the sentences paraphrase each other, we need to align constituents of these sentences. One possible alignment is shown in the figure, in which the dotted edges correspond to the aligned constituents. An alignment can be specified using binary variables corresponding to every edge between constituents, indicating whether the edge is included in the alignment. Different activations of these variables induce the space of *intermediate representations.*

To formalize this setting, let $\mathbf{x}$ denote the input to a decision function, which maps $\mathbf{x}$ to $z \in \{-1, 1\}$. We consider problems where this decision depends on an intermediate representation (for example, the collection of all dotted edges in Figure 5.1, which can be represented by a binary vector $\mathbf{h}$.

In the literature, a common approach is to separate the problem into two stages. First, a generation stage predicts $\mathbf{h}$ for each $\mathbf{x}$ using a pre-defined model or a heuristic. This is followed by a learning stage, in which the classifier is trained using $\mathbf{h}$. In our example, if the generation stage predicts the alignment shown, then the learning stage would use the features computed based on the alignments. Formally, the two-stage approach uses a pre-defined inference procedure that finds an intermediate representation $\mathbf{h}'$. Using features $\Phi(\mathbf{x}, \mathbf{h}')$ and a learned weight vector $\mathbf{w}'$, the example is classified as positive if $\mathbf{w}'^T \Phi(\mathbf{x}, \mathbf{h}') \geq 0$.

However, in the two stage approach, the latent representation, which is provided to the learning algorithm, is determined before learning starts, and without any feedback from the final task. It is dictated by the intuition of the developer. This approach makes two implicit assumptions: first, it assumes the existence of a "correct" latent representation and, second, that the model or heuristic used to generate it is the correct one for the learning problem at hand.

5.2 Joint Learning with an Intermediate Representation

In contrast to two-stage approaches, we use the annotated data for the final classification task to learn a suitable intermediate representation which, in turn, helps the final classification.

Choosing a good representation is an optimization problem that selects which of the elements (features) of the representation best contribute to successful classification given some legitimacy constraints; therefore, we (1) set up the optimization framework that finds legitimate representations (Section 5.2.1), and (2) learn an objective function for this optimization problem, such that it makes the best final decision (Section 5.2.1.)

5.2.1 Inference

Our goal is to correctly predict the final label rather than matching a "gold" intermediate representation. In our framework, attempting to learn the final decision drives both the selection of the intermediate representation and the final prediction.

For each $\mathbf{x}$, let $\Gamma(\mathbf{x})$ be the set of all substructures of the intermediate representation space. In Figure 5.1, this could be the set of all alignment edges connecting the constituents of the sentences. Given a vocabulary of such structures of size $N = |\Gamma(\mathbf{x})|$, we denote intermediate representations by $\mathbf{h} \in \{0,1\}^N$, which "selects" the components of the vocabulary that constitute the intermediate representation.

For example, in our paraphrasing example, $\Gamma(\mathbf{x})$ would be the alignment edges, pairs of edges, triples, and so on. We associate every element $s \in \Gamma(\mathbf{x})$ with a binary hidden variable h_s, each of which can be thought of as a "switch" that controls whether the corresponding substructure s is included in the predicted intermediate representation. Assume that there are total m features. We define $\Phi_s(\mathbf{x}) \in R^n$ to be a feature vector over the substructure s, which is used to describe the characteristics of s, and define a weight vector $\mathbf{w} \in R^n$ over these features.

Let $\mathcal{H}$ denote the set of feasible intermediate representations $\mathbf{h}$, specified by means of linear constraints over $\mathbf{h}$. While $\Gamma(\mathbf{x})$ might be large, the set of those elements in $\mathbf{h}$ that are active can be constrained by controlling $\mathcal{H}$. After we have learned a weight vector $\mathbf{w}$ that scores intermediate representations for the final classification task, we define our decision function as

$$f_{\mathbf{w}}(\mathbf{x}) = \max_{\mathbf{h} \in \mathcal{H}} \mathbf{w}^T \Phi(\mathbf{x}, \mathbf{h}) = \max_{\mathbf{h} \in \mathcal{H}} \mathbf{w}^T \sum_{s \in \Gamma(\mathbf{x})} h_s \Phi_s(\mathbf{x}), \tag{5.1}$$

and classify the input as positive if $f_{\mathbf{w}}(\mathbf{x}) \geq 0$.

In Eq. (5.1), $\mathbf{w}^T \Phi_s(\mathbf{x})$ is the score associated with the substructure s, and $f_{\mathbf{w}}(\mathbf{x})$ is the score for the entire

intermediate representation. Therefore, our decision function $f_{\mathbf{w}}(\mathbf{x}) \geq 0$ makes use of the intermediate representation and its score to classify the input. An input is labeled as positive if some underlying intermediate structure allows it to cross the decision threshold. The intermediate representation is chosen to maximize the overall score of the input. This design is especially beneficial for many phenomena in NLP, where only positive examples have a meaningful underlying structure. In our paraphrase identification example, good alignments generally exist only for positive examples.

One unique feature of our framework is that we treat Eq. (5.1) as an Integer Linear Programming (ILP) instance. A concrete instantiation of this setting to the paraphrase identification problem, along with the actual ILP formulation is shown in Section 5.3.

Learning

We now present an algorithm that learns the weight vector $\mathbf{w}$. For a loss function $\ell : \mathbb{R} \rightarrow \mathbb{R}$ and the labeled set $\{(\mathbf{x}_i, z_i)\}$, the goal of learning is to solve the following optimization problem:

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i L_B(\mathbf{x}_i, \mathbf{y}_i, \mathbf{w}) = \min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \ell\left(1 - z_i f_{\mathbf{w}}(\mathbf{x}_i)\right), \tag{5.2}$$

where the function $\ell : \mathbb{R} \rightarrow \mathbb{R}$ can be instantiated by many commonly used loss functions such as hinge loss, with $\ell(a) = \max(0, a)$, and squared-hinge loss, with $\ell(a) = \max(0, a)^2$. Note that we use L_B to represent the loss for binary labeled examples.

Here, C is the regularization parameter. Substituting Eq. (5.1) into Eq. (5.2), we get

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \ell\left(1 - z_i C \max_{\mathbf{h}\in\mathcal{H}} \mathbf{w}^T \sum_{s\in\Gamma(\mathbf{x})} h_s \Phi_s(\mathbf{x}_i)\right) \tag{5.3}$$

Note that there is a maximization term inside the global minimization problem, making Eq. (5.3) a non-convex problem. The minimization drives $\mathbf{w}$ towards smaller empirical loss while the maximization uses $\mathbf{w}$ to find the best representation for each example.

The algorithm for Learning over Constrained Latent Representations (LCLR) is listed in Algorithm 10. In each iteration, first, we find the best feature representations for all positive examples (lines 3-5). This step can be solved with an off-the-shelf ILP solver. Having fixed the representations for the positive examples, we update the $\mathbf{w}$ by solving Eq. (5.4) at line 6 in the algorithm. It is important to observe that for positive examples in Eq. (5.4), we use the intermediate representations $\mathbf{h}^*$ from line 4. One reason that we fix the hidden representation on only positive examples is because we can solve the non-convex objective function

Algorithm 10 *LCLR*: The optimization procedure that optimizes (5.3). Note that $z_i \in \{-1, 1\}$ represents the gold standard of the binary output problem.

1: initialize: $\mathbf{w} \leftarrow \mathbf{w}_0$
2: **repeat**
3: **for all** positive examples $(\mathbf{x}_i, z_i = 1)$ **do**
4: Find $\mathbf{h}_i^* \leftarrow \arg\max_{\mathbf{h} \in \mathcal{H}} \sum_s h_s \mathbf{w}^T \Phi_s(\mathbf{x}_i)$
5: **end for**
6: Update $\mathbf{w}$ by solving

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i:z_i=1} \ell(1 - \mathbf{w}^T \sum_s h_{i,s}^* \Phi_s(\mathbf{x}_i)) + C \sum_{i:z_i=-1} \ell(1 + \max_{\mathbf{h} \in \mathcal{H}} \mathbf{w}^T \sum_s h_s \Phi_s(\mathbf{x}_i)) \tag{5.4}$$

7: **until** convergence
8: return $\mathbf{w}$

Eq. (5.3) by solving a series of convex problems (Eq. (5.4)) at line 6 in the Algorithm 10). Theoretically, we can use any loss function that is convex. In the experiments in this section, we use the squared-hinge loss function: $\ell(1 - z f_{\mathbf{w}}(\mathbf{x})) = \max(0, 1 - z f_{\mathbf{w}}(\mathbf{x}))^2$.

It is important to note that the algorithm is asymmetric. It treats positive examples and negative examples differently. The reason of this is from the fact that the semantics of a positive example and a negative example are different – *only* a positive example requires a good intermediate representation to justify its positive label and a negative example cannot have a good intermediate representation. This asymmetric property in Algorithm 10, which we first find the representation for positive examples, is important. Without this asymmetric property, we will lose the guarantee that the objective function decreases monotonically in every iteration of the Algorithm 10.

Recall that Eq. (5.4) in line 10 of Algorithm 10 is *not* the traditional SVM or logistic regression formulation. This is because inside the inner loop, the best representation for each negative example must be found. Therefore, we need to perform inference for every negative example when updating the weight vector solution.

Note that our intuition on treating positive examples and negative examples differently also reflects on the optimization algorithm (Algorithm 10). Because Eq. (5.3) is a non-convex problem, following [Felzenszwalb et al., 2009], LCLR iteratively solves a series of easier problems (Eq. (5.4)). This is especially true for our loss function because Eq. (5.4) is convex and can be solved efficiently. Interestingly, the positive example are the reasons why this formulation is not convex. Hence, by fixing the representations for the positive examples, the inner problem becomes a convex problem.

We use a cutting plane algorithm to solve Eq. (5.4). A similar idea has been proposed in [Joachims et al., 2009]. The algorithm for solving Eq. (5.4) is presented as Algorithm 11. This algorithm uses a "cache" H_j to store all intermediate representations for negative examples that have been seen in previous iterations

Algorithm 11 Cutting plane algorithm to optimize Eq. (5.4)

1: for each negative example x_j, $H_j \leftarrow \emptyset$
2: **repeat**
3: **for** each negative example x_j **do**
4: Find $\mathbf{h}_j^* \leftarrow \arg\max_{\mathbf{h}\in\mathcal{C}} \sum_s h_s \mathbf{w}^T \phi_s(\mathbf{x}_j)$
5: $H_j \leftarrow H_j \cup \{\mathbf{h}_j^*\}$
6: **end for**
7: Solve

$$\min_{\mathbf{w}} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \sum_{i:y_i=1} \ell(-\mathbf{w}^T \sum_s h_{i,s}^* \phi_s(\mathbf{x}_i)) + \sum_{i:y_i=-1} \ell(\max_{\mathbf{h}\in H_i} \mathbf{w}^T \sum_s h_s \phi_s(\mathbf{x}_i)) \quad (5.5)$$

8: **until** no new element is added to any H_j
9: return $\mathbf{w}$

(lines 3-6) [2]. The difference between Eq. (5.5) in line 7 of Algorithm 11 and Eq. (5.4) is that in Eq. (5.5), we do not search over the entire space of intermediate representations. The search space for the minimization problem Eq. (5.5) is restricted to the cache H_j. Therefore, instead of solving the minimization problem Eq. (5.4), we can now solve several simpler problems shown in Eq. (5.5). The algorithm is guaranteed to stop (line 8) because the space of intermediate representations is finite. Furthermore, in practice, the algorithm needs to consider only a small subset of "hard" examples before it converges. The justification of the algorithm can be seen in Appendix A.2.

Inspired by [Hsieh et al., 2008], we apply an efficient coordinate descent algorithm for the *dual* formulation of (Eq. (5.5)) which is guaranteed to find its global minimum. The algorithm is extended and generalized in Chapter 6. For more detail discussions, please refer to Section 6.2.

5.3 Encoding with ILP: A Paraphrase Identification Example

In this section, we define the intermediate representation for the paraphrase identification task. Unlike the earlier example, where we considered the alignment of lexical items, we describe a more complex intermediate representation by aligning graphs created using semantic resources.

An input example is represented as two acyclic graphs, G_1 and G_2, corresponding to the first and second input sentences. Each vertex in the graph contains word information (lemma and part-of-speech) and the edges denote dependency relations, generated by the Stanford parser [Klein and Manning, 2003]. The intermediate representation for this task can now be defined as an alignment between the graphs, which

[2] In our implementation, we keep a global cache H_j for each negative example x_j. Therefore, in Algorithm 11, we start with a non-empty cache improving the speed significantly.

captures lexical and syntactic correlations between the sentences.

We use $V(G)$ and $E(G)$ to denote the set of vertices and edges in G respectively, and define four hidden variable types to encode vertex and edge mappings between G_1 and G_2.

- The **word-mapping** variables, denoted by h_{v_1,v_2}, define possible pairings of vertices, where $v_1 \in V(G_1)$ and $v_2 \in V(G_2)$.
- The **edge-mapping** variables, denoted by h_{e_1,e_2}, define possible pairings of the graphs edges, where $e_1 \in E(G_1)$ and $e_2 \in E(G_2)$.
- The **word-deletion** variables $h_{v_1,*}$ (or h_{*,v_2}) allow for vertices $v_1 \in V(G_1)$ (or $v_2 \in V(G_2)$) to be deleted. This accounts for omission of words (like function words).
- The **edge-deletion** variables, $h_{e_1,*}$ (or h_{*,e_2}) allow for deletion of edges from G_1 (or G_2).

Our inference problem is to find the optimal set of hidden variable activations, restricted according to the following set of linear constraints

- Each vertex in G_1 (or G_2) can either be mapped to a single vertex in G_2 (or G_1) or marked as deleted. In terms of the **word-mapping** and **word-deletion** variables, we have

$$\forall v_1 \in V(G_1), \sum_{v_2 \in V(G_2)} h_{v_1,v_2} + h_{v_1,*} = 1, \quad \forall v_2 \in V(G_2), \sum_{v_1 \in V(G_1)} h_{v_1,v_2} + h_{*,v_2} = 1 \tag{5.6}$$

- Each edge in G_1 (or G_2) can either be mapped to a single edge in G_2 (or G_1) or marked as deleted. In terms of the **edge-mapping** and **edge-deletion** variables, we have

$$\forall e_1 \in E(G_1), \sum_{e_2 \in E(G_2)} h_{e_1,e_2} + h_{e_1,*} = 1, \quad \forall e_2 \in E(G_2), \sum_{e_1 \in E(G_1)} h_{e_1,e_2} + h_{*,e_2} = 1 \tag{5.7}$$

- The edge mappings can be active if, and only if, the corresponding node mappings are active. Suppose $e_1 = (v_1, v_1') \in E(G_1)$ and $e_2 = (v_2, v_2') \in E(G_2)$, where $v_1, v_1' \in V(G_1)$ and $v_2, v_2' \in V(G_2)$. Then, we have

$$h_{v_1,v_2} + h_{v_1',v_2'} - h_{e_1,e_2} \leq 1, \quad h_{v_1,v_2} \geq h_{e_1,e_2}, \quad h_{v_1',v_2'} \geq h_{e_1,e_2} \tag{5.8}$$

These constraints define the feasible set for the inference problem specified in Eq. (5.1). This inference problem can be formulated as an ILP problem with the objective function from Eq. (5.1):

$$\max_{\mathbf{h}} \quad \sum_{s} h_s \mathbf{w}^T \Phi_s(\mathbf{x}) \tag{5.9}$$

$$\text{subject to} \quad \text{Eq. (5.6)-Eq. (5.8)}; \;\; \forall s; h_s \in \{0,1\} \tag{5.10}$$

This example demonstrates a principled approach that uses the ILP formulation to define intermediate representations incorporating linguistic intuitions.

5.4 Experiments

We applied our framework to three different NLP tasks: transliteration discovery [Klementiev and Roth, 2008], RTE [Dagan et al., 2006], and paraphrase identification [Dolan et al., 2004].

Our experiments are designed to answer the following research question: "given a binary classification problem defined over latent representations, will the joint LCLR algorithm perform better than a two-stage approach?" To ensure a fair comparison, both systems use the same feature functions and the same definition of intermediate representation. We use the same ILP formulation in both configurations, with a single exception – the objective function parameters: the two stage approach uses a task-specific heuristic, while LCLR learns it iteratively.

As we mentioned about, previously proposed two staged methods often use task-specific inference procedures. On the other hand, our constraint-based approach allows us to incorporate linguistic intuition very easily. Hence, our ILP formulation results in very strong two stage systems. For example, in the paraphrase identification task, even our two stage system achieves the current state-of-the-art performance. In these settings, the improvement obtained by our joint approach is non-trivial and can be clearly attributed to the superiority of the joint learning algorithm. Interestingly, we find that our more general approach is better or competitive to other joint approaches [Goldwasser and Roth, 2008; Das and Smith, 2009].

We want to stress that both LCLR and our two stage approach benefit from the flexible ILP formulation, which was adapted and optimized for the individual tasks. In the following, we refer to our two stage approach as "Alignment+Learning". The whole procedure is as follows:

1. Use domain-dependent heuristic to create a weight vector $\mathbf{w}_0$. The same weight vector is also used to initialize LCLR.

2. For each example $\mathbf{x}$, solve an integer linear programming problem using $\mathbf{w}_0$ to find the best alignment

$\mathbf{h}$ (hence the best constrained representation $\Phi(\mathbf{x}, \mathbf{h})$). Find the best representations for all examples in both training and testing data.

3. Train a support vector machine (square-hinge loss) using the presentations found in the previous step.

4. Test the weight vector found in the step 3 with the representation found by $\mathbf{w}_0$.

Since the objective function (5.3) of the joint approach is not convex, a good initialization is required. We use the weight vector learned by the two stage approach as the starting point for the joint approach. The algorithm terminates when the relative improvement of the objective is smaller than 10^{-5}.

5.4.1 Transliteration Discovery

Transliteration discovery is the problem of identifying if a word pair, possibly written using two different uni-character sets, refers to the same underlying entity. The intermediate representation consists of all possible character mappings between the two character sets. Identifying this mapping is not easy, as most writing systems do not perfectly align phonetically and orthographically; rather, this mapping can be context-dependent and ambiguous.

For an input pair of words (w_1, w_2), the intermediate structure $\mathbf{h}$ is a mapping between their characters, with the latent variable h_{ij} indicating if the i^{th} character in w_1 is aligned to the j^{th} character in w_2. The feature vector associated with the variable h_{ij} contains unigram character mapping, bigram character mapping (by considering surrounding characters). We adopt the *one-to-one mapping* and *non-crossing* constraint used in [Chang et al., 2009]. The detailed feature vector description is in Table 5.1.

Category	Feature
Bias	a dummy feature that is active for every substructure
Distance	the distance between two characters, the value of the feature is $1.0/(\|i-j\|+1)$
Unigram	the conjunction of the i^{th} character in w_1 and the j^{th} character of w_2
Bigram	the conjunction of the i^{th}, $(i+1)^{th}$ characters in w_1 and the j^{th} character of w_2
	the conjunction of the i^{th} character in w_1 and the j^{th}, $(j+1)^{th}$ characters of w_2

Table 5.1: The features used in the transliteration experiments. We describe the features for the "substructure" h_{ij}, which is associated with the alignment between the i^{th} character of w_1 and the j^{th} character of w_2.

We evaluated our system using the English-Hebrew corpus [Goldwasser and Roth, 2008], which consists of 250 positive transliteration pairs for training, and 300 pairs for testing. As negative examples for training, we sample 10% from random pairings of words from the positive data. We report two evaluation measures – (1) the Mean Reciprocal Rank (MRR), which is the average of the multiplicative inverse of the rank of the correct answer, and (2) the accuracy (Acc), which is the percentage of the top rank candidates being correct.

Transliteration System	Acc	MRR
[Goldwasser and Roth, 2008]	N/A	89.4
Alignment + Learning	80.0	85.7
LCLR	**92.3**	**95.4**

(a) Transliteration System Results

Entailment System	Acc
Median of TAC 2009 systems	61.5
Alignment + Learning	65.0
LCLR	**66.8**

(b) Entailment System Results

Figure 5.2: (**Left (a):**) Experimental results for transliteration. We compare a *two-stage* system (Alignment+Learning) with LCLR, our *joint* algorithm. Both Alignment+Learning and LCLR use the same features and the same intermediate representation definition. (**Right (b):**) Experimental results for recognizing textual entailment. The first row is the median of best performing systems of all teams that participated in the RTE5 challenge [Bentivogli et al., 2009]. *Alignment + Learning* is our two-stage system implementation. Details about these systems are provided in the text. Note that LCLR outperforms [Das and Smith, 2009], which is a joint approach based on quasi-synchronous dependency grammars.

We initialized the two stage inference process as detailed in [Chang et al., 2009] using a Romanization table to assign uniform weights to prominent character mappings. This initialization procedure resembles the approach used in [Bergsma and Kondrak, 2007]. An alignment is first built by solving the constrained optimization problem. Then, a support vector machine with squared-hinge loss function is used to train a classifier using features extracted from the alignment. We refer to our two stage approach as *Alignment+Learning*.

The results, summarized in Figure 5.2(a), show the significant improvement obtained by the joint approach (95.4 MRR) compared to the two stage approach (85.7). Moreover, our results are also significantly better than the joint model published in [Goldwasser and Roth, 2008], which achieve 89.4 MRR. In [Pasternack and Roth, 2009], the authors show better results on this dataset with more complex definition of the structure.

Note that in Chapter 6, we will revisit a variation of this problem. However, instead of focusing on telling whether two named entities are transliterations of each or not, we want to build the phonetic alignment between these two named entities. Please see Section 6.5.5 for more discussions and comparisons.

5.4.2 Textual Entailment

Recognizing Textual Entailment (RTE) is an important textual inference task of predicting if a given *text* snippet entails the meaning of another (the *hypothesis*). In many current RTE systems, the entailment decision depends on successfully aligning the constituents of the text and hypothesis, accounting for the internal linguistic structure of the input. Please refer to [Chang et al., 2010a] for the details of the feature design and the hidden variable definitions.

The raw input – the text and hypothesis – are represented as directed acyclic graphs, where vertices correspond to words. Directed edges link verbs to the head words of semantic role labeling arguments pro-

Hidden Variable	RTE features	Paraphrase features
word-mapping	WordNet, POS, Coref, Neg	WordNet, POS, NE, ED
word-deletion	POS	POS, NE
edge-mapping	NODE-INFO	NODE-INFO, DEP
edge-deletion	N/A	DEP

Table 5.2: Summary of latent variables and feature resources for the entailment and paraphrase identification tasks. See Section 5.3 for an explanation of the hidden variable types. The linguistic resources used to generate features are abbreviated as follows – POS: Part of speech, Coref: Canonical coreferent entities; NE: Named Entity, ED: Edit distance, Neg: Negation markers, DEP: Dependency labels, NODE-INFO: node alignment resources, N/A: Hidden variable not used.

duced by [Punyakanok et al., 2008]. All other words are connected by dependency edges. The intermediate representation is an alignment between the nodes and edges of the graphs. We used three hidden variable types from Section 5.3 – **word-mapping**, **word-deletion** and **edge-mapping**, along with the associated constraints as defined earlier.

The second column of Figure 5.2 lists the resources used to generate features corresponding to each hidden variable type. For word-mapping variables, the features include a WordNet based metric, indicators for the POS tags and negation identifiers. We used a state-of-the-art coreference resolution system to identify the canonical entities for pronouns and extract features accordingly. For word deletion, we use only the POS tags of the corresponding tokens to generate features. For edge mapping variables, we include the features of the corresponding word mapping variables, scaled by the word similarity of the words comprising the edge.

We evaluated our system using the RTE-5 challenge data [Bentivogli et al., 2009], consisting of 600 sentence pairs for training and testing respectively, in which positive and negative examples are equally distributed. In these experiments the joint LCLR algorithm converged in about 5 iterations. For the two stage system, we used a Wordnet similarity score (WNSim) to score alignments during inference. The word-based scores influence the edge variables via the constraints. The experimental results are in Figure 5.2(b). This two-stage system (the Alignment+Learning system) is significantly better than the median performance of the RTE-5 submissions. Using LCLR further improves the result by almost 2%, a substantial improvement in this domain.

5.4.3 Paraphrase Identification

Our final task is Paraphrase Identification, discussed in detail in Section 5.3. We use all the four hidden variable types described in that section. Again, for the feature design, please see [Chang et al., 2010a].

We used the MSR paraphrase dataset of [Dolan et al., 2004] for empirical evaluation. Additionally,

Paraphrase System	Acc
Experiments using [Dolan et al., 2004]	
[Qiu et al., 2006]	72.0
[Das and Smith, 2009]	73.86
[Wan et al., 2006]	75.6
Alignment + Learning	76.23
LCLR	**76.41**
Experiments using **Extended data set**	
Alignment + Learning	72
LCLR	**72.75**

Table 5.3: Experimental Result For Paraphrasing Identification. Our joint LCLR approach achieves the best results compared to several previously published systems, and our own two stage system implementation (Alignment + Learning). We evaluated the systems performance across two datasets: [Dolan et al., 2004] dataset and the Extended dataset, see the text for details. Note that LCLR outperforms [Das and Smith, 2009], which is a specifically designed joint approach for this task.

we generated a second corpus (called the *Extended dataset*) by sampling 500 sentence pairs from the MSR dataset for training and using the entire test collection of the original dataset. In the Extended dataset, for every sentence pair, we extended the longer sentence by concatenating it with itself. This results in a more difficult inference problem because it allows more mappings between words. Note that the performance on the original dataset sets the ceiling on the second one.

The results are summarized in Table 5.3. The first part of the table compares the LCLR system with a two stage system (*Alignment + Learning*) and three published results that use the MSR dataset (we only list single systems in the table[3]). Interestingly, although still outperformed by our joint LCLR algorithm, the two stage system is able perform significantly better than existing systems for that dataset.

We hypothesize that the similarity in performance between the joint LCLR algorithm and the two stage (*Alignment + Learning*) systems is due to the limited intermediate representation space for input pairs in this dataset. We evaluated these systems on the more difficult *Extended* dataset. Results indeed show that the margin between the two systems increases as the inference problem getting harder: the feasible region of the inference problem becomes much larger in the *Extended* dataset.

5.5 Analysis

In this section, we further analyze the results of LCLR by asking the following research questions:

- How important is it to choose the proper definition of the intermediate representation?
- How does LCLR compete with the approximation algorithm used in [Felzenszwalb et al., 2008; Cherry and Quirk, 2008; Chang et al., 2009]?

[3]Previous work [Das and Smith, 2009] has shown that combining the results of several systems improves performance.

Intermediate representation	vertex-vertex	edge-edge	consistence
one-to-one alignment	$\forall v_1, \sum_{v_2 \in V(G_2)} h_{v_1,v_2} = 1,$	$\forall e_1, \sum_{e_2 \in E(G_2)} h_{e_1,e_2} = 1,$	Eq. (5.8)
+ null word	Eq. (5.6)	$\forall e_1, \sum_{e_2 \in E(G_2)} h_{e_1,e_2} = 1,$	Eq. (5.8)
+ null word + null relation	Eq. (5.6)	Eq. (5.7)	Eq. (5.8)

Table 5.4: Three different definitions of the intermediate representation used in the experiments of analyzing the Paraphrase Identification task with LCLR. Note that we only show the constraints that flows from sentence 2 to sentence 1 for the sake of space. The definition of "+ null word + null relation" structure is the same as the one defined in Section 5.3.

- What will happen if we choose a different lexical similarity metric?

We focus on the task of Paraphrase Identification because it has the largest number of training instances among the three tasks. In order to provide more details, we also show the development set results, where we split the training data into a 80%-20% split and report the testing results on the 20% portion based on the systems trained on the 80% portion.

5.5.1 How important is it to choose the proper definition of the intermediate representation?

In the task of the Paraphrase Identification, the definition of the intermediate structures is defined in Section 5.3. To answer this question, we propose three different definitions of intermediate representations including the one described in Section 5.3. The definitions are described in Table 5.4. Note that in the structure "one-to-one alignment", all of the words (dependency edges) in the first sentence need to align to some words (dependency edges) in the second sentence and visa versa. In other words, no "null word" and "null edge" are used in this definition. The intermediate representation "+ null word" allows of the use of "null word" in both directions, but still does not use "null relation." The intermediate representation " + null word + null relation" is the same as the one we reported in Section 5.3.

The experimental results of using different intermediate representation are in Table 5.5. Note that the results shows the use of "null word" and "null relation" are very important as it brings significant impact to the paraphrase identification accuracy. On the development data, adding null word and null relation ends up boosting the accuracy by 3.4 percent, a significant improvement. On the testing data, adding null word and relationship also improves the results by 3 percent. Therefore, the results show that using a proper definition of intermediate representation is crucial for LCLR. Since changing the definition of the intermediate representation implies changing the feature function, picking up the proper definition of intermediate representation is very important.

It is important to note that the inference algorithm needs to be changed once we choose a different

Algorithms	(FULL) Algorithm 10		(APPROX.) Algorithm 12	
intermediate representation	Dev	Test	Dev	Test
one-to-one alignment	73.6	73.4	71.0	69.3
+ null word	75.8	76.2	72.9	72.9
+ null word + null relation	77.0	76.4	74.9	75.4

Table 5.5: The table serves two purposes. First, we want to see the impacts of using different definitions of intermediate representations. Second, we compared the LCLR (Algorithm 10) and the approximation algorithm (Algorithm 12). See the text for more details.

definition of latent structures. This strengthen our claim of formulating the inference problem as an integer linear programming problem because changing the definition of latent structures essentially means that adding variables and constraints into the formulation.

5.5.2 How does LCLR compete with the approximation algorithm used in [Felzenszwalb et al., 2008; Cherry and Quirk, 2008; Chang et al., 2009]?

In [Felzenszwalb et al., 2008; Cherry and Quirk, 2008; Chang et al., 2009], they proposed an approximation algorithm to solve the formulation that is similar to Eq. (5.3) by calling an external SVM solver iteratively. The approximation algorithm works as follows: Note that Algorithm 12 does not have any theoretical guarantee so the objective function needs not necessary to decrease. However, Algorithm 12 is simpler to implement given that it calls an external SVM solver several times. Also, the previous works report successes of using this approximation algorithm so it is interesting to know if the approximation algorithm can work well compared to LCLR on the Paraphrase Identification task.

Algorithm 12 *Approximation* Algorithm for solving Eq (5.3): The algorithm calls a SVM/LR package multiple times. Note that this algorithm does not have any theoretical guarantee possessed by Algorithm 10. Also note that this algorithm treats positive examples and negative examples equally.

1: initialize: $\mathbf{w} \leftarrow \mathbf{w}_0$
2: **repeat**
3: **for all** positive and negative examples $(\mathbf{x}_i, z_i)$ **do**
4: Find $\mathbf{h}_i^* \leftarrow \arg\max_{\mathbf{h} \in \mathcal{H}} \sum_s h_s \mathbf{w}^T \Phi_s(\mathbf{x}_i)$
5: **end for**
6: Update $\mathbf{w}$ by solving a regular SVM/Logistic formulation with features based on $\mathbf{h}_i^*$ for i-th example
7: **until** finish all iterations
8: return $\mathbf{w}$

The comparison results are also reported in Table 5.5. Note that the approximation algorithm fails to obtain good results over all different definitions of intermediate representations in both development set and testing set. In that task of Paraphrase Identification, it is apparently important to use the Algorithm 10, instead of using the approximated version, Algorithm 12.

Lexical Similarity Metric	WNSIM		WUP	
intermediate representation	Alignment+Learning	LCLR	Alignment+Learning	LCLR
one-to-one alignment	73.8	73.4	71.3	73.2
+ null word	75.6	76.2	72.2	76
+ null word + null relation	76.2	76.4	72.3	75.9

Table 5.6: The results of using different lexical similarity metrics WNSIM and WUP. Note that while using WUP gives the worse results with the Alignment+Learning approach. The system LCLR with WUP measurement is comparable results of the LCLR with WNSIM.

5.5.3 What will happen if we choose a different lexical similarity metric?

Among the features used in our Paraphrase Identification system, the most important one is the lexical similarity metric. The lexical similarity metric, which determines the distance between two words, is used in finding the initial alignment between two sentences and also affects the constructions of other features. We follow the settings used in [Do et al., 2009] and conduct experiments with two different similarity metrics: WUP [Wu and Palmer, 1994] and WNSIM [Do et al., 2009]. WUP is a hierarchical metric defined over the WordNet hierarchy, and WNSIM is a metric proposed in [Do et al., 2009] to overcome some problems of WUP. In the report of [Do et al., 2009], they observe that WNSIM performs a little bit better than WUP in their experiments, though the differences are quite small.

Note that in the experiments conducted by [Do et al., 2009], their systems just uses the similarity metric to generate a similarity score between two sentences without using other features. In our experiments, we integrate the similarity features with other features including POS, dependency tree and named entity features. The results are in Table 5.6. Interestingly, while WNSIM and WUP do not impact results a lot in the experiments conducted in [Do et al., 2009], WNSIM and WUP make a big difference in our Alignment+Learning approaches. The reason is that while the WNSIM and WUP perform similarly as a scoring function, they in fact generate very different alignments. Our syntactic features are associated to the alignments generated by the lexical similarity so different alignments can give very different paraphrase identification results. For example, when we use "+ null word + null relation", the difference of using WUP and WNSIM are about 4%, a very significant difference.

Interestingly, note that the differences between LCLR with WNSIM and LCLR with WUP are only 0.5%. While WUP does not generate good alignments for paraphrase identification at beginning, our joint approach LCLR is able to correct the alignments, find better alignments and generate better final results. On one hand, it probably shows that the WNSIM already generates very good alignments so that the improvement of LCLR over Alignment+Learning is not significant. On the other hand, when we adopt the WUP as our lexical similarity metric, LCLR performs significantly better than the Alignment+Learning approach.

5.6 Related Work

Recent NLP research has largely focused on two-stage approaches. Examples include RTE [Zanzotto and Moschitti, 2006; MacCartney et al., 2008; Roth et al., 2009]; string matching [Bergsma and Kondrak, 2007]; transliteration [Klementiev and Roth, 2008]; and paraphrase identification [Qiu et al., 2006; Wan et al., 2006].

[MacCartney et al., 2008] considered constructing a latent representation to be an independent task and used manually labeled *alignment* data [Brockett, 2007] to tune the inference procedure parameters. While this method identifies alignments well, it does not improve entailment decisions. This strengthens our intuition that the latent representation should be guided by the final task.

There are several exceptions to the two-stage approach in the NLP community [Haghighi et al., 2005; McCallum et al., 2005; Goldwasser and Roth, 2008; Das and Smith, 2009]; however, in those cases, the intermediate representation and the inference for constructing it are closely coupled with the application task. In contrast, LCLR provides a general inference formulation that allows use of expressive constraints, making it applicable to many NLP tasks that require latent representations.

While there are other general purpose latent variable SVM frameworks [Felzenszwalb et al., 2009; Yu and Joachims, 2009], one special feature for LCLR is that it utilizes the declarative inference framework that allows the use of *constraints* over an intermediate representation and provides a general platform for a wide range of NLP tasks.

The optimization procedure in this work and [Felzenszwalb et al., 2009] are quite different. We use the dual coordinate descent and cutting-plane methods, ensuring we have fewer parameters and that the inference procedure can be easily parallelized. Our procedure also allows different loss functions. [Cherry and Quirk, 2008] adopts the Latent SVM algorithm to define a language model. However, their implementation is not guaranteed to converge.

In CRF-like models with latent variables [McCallum et al., 2005], the decision function marginalizes over the all hidden states when presented with an input example. Unfortunately, the computational cost of applying their framework is prohibitive with constrained latent representations. In contrast, our framework requires only the *best* hidden representation instead of marginalizing over all possible representations, thus reducing the computational effort.

5.7 Summary

We consider the problem of learning over an intermediate representation. We assume the existence of a latent structure in the input, relevant to the learning problem, but not accessible to the learning algo-

rithm. Many NLP tasks fall into these settings and each can consider a different hidden input structure. We propose a unifying thread for the different problems and present a novel framework for Learning over Constrained Latent Representations (LCLR). Our framework can be applied to many different latent representations such as parse trees, orthographic mapping and tree alignments. Our approach contrasts with existing work in which learning is done over a *fixed* representation, as we advocate *jointly* learning it with the final task.

We successfully apply the proposed framework to three learning tasks – Transliteration, Textual Entailment and Paraphrase Identification. Our joint LCLR algorithm achieves superior performance in all three tasks. We attribute the performance improvement to our novel training algorithm and flexible inference procedure, allowing us to encode domain knowledge. This presents an interesting line of future work in which more linguistic intuitions can be encoded into the learning problem. For these reasons, we believe that our framework provides an important step forward in understanding the problem of learning over hidden structured inputs.

Chapter 6

Joint Structured Learning With Binary Indirect Supervision Signals

Many tasks in Natural Language Processing (NLP) and Computer Vision can be naturally modeled as structure learning problems. Consider, for example, the problem of *entity transliteration*. Given an English name (e.g., `Italy`), generate the corresponding name in Hebrew (pronounced `Ee'Tal'Ya`). To do so, there is a need to learn a model which, given a pair {English named entity, its Hebrew transliteration}, finds phonetic (character sequence) alignments between them.

In Chapter 3, we discussed the constrained framework for output structured variables. In Chapter 5, we proposed a constrained framework for latent structured variables. Interestingly, we can combine these two notions of "structure" in a unified and principled statistical learning framework. In this section, we formalize the observation that many structured output prediction problems have a *companion* binary decision problem: predicting whether an input can produce a good structure or not. Typically, structured output learning uses *direct supervision* consisting of annotated structures. The new framework incorporates binary labeled examples for the companion task into the learning process, acting as *indirect supervision*. Note that in Chapter 3, we treat constraints as indirect supervision resource signals. In this chapter, we use binary labeled examples as indirect supervision. In the following, we refer to this new type of signals as *binary indirect supervision signals*.

For instance, consider the following binary problem [Klementiev and Roth, 2008]: given Named Entities (NE) in two languages, determine if they represent transliterations of each other. Since transliterations of NEs should sound similar, this binary decision problem can be formulated as the following question: "Can the two NEs produce good phonetic sequence alignment?" This binary decision task determines whether a given input (here, a pair of English and Hebrew NEs) can generate a well-formed structure (in this case, the mapping). Our work is motivated by several recent works [Chang et al., 2009; Felzenszwalb et al., 2009; Chang et al., 2010a] which solve *binary decision problems* by learning to predict a (latent) structure on which the binary labels depend. We observe that it is often easy to obtain supervision (binary labels) for the companion problem, and we refer to the supervision for the companion binary task as *indirect supervision* for the target structure prediction task. The key research question addressed in this chapter is: *Can the*

*structured output prediction task benefit from **indirect supervision** to the companion problem?*

We propose a novel framework for Joint Learning with Indirect Supervision (**J-LIS**) which uses both structured and binary labeled data. In this framework, one can learn structure from a very small number of structured labeled examples, which are hard to come by, and gain a lot from indirect binary supervision for the companion decision problem that, as we show, are easy to obtain. It is important to note that while J-LIS and LCLR (the framework proposed in Chapter 5) are two strongly related learning frameworks, they are very different conceptually. The differences between these two algorithms are discussed in Chapter 6.6.3.

We develop a learning algorithm for this formulation that generalizes the structured output SVM (SSVM) by *jointly* learning from both forms of supervision. Moreover, our optimization algorithm allows the incorporation of constraints on the output space, an approach that is often found very useful in structured output problems.

6.1 Learning with Binary Indirect Supervision Signals

Before describing the model design, we elaborate the idea of using binary indirect supervision using the following two examples:

Information Extraction An advertisement on Craigslist contains fields like *size*, *rent* and *location*. Extracting these fields from text is a sequence tagging problem. A companion binary decision can be defined as that of determining if an article is a well-formed advertisement. A well-formed advertisement should contain certain fields in an appropriate ordering. While the binary task only asks whether advertisement fields can be extracted from the article, the relation between the binary task and the structured task is clear: only a well-formed post can generate good structure. The labeled data for the binary problem is easy to obtain, for example by crawling the web, and generating negative data by shuffling word sequence of the advertisement[1].

Object Part Recognition Consider the computer vision object recognition task of labeling the "parts" (e.g., wheels) of a car in an image. The companion binary task can be defined as classifying if an image contains a car or not. The relation between the problems is clear, as an image containing a car will also contain car parts. While labeling parts in an image is a difficult and time consuming task, obtaining car and

[1]Shuffling should also be considered as a supervision resource, since we *know* it will generate ill-formed examples for this domain with very high probability.

non-car images for the binary task is easy. We can collect the car images and non-car (flower, aircraft, etc.) images from the web relatively easily.

6.1.1 Model

We now formalize the intuition described above. First, we introduce the notation. Let $S = \{(\mathbf{x}_i, \mathbf{h}_i)\}_{i=1}^{l}$ denote the direct supervision set consisting of l examples $\mathbf{x}_i$ and their corresponding structures $\mathbf{h}_i$. Likewise, let $B = \{(\mathbf{x}_i, z_i)\}_{i=l+1}^{l+m}$ denote the *indirect supervision* set, where each $z_i \in \{1, -1\}$. For brevity, we write $i \in S$ to indicate $(\mathbf{x}_i, \mathbf{h}_i) \in S$ and $i \in B$ to indicate $(\mathbf{x}_i, z_i) \in B$. We denote B^+ and B^- as partitions of B consisting of positive and negative instances of B respectively. For any $\mathbf{x}$, $\mathcal{H}(\mathbf{x})$ denotes the set of all feasible structures. Let $\Phi(\mathbf{x}, \mathbf{h})$ be a feature generation function. We define $\mathcal{X}$ as the set of all feature vectors for an $\mathbf{x}$. That is, $\mathcal{X} = \{\Phi(\mathbf{x}, \mathbf{h}) \mid \mathbf{h} \in \mathcal{H}(\mathbf{x})\}$.

In the standard structured output prediction task, the goal of learning is to find a weight vector that, for every example $(\mathbf{x}_i, \mathbf{h}_i) \in S$, assigns the highest score to the correct element $\mathbf{h}_i$ of $\mathcal{H}(\mathbf{x}_i)$. That is, we wish to find $\mathbf{w}$ such that,

$$\mathbf{h}_i = \underset{\mathbf{h} \in \mathcal{H}(\mathbf{x})}{\arg\max}\, \mathbf{w}^T \Phi(\mathbf{x}_i, \mathbf{h}).$$

To use binary indirect supervision, the key assumption we make is that an input $\mathbf{x}$ is valid (that is, its $z = +1$) if and only if its best structure is well-formed. Conversely, the input is invalid (its $z = -1$) if every structure for that input is bad. We require that the weight vector $\mathbf{w}$ should satisfy two conditions:

$$\begin{aligned} &\forall\ (\mathbf{x}, -1) \in B^-, \quad \forall \mathbf{h} \in \mathcal{H}(\mathbf{x}), \mathbf{w}^T \Phi(\mathbf{x}', \mathbf{h}) \leq 0, \\ &\forall\ (\mathbf{x}, +1) \in B^+, \quad \exists \mathbf{h} \in \mathcal{H}(\mathbf{x}), \mathbf{w}^T \Phi(\mathbf{x}, \mathbf{h}) \geq 0 \end{aligned} \tag{6.1}$$

The geometric interpretation of this setting is shown in Fig. 6.1. In the figure, the circles represent the set $\mathcal{X} = \{\Phi(\mathbf{x}, \mathbf{h}) \mid \mathbf{h} \in \mathcal{H}(\mathbf{x})\}$, the feature vectors corresponding to feasible structures of an example $\mathbf{x}$. The vector $\mathbf{w}$ defines a hyperplane separating the examples into positive and negative classes. The figure on the left denotes standard structured output learning, where $\mathbf{w}$ is learned using a labeled set S. A positive example, $\mathbf{x}$ has a well defined structure, but our prediction is incorrect. In the right figure, the learning algorithm uses binary indirect supervision. Following the definition that structures obtained from negative examples should have low scores, the weight vector $\mathbf{w}$ is pushed into a better region, allowing the structure predictor to improve and predict the correct structure.

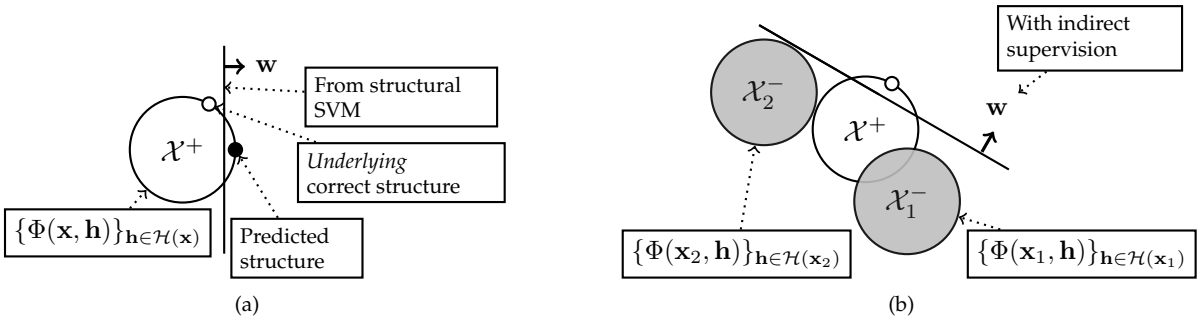


Figure 6.1: Learning with indirect supervision when the target output is H. Each circle represents the set of feature vectors of feasible structures of an example and $\mathbf{w}$ denotes a hyperplane. **(a)** Suppose we have learned a $\mathbf{w}$ using a structure labeled set S. For a positive example, $\mathbf{x} \in B^+$, we know there exists a well defined, unknown structure, but our prediction is incorrect. **(b)** After adding two negative examples: Negative examples, by definition, do not have a well formed structure. That is, *every* structure for $\mathbf{x}_1, \mathbf{x}_2 \in B^-$ should be scored below a threshold, and *some* structure of $\mathbf{x}$ should score above it. The negative examples restrict the space of hyperplanes supporting the right decision for $\mathbf{x}$. See Section 6.1 for details.

6.1.2 Learning

In the standard structural SVM, the goal of learning is to solve the following minimization problem:

$$\min_{\mathbf{w}} \frac{\|\mathbf{w}\|^2}{2} + C_1 \sum_{i \in S} L_S(\mathbf{x}_i, \mathbf{h}_i, \mathbf{w})$$

where, $L_S(\mathbf{x}_i, \mathbf{h}_i, \mathbf{w})$ represents the loss function for the structure prediction. The function L_S can be written as

$$L_S(\mathbf{x}_i, \mathbf{h}_i, \mathbf{w}) = \ell\left(\max_{\mathbf{h}} \left(\Delta(\mathbf{h}, \mathbf{h}_i) - \mathbf{w}^T \Phi_{\mathbf{h}_i,\mathbf{h}}(\mathbf{x}_i)\right)\right) \tag{6.2}$$

where $\Phi_{\mathbf{h}_i,\mathbf{h}}(\mathbf{x}_i) = \Phi(\mathbf{x}_i, \mathbf{h}_i) - \Phi(\mathbf{x}_i, \mathbf{h})$ and Δ is a function which returns the distance between two structures. We define Δ as the Hamming distance between structures, but our algorithm can be used with any suitable definition for the distance between structures. Again, the function $\ell : \mathbb{R} \to \mathbb{R}$ can be instantiated by many commonly used loss functions such as hinge loss, with $\ell(a) = \max(0, a)$, and squared-hinge loss, with $\ell(a) = \max(0, a)^2$.

We incorporate binary indirect supervision using the intuition from Eq. (6.1) to define the problem of **Joint Learning with Indirect Supervision (J-LIS)**:

Given a structure labeled dataset S and a binary labeled dataset B, the goal of learning is to find $\mathbf{w}$ that minimizes the objective function $Q(\mathbf{w})$, which is defined as

$$\frac{\|\mathbf{w}\|^2}{2} + C_1 \sum_{i \in S} L_S(\mathbf{x}_i, \mathbf{h}_i, \mathbf{w}) + C_2 \sum_{i \in B} L_B(\mathbf{x}_i, z_i, \mathbf{w}), \tag{6.3}$$

where L_S, as before, is the loss for the structure prediction, and L_B is the loss for the binary prediction.

The intuition from Eq. (6.1) is incorporated into the binary loss L_B. The first inequality of Eq. (6.1) is equivalent to requiring the highest scoring structure of all negative examples to be below the threshold. The second inequality will be satisfied if the highest scoring structure of every positive example is above the threshold. Thus, the two inequalities can be re-stated as follows:

$$\begin{aligned} \forall\ (\mathbf{x}, -1) \in B^-, \quad & \max_{\mathbf{h} \in \mathcal{H}(\mathbf{x})} \mathbf{w}^T \Phi(\mathbf{x}', \mathbf{h}) \leq 0, \\ \forall\ (\mathbf{x}, +1) \in B^+, \quad & \max_{\mathbf{h} \in \mathcal{H}(\mathbf{x})} \mathbf{w}^T \Phi(\mathbf{x}, \mathbf{h}) \geq 0. \end{aligned}$$

Furthermore, the design of the binary loss needs to account for the fact that different examples can have different sizes. Hence, the weight vectors $\Phi(\mathbf{x}_i, \mathbf{h})$ need to be normalized according to the size of the input. Accordingly, using $\kappa(\mathbf{x})$ as normalization for an input $\mathbf{x}$, we define the L_B as[2]

$$L_B(\mathbf{x}_i, z_i, \mathbf{w}) = \ell \left(1 - z_i \max_{\mathbf{h} \in \mathcal{H}(\mathbf{x})} (\mathbf{w}^T \Phi_B(\mathbf{x}_i, \mathbf{h})) \right) \tag{6.4}$$

where $\Phi_B(\mathbf{x}_i, \mathbf{h})^T = \frac{\Phi(\mathbf{x}_i, \mathbf{h})^T}{\kappa(\mathbf{x})}$. We also add a dummy feature in Φ_B to adjust the relative scale of L_B and L_S after normalizing the feature vector.

The J-LIS is related to several other learning frameworks such as structural SVM [Tsochantaridis et al., 2005], structural SVM with latent variables [Yu and Joachims, 2009] and Contrastive Estimation[Smith and Eisner, 2005]. The discussions of the relationships between J-LIS and these learning frameworks including the framework LCLR we proposed in Chapter 5 will be in Section 6.6.

The J-LIS is related to several other learning frameworks such as structural SVM [Tsochantaridis et al., 2005], structural SVM with latent variables [Yu and Joachims, 2009] and Contrastive Estimation[Smith and Eisner, 2005]. The discussions of the relationships between J-LIS and these learning frameworks will be in Section 6.6.

6.2 Optimization Algorithm

This section describes the optimization procedure for solving the objective function $Q(\mathbf{w})$ from Eq. (6.3). First, we study its convexity properties. We can rewrite Eq. (6.3) as

$$Q(\mathbf{w}) = F(\mathbf{w}) + G(\mathbf{w}),$$

[2]The function $\kappa(\mathbf{x})$ is not needed when the feature vectors take care of the scaling issue. It is only needed when the $\phi(\mathbf{x}, \mathbf{h})$ is sensitive to the size of $\mathbf{x}$.

Algorithm 13 Iterative algorithm for minimizing $Q(\mathbf{w})$ by repeatedly minimizing $A(\mathbf{w}, \mathbf{w}_t)$.

1: Initialize $\mathbf{w}_0$ with direct supervision S.
2: **repeat**
3: $\quad \mathbf{w}_{t+1} \leftarrow \arg\min_{\mathbf{w}} A(\mathbf{w}, \mathbf{w}_t)$
4: **until** convergence
5: Return the final weight vector.

where F and G are given by Eq. (6.5) and (6.6), respectively.

$$F(\mathbf{w}) = \frac{\|\mathbf{w}\|^2}{2} + C_1 \sum_{i \in S} L_S(\mathbf{x}_i, \mathbf{h}_i, \mathbf{w}) + C_2 \sum_{i \in B^-} L_B(\mathbf{x}_i, z_i, \mathbf{w}) \tag{6.5}$$

$$G(\mathbf{w}) = C_2 \sum_{i \in B^+} \ell\left(1 - \max_{\mathbf{h}}(\mathbf{w}^T \Phi_B(\mathbf{x}_i, \mathbf{h}))\right) \tag{6.6}$$

If ℓ is convex and non-decreasing, F is convex. However, the function G, which includes a maximization term within it, need not necessarily be convex or concave. This renders the function Q non-convex. This is an effect of the existential quantification over positive examples in Eq. (6.1).

Since G is not concave, the Concave-Convex procedure (CCCP) [Yuille and Rangarajan, 2003] cannot be applied as in [Yu and Joachims, 2009][3]. However, we can apply an optimization procedure similar to CCCP *without* requiring G to be concave. We use the fact that our loss function ℓ is non-decreasing, which holds for commonly used loss functions such as hinge loss, square-hinge loss and logistic loss. The algorithm is in the same spirit as the one proposed in Algorithm 10 in Section 5.2.1 but is more general given that we also have the term that associated with the structured labeled data.

6.2.1 Main Optimization Procedure

Algorithm 13 iteratively improves the objective function. At the t^{th} iteration of the loop, we denote $\mathbf{w}_t$ to be the current estimation of the weight vector, and denote $\mathbf{h}_i^t = \arg\max_{\mathbf{h}} {\mathbf{w}_t}^T \Phi(\mathbf{x}_i, \mathbf{h})$ to be the best structure for a positive example according to $\mathbf{w}_t$. We then define an approximation function of G using $\mathbf{w}_t$:

$$\hat{G}(\mathbf{w}, \mathbf{w}_t) = C_2 \sum_{i \in B^+} \ell(1 - \mathbf{w}^T \Phi_B(\mathbf{x}_i, \mathbf{h}_i^t)). \tag{6.7}$$

Unlike G, since the $\mathbf{h}_i^t$s are fixed using $\mathbf{w}_t$, the function $\hat{G}(\mathbf{w}, \mathbf{w}_t)$ is convex in $\mathbf{w}$ given ℓ is convex. Now, step 3 of the algorithm minimizes the following *convex* function $A(\mathbf{w}, \mathbf{w}_t)$ to obtain the next estimate of the

[3]CCCP cannot be applied to this split of Q into $F(\mathbf{w}) + G(\mathbf{w})$ because G is not concave. The theory of CCCP does not forbid a different split of the objective function where it is applicable. The split proposed in this chapter, however, leads to an intuitive and efficient algorithm which has similar guarantees as CCCP.

weights.

$$A(\mathbf{w}, \mathbf{w}_t) = F(\mathbf{w}) + \hat{G}(\mathbf{w}, \mathbf{w}_t)$$

Algorithm 13 has the following property:

Theorem 3. *If the loss function ℓ is a non-decreasing function, then in Algorithm 13, the objective function (Eq.* (6.3)*) will decrease with every iteration. That is, if $\mathbf{w}_t$ is the weight vector from the t^{th} iteration and $\mathbf{w}_{t+1}$ is the weight vector after running the Algorithm 13 for one more iteration. Then, $Q(\mathbf{w}_{t+1}) \leq A(\mathbf{w}_{t+1}, \mathbf{w}_t) \leq Q(\mathbf{w}_t)$.*

Proof. For any $\mathbf{h}$ and $\mathbf{w}$, we know that $\mathbf{w}^T\Phi(\mathbf{x}_i, \mathbf{h}) \leq \max_{\mathbf{h}'} \mathbf{w}^T\Phi(\mathbf{x}_i, \mathbf{h}')$. Since ℓ is non-decreasing, $\hat{G}(\mathbf{w}, \mathbf{w}_t) \geq G(\mathbf{w})$ for any $\mathbf{w}$. From definition of G and G_t, we have $\hat{G}(\mathbf{w}_t, \mathbf{w}_t) = G(\mathbf{w}_t)$. Thus,

$$\begin{aligned} Q(\mathbf{w}_t) =& F(\mathbf{w}_t) + G(\mathbf{w}_t) = F(\mathbf{w}_t) + \hat{G}(\mathbf{w}_t, \mathbf{w}_t) \\ \geq& F(\mathbf{w}_{t+1}) + \hat{G}(\mathbf{w}_{t+1}, \mathbf{w}_t) = A(\mathbf{w}_{t+1}, \mathbf{w}_t) \\ \geq& F(\mathbf{w}_{t+1}) + \hat{G}(\mathbf{w}_{t+1}, \mathbf{w}_{t+1}) = Q(\mathbf{w}_{t+1}). \end{aligned}$$

The first inequality is from line 3 in Algorithm 13. □

Algorithm 13 minimizes $Q(\mathbf{w})$ by constructing a sequence of convex problems $A(\mathbf{w}, \mathbf{w}_t)$ in each iteration. The algorithm is defined for any convex and non-decreasing loss functions ℓ. Next, we show how the inner loop of the Algorithm 13 (that is, minimizing $A(\mathbf{w}, \mathbf{w}_t)$) can be solved efficiently when $\ell(x) = \max(0, x)^2$, the squared hinge loss.

6.2.2 Cutting Plane Strategies for Optimization $A(\mathbf{w}, \mathbf{w}_t)$

Our approach for solving the $A(\mathbf{w}, \mathbf{w}_t)$ adapts the cutting plane strategy of [Joachims et al., 2009] to minimize $A(\mathbf{w}, \mathbf{w}_t)$. We define a **working set** for each element in S and B^- – let $\mathcal{W}_i$ and $\mathcal{V}_i$ denote the working sets of $\mathbf{x}_i \in S$ and $\mathbf{x}_i \in B^-$ respectively. While there are exponential number of possible structures for each example, the cutting plane strategy keeps only a small set of visited structures. We call the set that keeps visited structures a working set and we keep a working set for every example. We minimize A iteratively using Algorithm 14, which first updates the working sets (line 5-11) and then solves the following

Algorithm 14 Cutting plane algorithm for optimizing $A(\mathbf{w}, \mathbf{w}_t)$ in Algorithm 13 with square hinge loss.

Require: $\mathbf{w}_t$, the weight vector from t^{th} iteration
1: $\mathcal{W}_i \leftarrow \emptyset, \forall i \in S$
2: $\mathcal{V}_i \leftarrow \emptyset, \forall i \in B^-$
3: $\mathbf{h}_i^t = \arg\max_{\mathbf{h}} \mathbf{w}_t^T \Phi_B(\mathbf{x}_i, \mathbf{h}), \forall i \in B^+$
4: **repeat**
5: **for** $i \in B^-$ **do**
6: $\mathbf{h}_i^* \leftarrow \arg\max_{\mathbf{h}} \mathbf{w}^T \Phi_B(\mathbf{x}_i, \mathbf{h})$
7: Add $\mathbf{h}_i^*$ to $\mathcal{V}_i$
8: **end for**
9: **for** $i \in S$ **do**
10: $\mathbf{h}_i^* \leftarrow \arg\max_{\mathbf{h}} \mathbf{w}^T \Phi(\mathbf{x}_i, \mathbf{h}) + \Delta(\mathbf{h}_i, \mathbf{h})$
11: Add $\mathbf{h}_i^*$ to $\mathcal{W}_i$
12: **end for**
13: Update $\mathbf{w}$ by solving Eq. (5.5)
14: **until** no new element is added to any $\mathcal{W}_i$ and $\mathcal{V}_i$
15: return $\mathbf{w}$

minimization problem (line 13) :

$$\begin{aligned}
&\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C_1 \sum_{i \in S} \xi_i^2 + C_2 \sum_{i \in B} \xi_i^2 \\
&\text{S.T. } \forall i \in S, \mathbf{h} \in \mathcal{W}_i, \xi_i \geq \Delta(\mathbf{h}, \mathbf{h}_i) - \mathbf{w}^T \Phi_{\mathbf{h}_i, \mathbf{h}}(\mathbf{x}_i), \\
&\quad \forall i \in B^-, \mathbf{h} \in \mathcal{V}_i, \xi_i \geq 1 + \mathbf{w}^T \Phi_B(\mathbf{x}_i, \mathbf{h}), \\
&\quad \forall i \in B^+, \xi_i \geq 1 - \mathbf{w}^T \Phi_B(\mathbf{x}_i, \mathbf{h}_i^t)
\end{aligned} \tag{6.8}$$

The cutting plane method used in Algorithm 14 has one difference compared to the one used in [Tsochantaridis et al., 2005; Joachims et al., 2009]. Their algorithms solve the subproblem immediately when the working set is updated for one example. In our implementation, we update the working set for all of our examples and before solving the subproblem. The justification of the algorithm can be seen in Appendix A.2.

6.2.3 Dual Coordinate Descent Algorithm for Eq. (6.8)

After the working sets are fixed, we solve the convex subproblem Eq. (6.8) with a dual coordinate decrease method. We first derive the dual formation for Eq. (6.8). We would like to stress it that using square hinge loss in Eq. (6.8) enable us to discard we the $\xi_i \geq 0$ constraints. Hence, the dual formulation of Eq. (6.8) does not have any equality constraint. This allows us to use a very efficient coordinate descent method on the dual formulation of Eq. (6.8).

In the dual of Eq. (6.8), each variable corresponds to one constraint. We use αs to denote the dual

variables. For each $i \in S$, the dual contains $|\mathcal{W}_i|$ variables $\alpha_{i,j}$ for the $\mathbf{h}_{i,j} \in \mathcal{W}_i$. Similarly, we define $\alpha_{i,j}$ for every $\mathbf{h}_{i,j} \in \mathcal{V}_i, i \in B$.

The Lagrangian for (6.8) can be written down as follows:

$$\begin{aligned} L(\mathbf{w}, \{\xi\}, \{\alpha\}) =& \frac{1}{2}\|\mathbf{w}\|^2 + C_1 \sum_{i \in S} \xi_i^2 + C_2 \sum_{i \in B} \xi_i^2 \\ & - \sum_{i \in S} \sum_{j: \mathbf{h}_{i,j} \in \mathcal{W}_i} \alpha_{i,j} \left(\xi_i - \Delta(\mathbf{h}_{i,j}, \mathbf{h}_i) + \mathbf{w}^T \Phi_{\mathbf{h}_i, \mathbf{h}_{i,j}}(\mathbf{x}_i) \right) \\ & - \sum_{i \in B} \sum_{j: \mathbf{h}_{i,j} \in \mathcal{V}_i} \alpha_{i,j} \left(\xi_i - 1 + z_i \mathbf{w}^T \Phi_B(\mathbf{x}_i, \mathbf{h}_{i,j}) \right) \end{aligned}$$

Notice that for brevity, we extend the definition of $\mathcal{V}$ for elements of B^+, where each $\mathcal{V}_i$ is a singleton set consisting of $\mathbf{h}_i^t$. The gradient of Lagrangian are:

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{w}} = 0 \quad &\Rightarrow \quad \mathbf{w} = \sum_{i \in S} \sum_{j: \mathbf{h}_{i,j} \in \mathcal{W}_i} \alpha_{i,j} \Phi_{\mathbf{h}_i, \mathbf{h}_{i,j}}(\mathbf{x}_i) + \sum_{i \in B} \sum_{j: \mathbf{h}_{i,j} \in \mathcal{V}_i} z_i \alpha_{i,j} \Phi_B(\mathbf{x}_i, \mathbf{h}_{i,j}) \\ \frac{\partial L}{\partial \xi_i} = 0 \quad &\Rightarrow \quad \xi_i = \frac{1}{2C_1} \sum_{j: \mathbf{h}_{i,j} \in \mathcal{W}_i} \alpha_{i,j}, \text{ if } i \in S \\ &\Rightarrow \quad \xi_i = \frac{1}{2C_2} \sum_{j: \mathbf{h}_{i,j} \in \mathcal{V}_i} \alpha_{i,j}, \text{ if } i \in B \end{aligned}$$

Hence, the dual of (6.8) can be written as:

$$\begin{aligned} \mathbf{D}: \quad \min_{\{\alpha_{i,j}\}} \quad & \frac{1}{2} \| \sum_{i \in S} \sum_{j: \mathbf{h}_{i,j} \in \mathcal{W}_i} \alpha_{i,j} \Phi_{\mathbf{h}_i, \mathbf{h}_{i,j}}(\mathbf{x}_i) + \sum_{i \in B} \sum_{j: \mathbf{h}_{i,j} \in \mathcal{V}_i} z_i \alpha_{i,j} \Phi_B(\mathbf{x}_i, \mathbf{h}_{i,j}) \|^2 \\ & + \frac{1}{4C_1} \sum_{i \in S} \Big(\sum_{j: \mathbf{h}_{i,j} \in \mathcal{W}_i} \alpha_{i,j} \Big)^2 + \frac{1}{4C_2} \sum_{i \in B} \Big(\sum_{j: \mathbf{h}_{i,j} \in \mathcal{V}_i} \alpha_{i,j} \Big)^2 \\ & - \sum_{i \in S} \sum_{j: \mathbf{h}_{i,j} \in \mathcal{W}_i} \Delta(\mathbf{h}_{i,j}, \mathbf{h}_i) \alpha_{i,j} - \sum_{i \in B} \sum_{j: \mathbf{h}_{i,j} \in \mathcal{V}_i} \alpha_{i,j} \\ \text{S.T.} \quad & \alpha_{i,j} \geq 0, \forall i \in S, B, \forall j \end{aligned}$$

The idea of coordinate descent is simple. After each iteration, For each time we pick a variable $\alpha_{p,q}$ to change with all other $\alpha_{i,j}, \forall i \neq p$ or $j \neq q$ fixed. The subproblem is easily solvable, given that only one variable is involved. It has been shown that keeping track of the current weight vector $\mathbf{w}$ can greatly reduce the cost of calculating the gradient of dual variables $\alpha_{p,q}$ [Hsieh et al., 2008]. We keep the most updated $\mathbf{w}$ such that

$$\mathbf{w} = \sum_{i \in S} \sum_{j: \mathbf{h}_{i,j} \in \mathcal{W}_i} \alpha_{i,j} \Phi_{\mathbf{h}_i, \mathbf{h}_{i,j}}(\mathbf{x}_i) + \sum_{i \in B} \sum_{j: \mathbf{h}_{i,j} \in \mathcal{V}_i} z_i \alpha_{i,j} \Phi_B(\mathbf{x}_i, \mathbf{h}_{i,j}).$$

Once the values of αs change, it is easy to update $\mathbf{w}$ corresponding to the current values of αs.

We will discuss two cases separately:

Case 1: $i \in S$ Assume that we will change $\alpha_{p,q}$ to $\alpha_{p,q} + d$. The subproblem of finding the optimal d can be rewritten using the current $\mathbf{w}$:

$$\begin{aligned} \min_d \quad & \frac{1}{2}\|\mathbf{w} + d\Phi_{\mathbf{h}_p,\mathbf{h}_{p,q}}(\mathbf{x}_p)\|^2 + \frac{1}{4C_1}(\sum_{j:\mathbf{h}_{p,j}\in\mathcal{W}_p} \alpha_{p,j} + d)^2 - \Delta(\mathbf{h}_{p,q}, \mathbf{h}_p)d \\ \text{S.T.} \quad & d \geq -\alpha_{p,q} \end{aligned}$$

The solution of this subproblem is

$$d = \max\left(-\alpha_{p,q}, \frac{\Delta(\mathbf{h}_p, \mathbf{h}_{p,q}) - \mathbf{w}^T\Phi_{\mathbf{h}_i,\mathbf{h}_{i,j}}(\mathbf{x}_i) - \frac{\sum_j \alpha_{p,j}}{2C_1}}{\|\Phi_{\mathbf{h}_i,\mathbf{h}_{i,j}}(\mathbf{x}_i)\|^2 + \frac{1}{2C_1}}\right) \tag{6.9}$$

Case 2: $i \in B$ The subproblem can be written as:

$$\begin{aligned} \min_d \quad & \frac{1}{2}\|\mathbf{w} + dz_p\Phi_B(\mathbf{x}_p, \mathbf{h}_{p,q})\|^2 + \frac{1}{4C_2}(\sum_{j:\mathbf{h}_{p,j}\in\mathcal{V}_p} \alpha_{p,j} + d)^2 - d \\ \text{S.T.} \quad & d \geq -\alpha_{p,q} \end{aligned}$$

And the solution is:

$$d = \max\left(-\alpha_{p,q}, \frac{1 - z_p\mathbf{w}^T\Phi_B(\mathbf{x}_p, \mathbf{h}_{p,q}) - \frac{\sum_j \alpha_{p,q}}{2C_2}}{\|\Phi_B(\mathbf{x}_p, \mathbf{h}_{p,q})\|^2 + \frac{1}{2C_2}}\right) \tag{6.10}$$

Algorithm 15 summarizes the dual coordinate descent algorithm for minimizing Eq. (6.8).

Our optimization algorithm is related to the convex-concave procedure (CCCP) [Yuille and Rangarajan, 2003], which has been used for solving many non-convex optimization problems [Yu and Joachims, 2009]. We show that it is not necessary to decompose the objective function into a sum of convex and concave functions to use a CCCP-like iterative procedure. The object recognition work of [Felzenszwalb et al., 2009] uses a similar optimization procedure for their problem. However, not only is the intent of our algorithm is completely different (that work only has labels for the binary problem and the goal is to improve binary classification performance), our main optimization algorithm (Algorithm 13) can handle general loss

Algorithm 15 Dual coordinate descent for minimizing Eq. (6.8).

repeat
  Pick any variable $\alpha_{i,j}$
  if $i \in S$ **then**
    $\eta_1 = \Delta(\mathbf{h}_i, \mathbf{h}_{i,j}) - \mathbf{w}^T \Phi_{\mathbf{h}_i, \mathbf{h}_{i,j}}(\mathbf{x}_i) + \frac{1}{2C_1} \sum_{j=1}^{|\mathcal{W}_j|} \alpha_{i,j}$
    $\eta_2 = \|\Phi_{\mathbf{h}_i, \mathbf{h}_{i,j}}(\mathbf{x}_i)\|^2 + \frac{1}{2C_1}$
  else
    $\eta_1 = 1 - z_i \mathbf{w}^T (\Phi_B(\mathbf{x}_i, \mathbf{h}_{i,j})) + \frac{1}{2C_1} \sum_{j=1}^{|\mathcal{V}_j|} \alpha_{i,j}$
    $\eta_2 = \|\Phi_B(\mathbf{x}_i, \mathbf{h}_{i,j})\|^2 + \frac{1}{2C_2}$
  end if
  $\alpha_{i,j} \leftarrow \max(\alpha_{i,j} + \frac{\eta_1}{\eta_2}, 0)$
until convergence

functions which are convex and non-decreasing [4].

6.3 Implementation Details

There are several implementation details that are worthwhile to mention here.

Stopping Conditions We hope to run our algorithm until the relative change in the objective function became less than ν:

$$\frac{Q(\mathbf{w}_t) - Q(\mathbf{w}_{t-1})}{Q(\mathbf{w}_{t-1})} \leq \nu$$

Unfortunately, calculating the objective function $Q(\mathbf{w})$ is very expensive given that it requires resolving the inference problems for all examples.

On the other hand, we know the value of $A(\mathbf{w}_t, \mathbf{w}_{t-1})$ while running Algorithm 13 ($\mathbf{w}_0$ is the initial weight vector). In our system, we use the following stopping condition

$$\frac{A(\mathbf{w}_t, \mathbf{w}_{t-1}) - A(\mathbf{w}_{t-1}, \mathbf{w}_{t-2})}{A(\mathbf{w}_{t-1}, \mathbf{w}_{t-2})} \leq \nu \tag{6.11}$$

Note that from Theorem 3,

$$Q(\mathbf{w}_t) \leq A(\mathbf{w}_t, \mathbf{w}_{t-1}) \leq Q(\mathbf{w}_{t-1})$$

It means that Eq. (6.11) implies

$$\frac{Q(\mathbf{w}_t) - Q(\mathbf{w}_{t-2})}{Q(\mathbf{w}_{t-2})} \leq \frac{A(\mathbf{w}_t, \mathbf{w}_{t-1}) - A(\mathbf{w}_{t-1}, \mathbf{w}_{t-2})}{A(\mathbf{w}_{t-1}, \mathbf{w}_{t-2})} \leq \nu \tag{6.12}$$

[4] For loss functions other than square-hinge loss, optimization methods other than the dual coordinated descent method might be needed to solve the subproblem.

In our implementation, we set ν to be 10^{-5}.

Parallelization It is important to know that in the Algorithm 14, the process of finding the most violated structures can easily parallelized. In our implementation, we take advantages of this property and implement a multi-thread approach for solving inference problems (line 5-12 in Algorithm 14). This is especially important when solving the inference problem is time consuming.

Dual Coordinate Descent In Algorithm 15, we state that we pick any i, j pairs randomly for the sake of simplicity. In our implementation, we arrange i, j in a random order such that we will touch each pair at least once before we pick the same i, j pair again. We found that this ordering improves the speed of the convergence.

6.4 Experiments

We verify the effectiveness of J-LIS by applying it on several NLP tasks defined over complex structures – Phonetic Transliteration Alignment, Information Extraction [McCallum et al., 2000; Grenager et al., 2005] and Part of Speech Tagging [Smith and Eisner, 2005]. Our experiments provide insights into the settings in which binary indirect supervision is most effective. We also conduct a document classification experiment in this section. All three applications are taken from active research areas, and have been widely discussed in the literature. The binary classification problem in these applications is defined over rich structures, which in turn present a non-trivial structured learning problem. In the following subsections we explain the experimental setting for each domain and show that using binary indirect supervision consistently improves performance. For all our experiments, we selected parameters C_1 and C_2 from the set $\{10^{-1}, 10^0, 10^1\}$ by sampling the data and evaluating the results on a held-out set.

6.4.1 Phonetic Transliteration Alignment

Given a source language named entity (NE) and a corresponding target language NE, the goal of Phonetic Transliteration Alignment is to find the best phonetic alignment between the character sequences of the two NEs.[5] The companion binary classification problem is the task of determining whether two words from different languages correspond to the same underlying entity.

Given two words $a_s^1, a_s^2, ..., a_s^n$ and $a_t^1, a_t^2, ..., a_t^m$, the phonetic alignments are defined over $(a_s^i \dots a_s^k)$ and $(a_t^p \dots a_t^q)$ pairs (We only allows the character sequence contains less or equal to three characters). We allow

[5]The character sequences can consist of multiple characters.

a character sequences to map into a empty set but disallow cross alignments. Hence, given an input example (in our case, an English and Hebrew NE pair), the best sequence alignment can be efficiently found using a dynamic programming algorithm. Our feature vector corresponds to the alignment edges, using the corresponding character sequences. Note that this definition of the structure is different from the one used in Section 5.4.1, because our domain expert believes that the true underlying alignment should be the alignments of the character sequences instead of the alignments of single characters. The comparisons between the LCLR system using the single character alignment structure (the definition used in Section 5.4.1) and the character sequence structure (the definition used here) will be discussed in 6.5.5.

We used the English-Hebrew data-set from [Chang et al., 2009], consisting of 300 pairs and manually annotated the alignments between the NEs' segments. We used 100 pairs for training and 200 for testing. Our binary data, obtained by crawling the web, consists of 1,000 positive and 10,000 negative pairs[6]. Note that obtaining the binary supervision is considerably easier than labeling the alignment.

To measure the impact of binary indirect supervision, we vary the size of the direct and binary indirect supervision sets. We report the F1 measure for the alignment in Table 6.3(a). Adding binary indirect supervision improves the structure predictor significantly. For example, when we have only ten structured labeled pairs, the error reduction rate is 26%.

6.4.2 Part-Of-Speech tagging

Our part-of-speech (POS) tagging data is from the Wall Street Journal corpus [Marcus et al.]; we used 25600 tokens for training and another 25600 tokens testing (which each correspond to 1000 sentences). Using a separate set of sentences corresponding to 51200 tokens, we generated 2500 binary indirect supervision examples. The negative examples among these were generated by randomly shuffling "positive" sentences in the corpus. Note that we use the standard definition of POS with 45 possible POS tags. This experiment setting is different from the one conducted in [Chang et al., 2010b], where we used a reduced set of tags that contains only 16 tags.

Our model was a first order Markov model with spelling features representing capitalization and suffixes of the current word. The full feature description[7] can be found in Table 6.1. In our experiments, the largest model used approximately 1 million features. We set our $\kappa(\mathbf{x})$ to be the number of words in the sentence $\mathbf{x}$. All reported results are averaged over 10 rounds.

The results of the experiments are summarized in Fig. 6.2. We observe that binary indirect supervision

[6]Negative pairs were created by pairing an English NE to a random Hebrew NE.

[7]If we use the same features to train a POS model on section 2 to section 21 of the Wall Street Journal corpus. The testing accuracy is 96.5 on Section 23.

Feature type	Feature Template
Transition	$t_{i-1} = T'$ and $t_i = T$ $t_i = T$
Word	w_{i-1} and $t_i = T$
Spelling Features	suffix(w_i) and $t_i = T$ such that $\|\text{suffix}(w_i)\| \leq 3$ prefix(w_i) and $t_i = T$ such that $\|\text{prefix}(w_i)\| \leq 3$ the first character w_i is capitalized and $t_i = T$ w_i contains a hyper and $t_i = T$ w_i contains a number and $t_i = T$

Table 6.1: Features used in our POS Experiments. We use $\{t_i\}$ variables to represent labels to be predicted – t_i represent tags, and w_i represent word tokens for the i-th word, respectively. We use T to represent a possible tag among the 45 possible tags. All features are binary.

is most effective when the size of the structured labeled data-set is small. For example, when S consists of 200 tokens, adding binary indirect supervision improves the predictor from 66.17% to 71.52%. As in the transliteration domain, increasing the binary indirect supervision often results in better performance, the trend is very clear in Fig. 6.2. While the effect of binary indirect supervision decreases when the supervision set for the structure learning problem increases, we found that J-LIS is competitive with structural SVM even when we used all the labeled data (25.6k). The supervised SVM achieves 93.51% with all labeled examples, compared to J-LIS's 93.58%.

6.4.3 Information Extraction

Information Extraction (IE) is the task of identifying predefined fields in text. We report results for two IE tasks: (i) Extracting fields from citation (e.g., author, title, year) [McCallum et al., 2000], and (ii) Extracting fields from advertisements (e.g., size, rent and neighborhood) [Grenager et al., 2005]. The companion binary problem is to classify whether a text is a well structured citation/advertisement. These two tasks are only modeled as a sequential tagging problem. Followed the setting in [Grenager et al., 2005], for citations, we used 300 structured labeled examples for training, 100 for development and 100 for testing. In the advertisements domain, we used 100 labeled examples for training, development and testing. For each domain, our positive data contains 1k entries (50k tokens for the citation domain and 200k tokens for the advertisement dataset). We generate 1k negative entries for each domain by randomly shuffling the tokens from the positive examples of each domain.

We use features corresponding to the current word and previous state allowing us to use Viterbi to find the best sequence efficiently. The features can be find in Table 6.1 without spelling features. The $\kappa(\mathbf{x})$ is set to the number of tokens of this entry.

Our experiments evaluated the extent to which labeled data for the binary classification task could be

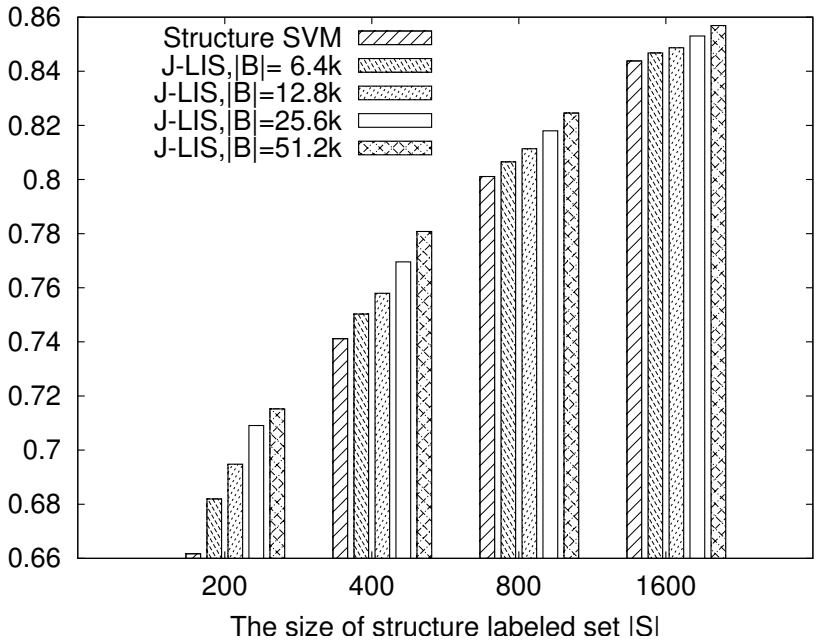


Figure 6.2: **Results for the part-of-speech tagging.** Adding binary indirect supervision significantly improves the results. Also, the results are better when more binary indirect supervision is used. We report the size of the data-sets used by counting the total number of **tokens** due to the variance in sentences size. See the text for more details. Even when *all* of our labeled data is used (25.6k), SSVM and J-LIS are comparable(93.51% v.s. 93.58%)

used as binary indirect supervision for the information extraction task. In the citation domain, we used 300 labeled examples for training, 100 examples for development and 100 examples for testing. Out of these examples token level examples were generated, 9K for training, 3K testing. A set of 1000 positive citation entries were used as binary indirect supervision, and a further 1000 negative examples were generated, which correspond to 50000 token level negative examples. In the advertisement domain, we used 100 labeled examples for training, development and testing. These correspond 19000 token level examples for training and 20000 token level examples for testing. A set of 1000 positive citation entries were used as binary indirect supervision, and a further 1000 negative examples were generated, which correspond to 20000 token level negative examples.

The token-level accuracy results for both domains are summarized in Table 6.3(b), where we vary $|S|$ and fix $|B|$ for each domain. In the advertisement domain, when the number of tokens in S is 500 tokens, structural SVM attains an accuracy of approximately 58%. Adding binary indirect supervision boosts the accuracy to over 66%, which even outperforms the structural SVM results with $|S| = 1000$. In the citation domain, we observe similar trends.

	SSVM	J-LIS		
	Size of B			
$\|S\|$	0	$2k$	$4k$	$8k$
10	72.9	78.8	79.8	80.0 (26.2%)
20	82.1	84.6	84.7	85.4 (18.4%)
40	85.7	86.9	87.2	87.4 (12.0%)
80	88.6	89.4	89.0	89.4 (7.1%)

(a) Phonetic Alignment Results

Advertisements			Citations		
$\|S\|$	SSVM	J-LIS	$\|S\|$	SSVM	J-LIS
500	57.18	**66.60**	100	58.24	**64.05**
1000	64.93	**70.09**	400	70.53	**73.63**
2000	69.57	**72.75**	1600	82.39	**84.28**
4000	74.24	**75.80**	6400	90.15	90.33
19k	78.07	79.00	9k	92.24	92.31

(b) Information Extraction Results

Figure 6.3: **Left (a): F1 measure for the phonetic transliteration alignment task.** The amount of direct supervision used for the structured prediction task ($|S|$) varies across the rows of the table, while the size of binary indirect supervision ($|B|$) changes across the columns. The first column, ($|B| = 0$) is the standard structural SVM (SSVM). Results show that binary indirect supervision is especially effective when little supervision exists for the structured task. The error reduction compared to structural SVM is in parentheses. **Right (b): Results for two IE tasks.** The size of structured supervised set S is measured by number of **tokens**. Performance is evaluated by token-level accuracy with fixed binary indirect supervision set B. The results are bold faced when the improvement obtained by J-LIS is statistically significantly under paired-t test $p < 0.01$.

6.4.4 Document Classification

The J-LIS can also be applied on document classification tasks. In the document classification tasks, our goal is to classify the document to a set of predefined classes of categories. For example, we want to classify a news document into the "Sports" category or the "Health" category. An important insight here is that there are documents that do not belong to any of these categories[8], and we can take advantages of those documents to improve our target classifiers.

In order to have a better understanding how to set up this experiment, we list the semantics of each type of data in Table 6.2. In our experiment, we conduct this experiment on the commonly used 20 newsgroup dataset [Lang, 1995; Fan and Lin]. We randomly select 10 categories as our predefined categories. That is, our main focus now is a 10-way classification problem. In the structured labeled data, we have the standard definition of "labeled data", where we know the correct category for each document. We separate the original training data into two sets. We sample the structured labeled data from the first set, and we artificially generated the binary labeled data from the second set. The positive binary labeled data is generated by picking examples that belong to category 1 to 10 from the second set and then throw away the label associated to each example. The negative data is based on the examples that belong to category 11 to category 20.

It is important to note that we only present to the classifier the examples that belong to these predefined categories (category 1 to category 10) in the testing time. We varies the size of structured labeled data here and fixed the size of the binary labeled data in all of our experiments. We set $\kappa(\mathbf{x}) = 1$ here and report

[8]We assume that in the "Miscellaneous" does not belong to our predefined categories.

Data Type	Semantics	Mathematics Semantics
Structured Labeled Data S	Document $\mathbf{x}_i$ and the label $\mathbf{h}_i$ belongs to class 1 to m	$\mathbf{w}^T\Phi(\mathbf{x}_i, \mathbf{h}_i) \geq \max_{\mathbf{h}=\{1,\ldots,m\}} \mathbf{w}^T\Phi(\mathbf{x}_i, \mathbf{h})$
Positive Data B^+	Document $\mathbf{x}$ belongs to class 1 to m. However, we do not now which class it is.	$\max_{\mathbf{h}=\{1,\ldots,m\}} \mathbf{w}^T\Phi_B(\mathbf{x}_i, \mathbf{h}) \geq 0$
Negative Data B^-	Document $\mathbf{x}$ does **NOT** belong to class 1 to m.	$\max_{\mathbf{h}=\{1,\ldots,m\}} \mathbf{w}^T\Phi_B(\mathbf{x}_i, \mathbf{h}) < 0$
Testing Data	This document belongs to class 1 to m. However, we do not now which class it is.	prediction: $\arg\max_{\mathbf{h}=\{1,\ldots,m\}} \mathbf{w}^T\Phi(\mathbf{x}_i, \mathbf{h})$

Table 6.2: The semantics of structured and binary labeled data in the document classification problem. Assume that our target task is to classify a document into a set of predefined categories: $\{1, \ldots m\}$. Note that the binary negative examples can be obtained by collecting documents which do not belong to class $1 \ldots m$.

results over 10 rounds.

The experimental results are in Figure 6.4. In the figure, it is clear that binary labeled data significantly improve the accuracy on the target task, especially when we do not have a lot of labeled data.

In this experiment, we found one limitation of the J-LIS framework. When there are extremely few labeled examples (less than 10 examples for each category), adding binary labeled data can hurt the performance (in fact, it can drop the accuracy by 20%). Our hypothesis is that given that our objective function is non-convex, the initialization point is very important. When there are not enough amount of labeled examples, we do not have enough information to construct a good initialization point. This explains why adding binary labeled data here can hurt us. Setting up a procedure to ensure a good initialization weight vector is one of our future work.

6.5 Analysis

In this section we discuss several important aspects of our framework. First, we would like to know how much impacts negative examples bring. The use of negative examples is unique in J-LIS compared to other discriminative semi-supervised learning framework.

We would also want to investigate some other properties of our algorithm. This includes analyzing the number of iterations it needs as well as the impact of generating more negative examples.

Moreover, we extrapolate the experiments in Section 6.4 to consider the case when there are no labeled structure examples and discuss learning with just the binary indirect supervision.

In the final part of this section, we consider a common scenario in many natural NLP and computer vision applications is using Y (instead of H) as the target output space. We explore the question of using

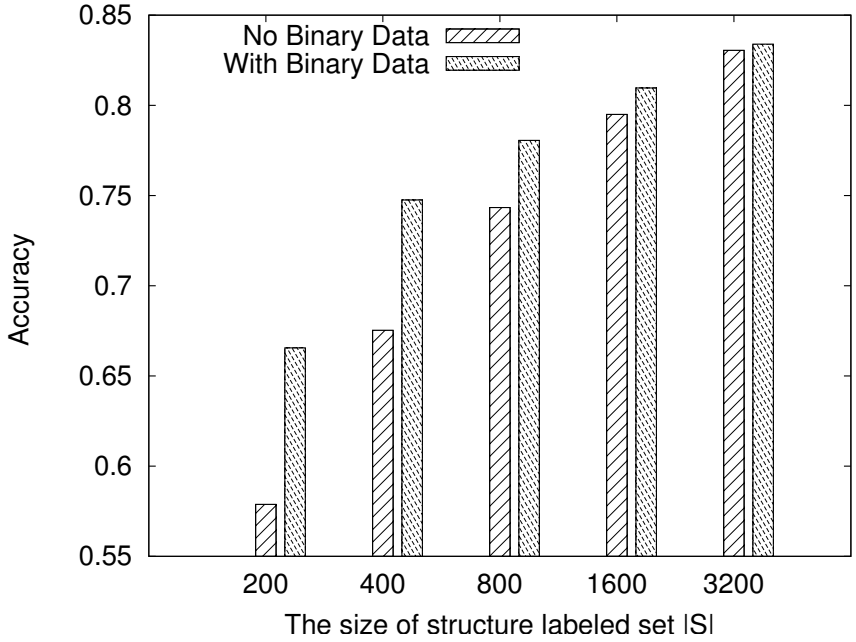


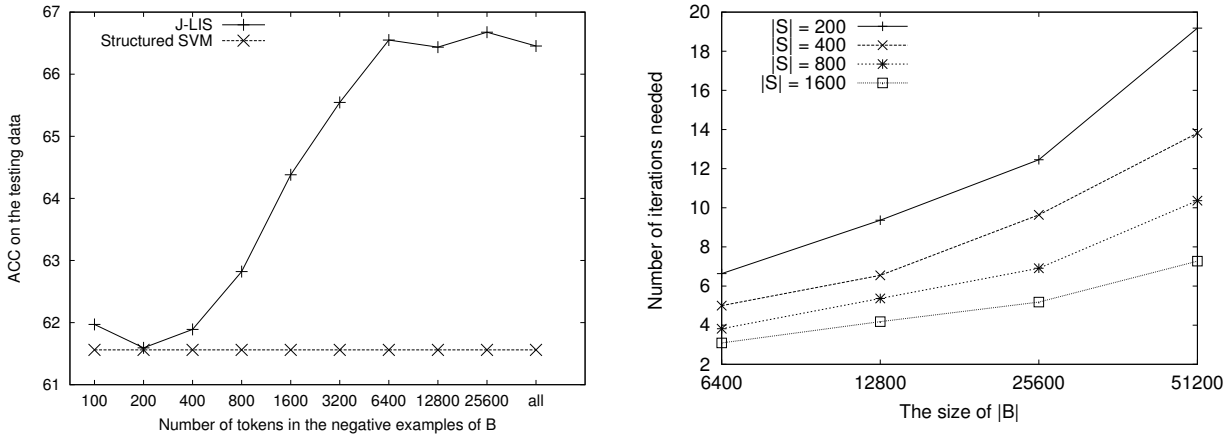
Figure 6.4: Document Classification Experiments on 20 newsgroup. We do not show the results when $|S| < 100$ give that we find adding binary labeled data stop working in these cases. See the text in Section 6.4.4 for more details.

supervision for the structure (H) as binary indirect supervision for the binary classification task. The key novel property of our algorithm is that it uses "invalid" examples as a source of supervision. In the third part of the section, we study experimentally the question of how "invalid" examples impact learning from binary indirect supervision.

6.5.1 How much impacts does negative examples bring?

The key difference between J-LIS and previous work on discriminative semi-supervised structured output prediction (e.g., [Brefeld and Scheffer, 2006; Zien et al., 2007]) is the use of *invalid* ($y = -1$) examples. These approaches use well-formed unlabeled examples for training. These correspond only to our $y = +1$ class. To provide further insight into the role of negative examples, we isolate the contribution of the invalid examples in the binary indirect supervision dataset by fixing the number of positive examples, and show the effect of varying the number of negative examples in the citation domain. We fix the structured labeled set ($|H| = 100$ tokens). The binary indirect training set is created as follows – positive examples are fixed (34767 tokens) and the number of negative examples is varied.

Results in Fig. 6.5(a) show that increasing the number of negative examples improves the performance of the structure predictor. This stresses the advantage of J-LIS over the standard discriminative



(a) (b)

Figure 6.5: **(a) Left:** Impact of negative examples in the citation domain. Results show that as more negative examples ($y = -1$) are used, binary indirect supervision improves performance.
(b) Right: Number of iterations needed in the POS experiments. The results show that we need more iterations when the number of structured labeled examples decreases or the number of binary labeled examples increases.

semi-supervised approaches which cannot gain from negative examples.

6.5.2 How many iterations does J-LIS take to converge?

Algorithm 13 seldom takes more than 50 iterations to stop respect with our stopping condition (Eq. (6.11)). In Fig. 6.5(b), we show the average number of iterations over 10 rounds in the POS experiments. Interestingly, we found that J-LIS needs to run more iterations when the number of structured labeled examples decreases or the number of binary labeled examples increases. When there is a large amount of structured labeled data, the algorithm sometimes finish the algorithms in just 2 or 3 iterations. Our hypothesis is that when there a large amount structured labeled data, the binary task becomes very easy to learn and a small number of iterations are often enough.

6.5.3 What are the impacts of generating more negative examples?

In the POS Experiments in Section 6.4.2, we generate one negative example by shuffling a positive example. In this section, we conduct the following experiments to understand the impacts of generating more negative experiments.

In this experiment, we use 10 labeled sentences and approximately 250 sentences in the positive training data. Then, we generate k negative sentences from each sentence in the 250 positive sentence dataset, where

$k = 1, 2, 4, 8, 16$. The experimental results are in Figure 6.6(a). It shows that there exists a positive correlation between the number of generated examples and the final accuracy. Again, it implies that J-LIS uses negative examples to learn "what not to do" so generating more examples can be helpful. Generating informative negative examples is an important future research topic for this framework.

6.5.4 When the size of S is zero: compare to Contrastive Estimation

Contrastive Estimation [Smith and Eisner, 2005] (CE) is a probabilistic framework for unsupervised structured learning with log-linear models. The key idea of CE is to find bad "neighbors" for a given example and then push the probability mass from the bad neighbors to the given example. More precisely, for an given example $\mathbf{x}$, it models the probability $P(\mathbf{x})$ by

$$P(\mathbf{x}) = \frac{\sum_{\mathbf{h}} \exp(\mathbf{w}^T \Phi(\mathbf{x}, \mathbf{h}))}{\sum_{\mathbf{h}, \hat{\mathbf{x}} \in \mathcal{N}(\mathbf{x})} \exp(\mathbf{w}^T \Phi(\hat{\mathbf{x}}, \mathbf{h}))},$$

where $N(\mathbf{x})$ contains bad neighbors for the current example. Note that when $N(\mathbf{x})$ contains all possible input, this framework reduce to the Expectation-Maximization framework.

CE is conceptually related to this work. However, the goal of J-LIS is to *jointly* learn from both structured and binary supervision, while CE is not designed to use labeled structures. Therefore, we compare to CE by restricting J-LIS to use $|S| = 0$. Next, we describe the conceptual difference between the approaches and then empirically compare the two algorithms and Expectation-Maximization(EM).

Even without any labeled structure, J-LIS is less restricted than CE. First, in CE, a "good" example and its "bad" neighbors need to be grouped together and CE cannot be directly applied when the relationship between good and bad examples is not known. In contrast, our framework can be directly applied to *existing* binary datasets. Moreover, CE needs to marginalize over all possible hidden structures, while J-LIS only looks for the best structure. Hence, the practical computational cost of the inference problem is lower. This property also allows us to incorporate complex domain specific constraints, which as previous work has shown can significantly boost the performance of structure predictors [Roth and Yih, 2005, 2007; Martins et al., 2009b]. It is not clear how to use arbitrary constraints in CE without using an approximated inference procedure.

Following [Smith and Eisner, 2005], we adopt the commonly used tagging dictionary assumption: for any word, we know all its possible POS (e.g. 'play' can be a verb or a noun). We used a 96K word subset of the WSJ corpus, and evaluated on ambiguous words (that is, words with more than one allowed tag), as in the CE work. All models used the second order Markov assumption and exactly the same features.

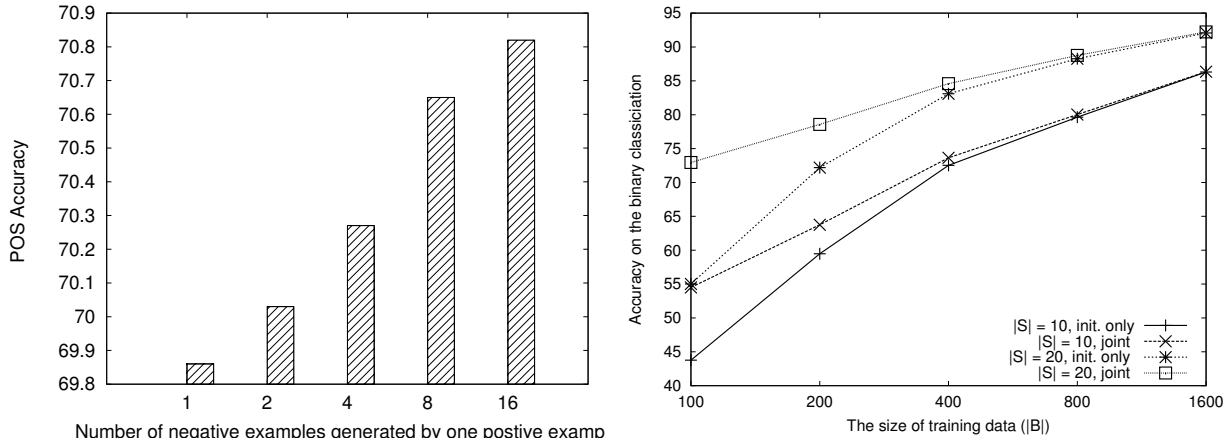


Figure 6.6: **(a) Left:** The impact of adding more negative examples in a POS experiments. As we generate more negative examples, we get better final results.
(b) Right: The impact of structured labeled data when binary classification is our target. Results (for transliteration identification) show that joint training significantly improves performance, especially when direct supervision is scarce.

The spelling features are not used here because we want to compare to EM. For J-LIS, we generate four negative examples for each positive one by randomly shuffling its words. We compare to the closest CE experimental setting which transposes neighboring words.

We found that in this setting, it is very important for J-LIS to have an inference algorithm capable of randomly choosing among equivalent best solutions. This can be easily done by random shuffling the order of the operators in the dynamic programming algorithm. The reason is that J-LIS only finds a single best structure for an instance. It is likely that many structures are assigned the same score as the best one. Without any randomization, the J-LIS will pick the first encounter structure, which is not ideal in the case of unsupervised learning.

The token accuracy of a second-order EM and CE with trigrams are 60.9% and 74.7%, respectively in a pure unsupervised learning. When hyper-parameters are tuned using supervised data, the performance can go up to 62.6% and 79.0% [Smith and Eisner, 2005]. Without any randomization, J-LIS obtains 62.89%. With randomized inference procedure the averaged accuracy for J-LIS over ten times is 70.16% (with standard deviation of 0.8%), which is significantly better than EM but worse than CE. *Interestingly, if we run J-LIS for 50 times and pick the one with the lowest objective value, we can get the accuracy 76.2% — a very significant improvement over the model with random initialization.* Again, we hypothesize that the reason is that J-LIS only finds a single best structure. While this does not affect CE, which sums over the score of all structures, J-LIS might commit to a solution too early. We further verified this intuition by adding merely

5 structured labeled sentences that provide better initial point, resulting in an accuracy of 79.1%.

6.5.5 Using structure as indirect supervision

In this section we consider the reverse question: can structured data act as indirect supervision for the binary task. Since J-LIS does not make any assumptions about S and B, it can be applied *directly* when the binary task is the target. We briefly describe experimental results in this setting using the *transliteration identification* task (determining if a given NE pair is a transliteration pair). This is the companion problem to the transliteration alignment problem considered in Sec. 6.4.1. We matched each English NE in the test data with the best Hebrew NE using the classifier's confidence and measured the accuracy of the top ranking prediction. Our test data consists of 200 English and Hebrew NEs.

There are two ways of using S to improve binary prediction: (1) Initialize the weight vector using S, and apply J-LIS to B. (2) Initialize the weight vector with S, and then apply J-LIS on both B and S (the **joint approach**).

We vary the size of the $|B|$ from 100 to 800 and keep the positive to negative ratio to 0.1 and report results for $|S| \in \{10, 20\}$. Fig. 6.6(b) shows that increasing $|S|$ improves accuracy. Furthermore, the joint method performs significantly better than using S for initialization only.

Note that the experiments in this section are similar to the experiments in Section 5.4.1. If there are no additional structured labeled examples, J-LIS becomes LCLR. However, in Section 6.4.1 and Section 5.4.1, we use two different definitions of the alignment between named entities. Here we re-run the LCLR systems with two different definitions of the structures on a slightly different training and testing split of the dataset. When using the single character alignment definition on this split (the definition used in Section 5.4.1), LCLR obtains 93.9 of the MRR. When using the character sequence alignment (the definition used in this Chapter), LCLR obtains 93.25 of MRR. When adding 50 labeled structured examples (we only have labels for character sequence alignment) as indirect supervision, J-LIS obtains 94.5 MRR. It shows that adding structured labeled example can indeed help boost the performance of the binary task.

6.6 Discussion and Related Work

In this section, we discuss the relationships between J-LIS and other learning frameworks. Note that we have discuss the relationship between our framework and Contrastive Estimation in Section 6.5.

6.6.1 Structural SVM

Several discriminative algorithms for learning structured output predictors have been proposed in the literature: these include conditional random fields [Lafferty et al., 2001], max-margin Markov network [Taskar et al., 2004] and structural SVM [Tsochantaridis et al., 2005]. These methods use feature vectors of input-output space to capture the interdependency between output variables.

Our framework generalizes some of these structure learning frameworks. When $B = \emptyset$ and ℓ is the hinge loss or squared hinge loss, it reduces to the structural SVM framework. When $S = \emptyset$ and the goal is the binary task, it is a latent variable framework similar to [Felzenszwalb et al., 2009], which learns a binary SVM over latent structures and is an instantiation of the latent structural SVM of [Yu and Joachims, 2009]. Our approach differs from both of these frameworks as it aims to use indirect supervision from one task to help the companion target task.

6.6.2 Latent Structural SVM

Latent Structural SVM [Yu and Joachims, 2009] is a framework that allows using latent structures in the structural SVM framework. The goal of Latent Structural SVM is to use latent structures as an intermediate step to improve the target task performance. J-LIS, on the other hand, allows the users of binary labeled data for structured learning tasks. Nevertheless, there are interesting connections between these two frameworks.

Recall that Latent Structural SVM can be written as

$$\min_{\mathbf{w}} \frac{\|\mathbf{w}\|^2}{2} + C \sum_i \left(\max_{\mathbf{y},\mathbf{h}} \left(\Delta(\mathbf{y}_i, \mathbf{y}, \mathbf{h}) + \mathbf{w}^T \Phi(\mathbf{x}_i, \mathbf{y}, \mathbf{h}) \right) - \max_{\mathbf{h}} \left(\mathbf{w}^T \Phi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}) \right) \right), \tag{6.13}$$

where $\mathbf{y}$ represents the *output* structures and $\mathbf{h}$ represents the *latent* structures[9].

If we restrict the output structures to be binary ($\mathbf{y} \in \{-1, +1\}$) and *take away all features used in the negative data* such that $\Phi(\mathbf{x}_i, +1, \mathbf{h}) = \Phi(\mathbf{x}_i, \mathbf{h})$, and $\Phi(\mathbf{x}_i, -1, \mathbf{h}) = \mathbf{bz}$, where $\mathbf{bz}$ is a zero vector. Latent Structural SVM will be reduced to an instantiation of the J-LIS objective function, which assume $|S| = 0$. It

[9]Note that in the Latent Structural SVM, they only use hinge loss function.

can be showed from the following derivation:

$$
\begin{aligned}
& C \sum_i \left(\max_{\mathbf{y},\mathbf{h}} \left(\Delta(\mathbf{y}_i, \mathbf{y}, \mathbf{h}) + \mathbf{w}^T \Phi(\mathbf{x}_i, \mathbf{y}, \mathbf{h}) \right) - \max_{\mathbf{h}} \left(\mathbf{w}^T \Phi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}) \right) \right) \\
=& C \sum_{\mathbf{y}_i=1} \left(1 - \max_{\mathbf{h}} \mathbf{w}^T \Phi(\mathbf{x}_i, \mathbf{h}) \right) + C \sum_{\mathbf{y}_i=-1} \left(1 + \max_{\mathbf{h}} \mathbf{w}^T \Phi(\mathbf{x}_i, \mathbf{h}) \right) \\
=& C \sum_i \left(1 - \mathbf{y}_i \max_{\mathbf{h}} \mathbf{w}^T \Phi(\mathbf{x}_i, \mathbf{h}) \right)
\end{aligned}
\tag{6.14}
$$

We refer to this trick as "asymmetric feature reduction", and this trick has also been used for reducing a structural SVM into a binary SVM (Section 2.2.1).

Eq. (6.14) shows one important property of J-LIS: it *only* models the positive data using the same definition of structures used in the structured labeled data. It does not model negative examples. In other words, the process here is asymmetric – J-LIS only measures if an example is qualify to be a positive example or not.

We argue that this asymmetric design is natural and important for J-LIS. It is often hard define an inference problem to support negative examples. Take the phonetic alignment problem as an example. In the positive examples (two words in an example represents transliteration pairs), we know that there exists good phonetic alignments. For a negative example, it is hard to find a structure to justify that there do not exist good alignments. It is tempting to set up a similar inference problem that finds "the worst alignment" to support negative examples. However, this approach will not work because bad alignments also exist in positive examples.

On the other hand, how to use Latent Structural SVM to allow using structural data as binary indirect supervision for another structural prediction problems is an interesting future research direction.

6.6.3 Relation between J-LIS and LCLR

Note that J-LIS and LCLR (the framework proposed in Chapter 5) are two strongly related learning frameworks. However, conceptually they are very different. In LCLR, the target task is to make a binary prediction and we only have binary supervision signals. While LCLR searches for the intermediate representation, we do not care about the correctness of the intermediate representation.

On the other hand, the design goal of J-LIS is to see if the binary supervision signal can help learn structures or not. Hence, in the experimental section, we use the binary signal as supervision but evaluate the models on structured labeled data. Moreover, J-LIS supports *jointly* learning from the structured labeled data and binary labeled data, hence allows the two-way communication between the target task and the

companion task.

6.6.4 Semi-Supervised Structural SVM

The objective function Eq. (6.3) contains two loss terms corresponding to the direct and binary indirect supervision. While the form of the objective function resembles the objective function of semi-supervised structure learning [Zien et al., 2007; Brefeld and Scheffer, 2006], J-LIS is very different both conceptually and technically. The difference stems from our interpretation of the relation between the companion problems allowing us to use invalid examples ($y = -1$), which are not used by semi-supervised learning frameworks. We further discuss the impact of negative examples empirically in Sec. 6.5.1.

Our work is conceptually related to Contrastive Estimation (CE) [Smith and Eisner, 2005], where the goal is to learn a structure predictor by pushing the probability mass away from the "bad" neighbors. There are several technical differences between the two approaches. These differences are discussed in Sec. 6.5 with related experiments.

6.6.5 Generative Models

Generative models specify a joint probability over observations and the corresponding output structures. Many generative models have been proposed for structured prediction tasks [Rabiner and Juang, 1986; Eisner, 1996]. Assume that a generative model parametrized by θ, the model can be obtained through maximizing the joint likelihood $P(\mathbf{x}, \mathbf{y}|\theta)$.

J-LIS is related to generative models. It tries to separate the correct examples from the "incorrect" examples, while generative models maximize the joint probability of the current "correct" examples among all possible input-output combinations. However, there are two important differences.

First, J-LIS allows us to choose or generate negative examples. One important advantage of this property is that J-LIS can use existing labeled examples as a supervision resource. Second, while J-LIS has strong relationships to generative models, it is a *discriminative* model. On the other hand, in Section 6.5.4, we show that J-LIS in fact performs better than a generative model with exactly the same features. Investigating the relationship between J-LIS and generative models would be an interesting and important future work.

6.6.6 Multi-task Learning Framework

One might also consider J-LIS as a framework for multi-task learning [Caruana, 1997; Argyriou et al., 2007; Kato et al., 2008], given that it accepts two different tasks in one unified framework. One difference of the

J-LIS framework is that it clearly defines the relationships between two tasks. On one hand, we use Eq. (6.1) to capture the relations between the structured task and the binary task. On the other hand, most multitask learning frameworks use either shared representation [Caruana, 1997] or a well-designed regularization term to capture the relationships between different tasks [Argyriou et al., 2007; Kato et al., 2008]. Moreover, the J-LIS framework is designed for bridging a general structured task and its companion binary task, while many other multi-task framework focuses on jointly learn multiple multiclass or binary classification tasks.

6.6.7 Optimization Techniques

The idea of convex-concave procedure (CCCP) [Yuille and Rangarajan, 2003] has been used in solving many non-convex optimization objective function [Collobert et al., 2006; Yu and Joachims, 2009]. In this chapter, we show that we do not necessary need to decompose the object function to a convex function and a concave function to performance an iterative procedure that is related to the CCCP algorithm. This discovery allows us to use square-loss function and our coordinate descent method on the dual formulations of Eq. (6.8).

To best of our knowledge, this work (along with [Chang et al., 2010b]) is one of the first work that apply dual coordinate descent algorithm for structural SVM. This greatly reduce the software complexity so that we can implement Structural SVM from scratch without using existing implementations such as SVM^{light}. The advantages and disadvantages of our optimization procedure remains an open problem for the future.

6.7 Summary

This chapter studies two companion problems – structured output prediction and a binary decision problem over the structure. The key contribution of this work is the development of a discriminative joint learning framework, J-LIS, that exploits the relationship between the two problems. Consequently it can make use of easy-to-get supervision for the binary problem to improve structure prediction, where supervision is hard to get. We apply our framework to three structure learning tasks - phonetic transliteration alignment, information extraction and part-of-speech tagging and show significant empirical improvements. Interestingly, in all three domains the most significant improvement is obtained when little direct supervision is available for the structure prediction task, thus demonstrating the benefit of using our framework especially in data-poor domains, where more supervision is required.

Chapter 7

Conclusion and Discussion

Structured prediction is at the heart of natural language processing. From parsing to textual entailment, and even to learning to map human queries to meaning representations, almost all natural language processing tasks are structural. In this thesis, we examine the possibility of using indirect binary supervision signals and constraints to learn structural models. We show the effectiveness of using indirect supervision for many different tasks, especially when the number of labeled examples is limited.

In this chapter, we conclude this thesis by first reviewing several recent works that use the idea of indirect supervision. Then, we discuss possible future directions of this thesis.

7.1 Recent NLP Systems that Adopt the Idea of Indirect Supervision

Recently, the idea of using indirect supervision signals has already impacted the natural language processing community. In the followings, we list some of the recent works that adopts similar ideas (where most of the listed papers directly cite our work), and use indirect supervision signals to boost their models. We believe this line of work shows the effectiveness of indirect supervision signals and sheds some light on future work of using indirect supervision signals.

Semantic Parsing The idea of using indirect supervision have impacted the task of semantic parsing, a task at the heart of communicating with Artificial Intelligence agents (the example in Figure 1.1). The goal of a semantic parsing system is to parse a human query and generate the corresponding meaning representation. In [Clarke et al., 2010], we address this the issue of labeling the semantic parsing and propose to use the world response as an indirect binary supervision resource through interacting through interacting with a database. [1] We show that good models can be obtained using this weak binary supervision signal *without* using any labeled query-representation pairs. Our learning algorithm is able to achieve 73.2% accuracy, which is only 7% below a fully supervised model.

[1] See Section 1.1 for more discussions.

In [Liang et al., 2011], the authors follow the same setting and explore integrating more domain knowledge of the task. Surprisingly, by just using the same indirect world response, they build a system that outperforms systems trained with labeled examples (but without domain knowledge). We believe that this work strengthens our claim of the importance of indirect supervision signals.

Large scale information extraction The idea of using indirect supervision also inspires some works in the area on large scale information extraction. Never-Ending Language Learning (NELL) is a research project that attempts and create a computer system that learns over time to read the web to create a relational database. In [Carlson et al., 2010], the authors claim that adding constraints in their bootstrap learning procedure for NELL greatly improve the performance of their system.

The concept of *distance supervision* is proposed by [Mintz et al., 2009], where they use Freebase, a large semantic database of several thousand relations, to provide distant supervision for a system to find relations between entities. In [Riedel et al., 2010], the authors propose to train their graphical model by framing distant supervision as an instance of a constraint-driven semi-supervision framework. They include an additional constraint to help filter noisy feedback during their learning procedure. Their test results show that the constrained-infused model is significantly better than the distant supervision model.

Parsing Recently, the idea of using indirect supervision signals is frequently used to build better parsers. The system proposed by [Katz-Brown et al., 2011] reranks the parser by using an indirect external loss signal: word reordering performance for machine translation. They show that by using this external signals, while they cannot improve the parsing results, the performance on the reordering has improved significantly. The system [McDonald et al., 2011] cleverly "invents" constraints by projecting the labels in English into different languages to form the constraints for other languages. They build a simple but effective constraint driven learning framework. [Hall et al., 2011] proposes a multi-objective perceptron learning algorithm. The algorithm, which is conceptually similar to the algorithm proposed in Chapter 6, allows the use both direct and indirect supervision resources.

In [Spitkovsky et al., 2011], the authors propose to use punctuations as a clue to generate different constraints, and introduce these constraints in a hard EM algorithm. The framework is basically similar to the CoDL algorithm proposed in Chapter 3. Their unsupervised, constrained training algorithm achieves 59.5% accuracy on the out-of-domain data, which is more than 6% higher than the previous best results.

Sentiment Classifications Latest works on sentiment classification also embrace the idea of using multiple levels of supervision signals. Several works focus on document-level sentiment labels, sentence-level

sentiment, and their interactions [Yessenalina et al., 2010; Tackstrom and McDonald, 2011].

7.2 Future Directions

Better usage of the existing resources One natural question to ask is how to obtain indirect supervision signals. In this thesis, we mainly use human knowledge to provide these supervision signals. For instance, in Chapter 6, we point out that human can invent binary supervision signals for learning structures. Recently, many works show that there are many existing resources that we can take advantages of. For example, [McDonald et al., 2011] shows that constraints for other languages can be generated by projecting labels of English data into these languages. We believe that many existing indirect supervision resources are waiting for us to explore.

Better interactive learning protocols One strength of using indirect supervision is that it allows users to provide easy labels instead of complex structures. Building a better interactive learning interface is a critical issue, given that it allows us to take advantage of crowd sourcing sites like Amazon Turk more efficiently. [Snow et al., 2008] shows that non-expert annotators can provide pretty good labels for simple classification tasks. However, it is still not clear how to use sites like Amazon Turk to help structured tasks. We believe that indirect supervision framework could possibly provide a solution here.

Some preliminary works have been done along this direction. In [Clarke et al., 2010], we propose to use the world response as indirect supervision resources to build a query-translation model. In [Hall et al., 2011], the authors show that by just using a weak supervision signal, they can boost the parsing performance on the out-of-domain data very significantly. It would also be interesting to apply active learning [Tong and Koller, 2001] to indirect supervision frameworks.

Multiple levels of the supervision signals Supervision signals can come in many different levels. At the highest level, the full structure is the most informative but also the most expensive. The supervision resource can also be a binary signal, but the information it carries might vary. Fortunately, there are many other levels of supervision signals. For example, the annotators can provide real value feedback instead of binary feedback. One can also try to use partial labels (for example, in the POS tagging task, only certain words are labeled) as indirect supervision signals [Do and Artières, 2009]. For different forms of the supervision signals, we need to adapt the learning frameworks to benefit from these signals. Hence, it is necessary to explore this direction and invent new learning mechanisms. Another important direction is to analyze the cost-and-effect relations on different levels of supervision signals so that we can choose the

most effective one.

In this thesis, the proposed systems mostly accept two or three types of supervision signals. It would be interesting to see if we can have learning frameworks that accept more types of supervision. Importantly, having such frameworks which can accept more supervision *without* retraining the whole system would be highly desirable.

Enhancing the learning frameworks Another natural extension is to improve the learning framework. In this thesis, the proposed learning frameworks assume that most of the indirect supervision signals are perfect (but they are not). When applying indirect supervision frameworks in the large-scale datasets, it is likely to have noisy feedback. Therefore, how to enhance the learning frameworks to be robust to noise is a crucial issue to investigate.

Another direction is to exploit the effect of the different regularization terms. Many machine learning studies have been devoted to evaluating the value of using different regularization terms. However, little analysis has been done on structural learning and latent variable learning algorithms. The commonly used L2 regularization $\|\mathbf{w}\|^2$ might not be ideal for many NLP tasks, given that there are often many noisy features. Therefore, it is worthwhile to explore this issue further.

Appendix A

A.1 Derivations for Eq. (4.8)

Our goal here is to provide a general solution of the following optimization problem when $\gamma \geq 0$:

$$\min_{q \in \mathcal{Q}} D(q(\mathbf{h}), P_{\theta^t}(\mathbf{h}|\mathbf{x}); \gamma).$$

By defining $\log 0 = -\infty$ and $0 \log 0 = 0$, we can rewrite the formulation as follows:

$$\begin{aligned} \min_{q} \quad & -\textstyle\sum_{\mathbf{h}' \in \mathcal{H}(\mathbf{x})} q_{\mathbf{h}'} \log P_\theta(\mathbf{h}'|\mathbf{x}) + \gamma \sum q_{\mathbf{h}'} \log q_{\mathbf{h}'} && \text{(A.1)} \\ \text{S.T.} \quad & E_q[\mathbf{u}(\mathbf{x}, \mathbf{h})] \leq \mathbf{b}, \\ & q_{\mathbf{h}'} \geq 0, \forall \mathbf{h}' \in \mathcal{H}(\mathbf{x}), && \text{(A.2)} \\ & \textstyle\sum_{\mathbf{h}' \in \mathcal{H}(\mathbf{x})} q_{\mathbf{h}'} = 1 \end{aligned}$$

First note that each $q_{\mathbf{h}'}$ needs to be greater or equal to zero, otherwise the objective function is not well defined. However, removing linear inequalities Eq. (A.2) might still not be safe, given that the optimal value for some $q_{\mathbf{h}'}$ could be zero.

However, let us remove Eq. (A.2) first and at the end of the section, we argue that it is safe to remove Eq. (A.2) by using the standard tricks.

Without Eq. (A.2), the Lagrange function of Eq. (A.1) can be written as:

$$L = \gamma \sum q_{\mathbf{h}'} \log q_{\mathbf{h}'} - \sum_{\mathbf{h}' \in \mathcal{H}(\mathbf{x})} q_{\mathbf{h}'} \log P_\theta(\mathbf{h}'|\mathbf{x}) - \beta(\sum_{\mathbf{h}' \in \mathcal{H}(\mathbf{x})} q_{\mathbf{h}'} - 1) - \lambda^T(\mathbf{b} - E_q[\mathbf{u}(\mathbf{x}, \mathbf{h})])$$

One of the optimality conditions of q is:

$$\frac{\partial L}{\partial q_{\mathbf{h}'}} = \gamma(\log q_{\mathbf{h}'} + 1) - \log P_\theta(\mathbf{h}'|\mathbf{x}) - \beta + \lambda^T(\mathbf{b} - \mathbf{u}(\mathbf{x}, \mathbf{h}')) = 0,$$

and it follows that

$$q_{\mathbf{h}'} = \log P_\theta(\mathbf{h}'|\mathbf{x})^{\frac{1}{\gamma}} \exp^{\frac{-\lambda^T \mathbf{u}(\mathbf{x},\mathbf{h}')}{\gamma}} \exp^{\frac{\beta}{\gamma}-1}, \tag{A.3}$$

where the value of β can be obtained by enforcing the sum-to-1 constraint. Eq. (4.8) can be directly obtained from this equation.

Next, we discuss that why dropping Eq. (A.2) is safe. Note that for a specific $\hat{\mathbf{h}}$, if $P_{\theta^t}(\hat{\mathbf{h}}|\mathbf{x});\gamma) = 0$, then the corresponding $q_{\hat{\mathbf{h}}}$ needs to be zero. Otherwise, the objective function well be negative infinity. Therefore, we can remove all other $\hat{\mathbf{h}}$ such $P_{\theta^t}(\hat{\mathbf{h}}|\mathbf{x});\gamma) = 0$, and only consider $\hat{\mathbf{h}}$ such that $P_{\theta^t}(\hat{\mathbf{h}}|\mathbf{x});\gamma) > 0$. Interestingly, Eq. (A.3) implies that the answer of $q_{\hat{\mathbf{h}}}$ will always be greater than zero if $P_{\theta^t}(\hat{\mathbf{h}}|\mathbf{x});\gamma) > 0$. Therefore, the constraints are never active, so it is safe to remove Eq. (A.2).

A.2 The Correctness of Algorithm 11 and Algorithm 14

In Chapter 5 and 6, we described both Algorithm 11 and Algorithm 14 with some details omitted. In this section, we show that Algorithm 11 and Algorithm 14 are well justified. We also describe the implementation and the stopping conditions in more details. Note that Algorithm 14 can be treated as a generalization over Algorithm 11, so we mainly focus on Algorithm 14 here. We only discuss the case of using square hinge loss, as it is the main loss function we used throughout this thesis. The proof is mostly based the techniques proposed by [Tsochantaridis et al., 2005; Joachims et al., 2009].

Recall that Algorithm 14 solves the following objective function:

$$\begin{aligned}
\min_{\mathbf{w}} \quad & \tfrac{1}{2}\|\mathbf{w}\|^2 + C_1 \sum_{i\in S} \xi_i^2 + C_2 \sum_{i\in B} \xi_i^2 \\
\text{S.T.} \quad & \forall i \in S, \mathbf{h} \in \mathcal{H}(\mathbf{x}_i), \xi_i \geq \Delta(\mathbf{h}, \mathbf{h}_i) - \mathbf{w}^T \Phi_{\mathbf{h}_i,\mathbf{h}}(\mathbf{x}_i), \\
& \forall i \in B, \mathbf{h} \in \mathcal{H}(\mathbf{x}_i), \xi_i \geq 1 - z_i \mathbf{w}^T \Phi_B(\mathbf{x}_i, \mathbf{h}),
\end{aligned}$$

We copy Algorithm 14 here for the sake of easiness to read and add more descriptions in it. The details are in Algorithm 16. Note that there is a parameter ϵ that controls the stopping condition of the algorithm. Therefore, the stopping condition for the cutting-plane strategy is to ensure that there are no constraints that are ϵ-violated. In the following, we show that the algorithm stops at finite number of iterations.

The following lemma helps us build our main theorem in this section.

Lemma 1. *[Tsochantaridis et al., 2005] [Lemma 10, Corollary 13] Let A be a symmetric, positive semi-denite matrix, and dene a concave objective in $F(\alpha) = \frac{1}{2}\alpha^T A\alpha + b^T\alpha$, which we assume to be bounded from above. Assume that a solution α^0 and an optimization direction $\eta = e_r$ for a single coordinate optimization direction are given. Denote the*

Algorithm 16 Cutting plane algorithm for optimizing $A(\mathbf{w}, \mathbf{w}_t)$ in Algorithm 13 with square hinge loss. This is a detailed version of Algorithm 14. Note that we only add constraints if they are ϵ-violated.

Require: $\mathbf{w}_t$, the weight vector from t^{th} iteration
1: $\mathcal{W}_i \leftarrow \emptyset, \forall i \in S$
2: $\mathcal{V}_i \leftarrow \emptyset, \forall i \in B^-$
3: $\mathbf{h}_i^t = \arg\max_{\mathbf{h}} \mathbf{w}_t^T \Phi_B(\mathbf{x}_i, \mathbf{h}), \forall i \in B^+$
4: **repeat**
5: **for** $i \in B^-$ **do**
6: $\mathbf{h}_i^* \leftarrow \arg\max_{\mathbf{h}} \mathbf{w}^T \Phi_B(\mathbf{x}_i, \mathbf{h})$
7: **if** $1 - z_i \mathbf{w}^T \Phi_B(\mathbf{x}_i, \mathbf{h}) \geq \xi_i + \epsilon$ **then** {The constraint is ϵ-violated}
8: Add $\mathbf{h}_i^*$ to $\mathcal{V}_i$
9: **end if**
10: **end for**
11: **for** $i \in S$ **do**
12: $\mathbf{h}_i^* \leftarrow \arg\max_{\mathbf{h}} \mathbf{w}^T \Phi(\mathbf{x}_i, \mathbf{h}) + \Delta(\mathbf{h}_i, \mathbf{h})$
13: **if** $\Delta(\mathbf{h}, \mathbf{h}_i) - \mathbf{w}^T \Phi_{\mathbf{h}_i, \mathbf{h}}(\mathbf{x}_i) \geq \xi_i + \epsilon$ **then** {The constraint is ϵ-violated}
14: Add $\mathbf{h}_i^*$ to $\mathcal{W}_i$
15: **end if**
16: **end for**
17: Update $\mathbf{w}$ by solving Eq. (5.5)
18: **until** no new element is added to any $\mathcal{W}_i$ and $\mathcal{V}_i$ {No constraints are ϵ-violated}
19: return $\mathbf{w}$

step size $\beta \geq 0$. Then, the objective function improves at least by

$$F(\alpha^0 + \beta e_r) - F(\alpha^0) \geq \frac{1}{2} \frac{\frac{\partial F}{\partial \alpha_r}(\alpha^0)^2}{A_{rr}},$$

where α_r represents a single variable that associated with e_r, and A_{rr} represents the r-th diagonal element of the matrix A.

First, notice that dual formulation of Eq. (A.4) can be exactly be written down in the form of $F(\alpha)$. In our formulation, each variable α is associated with a pair (i, q) to indicate its association to the q-th structure of the i-th example. Also notice that in Algorithm 16, we might choose more than one new structure to add at each iteration. Fortunately, jointly optimizing over a set of variables that include $\alpha_{i,q}$ can only further increase the value of the dual objective.

Proposition 1. *Define $R_s \equiv \max_{i,q} \|\Phi_{\mathbf{h}_i, \mathbf{h}_{i,q}}(\mathbf{x}_i)\|$ and $R_b \equiv \max_{i,q} \|\Phi_B(\mathbf{x}_i, \mathbf{h}_{i,q})\|^2$, then in Algorithm 16, in each iteration, the objective value of dual of Eq.* (A.4) *improves at least*

$$\min\{\frac{1}{2} \frac{\epsilon^2}{R_s^2 + \frac{1}{2C_1}}, \frac{1}{2} \frac{\epsilon^2}{R_b^2 + \frac{1}{2C_2}}\}.$$

Proof.

Assume that $e_{i,q}$ is a single coordinate direction along with $\alpha_{i,q}$, it is trivial to show that if $i \in S$, then

$$\frac{\partial F}{\partial \alpha_{i,q}} = \Delta(\mathbf{h}_i, \mathbf{h}_{i,q}) - \mathbf{w}^T \Phi_{\mathbf{h}_i, \mathbf{h}_{i,j}}(\mathbf{x}_i) - \xi_i \geq \epsilon.$$

Hence,

$$F(\alpha^0 + \beta e_{i,q}) - F(\alpha^0) \geq \frac{1}{2} \frac{\epsilon^2}{\|\Phi_{\mathbf{h}_i, \mathbf{h}_{i,q}}(\mathbf{x}_i)\|^2 + \frac{1}{2C_1}}.$$

Similar, if $i \in B$, then

$$F(\alpha^0 + \beta e_{i,q}) - F(\alpha^0) \geq \frac{1}{2} \frac{\epsilon^2}{\|\Phi_B(\mathbf{x}_i, \mathbf{h}_{i,q})\|^2 + \frac{1}{2C_2}}.$$

Note that in Algorithm 16, we might choose more than one new cutting plane to add at each iteration, but jointly optimizing over a set of variables that include $\alpha_{i,q}$ can only further increase the value of the dual objective. Therefore, at each iteration, the improvement over the cutting plane method is at least

$$\min\{\frac{1}{2} \frac{\epsilon^2}{R_s^2 + \frac{1}{2C_1}}, \frac{1}{2} \frac{\epsilon^2}{R_b^2 + \frac{1}{2C_2}}\}.$$

□

Theorem 4. *The number of iterations of Algorithm 16 is bounded by*

$$\frac{|S|C_1 + |B|C_2}{\min\{\frac{1}{2} \frac{\epsilon^2}{R_s^2 + \frac{1}{2C_1}}, \frac{1}{2} \frac{\epsilon^2}{R_b^2 + \frac{1}{2C_2}}\}}$$

Proof. The theorem follows by observing the dual gap of (A.4) can be bounded by $|S|C_1 + |B|C_2$. □

References

E. L. Allwein, R. E. Schapire, and Y. Singer. Reducing multiclass to binary: a unifying approach for margin classifiers. In *Proceedings of the 17th International Workshop on Machine Learning*, pages 9–16, 2000.

A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In *The Conference on Advances in Neural Information Processing Systems (NIPS)*. MIT Press, 2007.

R. Barzilay and M. Lapata. Aggregation via Set Partitioning for Natural Language Generation. In *Proc. of HLT/NAACL*, June 2006.

K. Bellare, G. Druck, and A. McCallum. Alternating projections for learning with expectation constraints. In *Proceedings of Uncertainty in Artificial Intelligence (UAI)*, 2009.

L. Bentivogli, I. Dagan, H. T. Dang, D. Giampiccolo, and B. Magnini. The fifth PASCAL recognizing textual entailment challenge. In *Proc. of TAC Workshop*, pages 14–24, 2009.

S. Bergsma and G. Kondrak. Alignment-based discriminative string similarity. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 656–663, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P/P07/P07-1083`.

C. M. Bishop. *Neural networks for pattern recognition*. Oxford University Press, Oxford, UK, 1996. ISBN 0-19-853849-9.

A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Annual ACM Workshop on Computational Learning Theory (COLT)*, pages 92–100, 1998.

S. Borman. The expectation maximization algorithm – a short tutorial. Introduces the Expectation Maximization (EM) algorithm and fleshes out the basic mathematical results, inclusing a proof of convergence. The Generalized EM algorithm is also introduced., July 2004.

L. Bottou. Stochastic learning. In Olivier Bousquet and Ulrike von Luxburg, editors, *Advanced Lectures on Machine Learning*, Lecture Notes in Artificial Intelligence, LNAI 3176, pages 146–168. Springer Verlag, Berlin, 2004.

S. Boyd and A. Mutapcic. Stochastic subgradient methods, 2008. Notes for EE364b, Stanford University, Winter 2006-2007.

U. Brefeld and T. Scheffer. Semi-supervised learning for structured output variables. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2006.

C. Brockett. Aligning the RTE 2006 corpus. In *Technical Report MSR-TR-2007-77, Microsoft Research*, 2007.

A. Carlson, J. Betteridge, R. C. Wang, E. R. Hruschka Jr., and T. M. Mitchell. Coupled semi-supervised learning for information extraction. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, 2010.

R. Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

Y. Censor and S. A. Zenios. *Parallel Optimization — Theory, Algorithms, and Applications*. Oxford University Press, 1997.

M. Chang, L. Ratinov, and D. Roth. Guiding semi-supervision with constraint-driven learning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 280–287, Prague, Czech Republic, Jun 2007. Association for Computational Linguistics. URL `http://l2r.cs.uiuc.edu/~danr/Papers/ChangRaRo07.pdf`.

M. Chang, L. Ratinov, N. Rizzolo, and D. Roth. Learning and inference with constraints. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 1513–1518, July 2008a. URL `http://l2r.cs.uiuc.edu/~danr/Papers/CRRR08.pdf`.

M. Chang, L. Ratinov, and D. Roth. Constraints as prior knowledge. In *ICML Workshop on Prior Knowledge for Text and Language Processing*, pages 32–39, July 2008b. URL `http://l2r.cs.uiuc.edu/~danr/Papers/ChangRaRo08.pdf`.

M. Chang, D. Goldwasser, D. Roth, and Y. Tu. Unsupervised constraint driven learning for transliteration discovery. In *Proceedings of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*, May 2009. URL `http://l2r.cs.uiuc.edu/~danr/Papers/CGRT09.pdf`.

M. Chang, D. Goldwasser, D. Roth, and V. Srikumar. Discriminative learning over constrained latent representations. In *Proceedings of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*, Jun 2010a. URL `http://l2r.cs.uiuc.edu/~danr/Papers/CGRS10.pdf`.

M. Chang, V. Srikumar, D. Goldwasser, and D. Roth. Structured output learning with indirect supervision. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2010b. URL `http://l2r.cs.uiuc.edu/~danr/Papers/CSGR10.pdf`.

E. Charniak and M. Johnson. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 173–180, Ann Arbor, Michigan, 2005. ACL.

C. Cherry and C. Quirk. Discriminative, syntactic language modeling through latent svms. In *Proc. of the Eighth Conference of AMTA*, Honolulu, Hawaii, October 2008.

J. Clarke and M. Lapata. Constraint-based sentence compression: An integer programming approach. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 144–151, Sydney, Australia, July 2006. ACL.

J. Clarke, D. Goldwasser, M. Chang, and D. Roth. Driving semantic parsing from the world's response. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning (CoNLL-2010)*, July 2010. URL `http://l2r.cs.uiuc.edu/~danr/Papers/CGCR10.pdf`.

W. Cohen. Exploiting dictionaries in named entity extraction: Combining semi-markov extraction processes and data integration methods. In *Proceedings of International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 89–98, 2004.

W. Cohen and S. Sarawagi. Exploiting dictionaries in named entity extraction: Combining semi-markov extraction processes and data integration methods. In *Proc. of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004.

M. Collins. Discriminative reranking for natural language parsing. In *Proc. 17th International Conf. on Machine Learning*, pages 175–182. Morgan Kaufmann, San Francisco, CA, 2000.

M. Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, 2002.

M. Collins and Y. Singer. Unsupervised models for named entity classification. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, 1999.

M. Collins, A. Globerson, T. Koo, X. Carreras, and P. L. Bartlett. Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *Journal of Machine Learning Research*, 9, 2008.

K. Collins-thompson. Reducing the risk of query expansion via robust constrained optimization. In *Proceedings of ACM Conference on Information and Knowledge Management (CIKM)*, pages 837–846, 2009. doi: 10.1145/1645953.1646059.

R. Collobert, F. Sinz, J. Weston, and L. Bottou. Trading convexity for scalability. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2006.

F. G. Cozman, I. Cohen, and M. C. Cirelo. Semi-supervised learning of mixture models. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 99–106, 2003.

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, March 2002. ISSN 1532-4435.

K. Crammer, R. Mcdonald, and F. Pereira. Scalable large-margin online learning for structured classification. Technical report, Department of Computer and Information Science, University of Pennsylvania, 2005.

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 2006.

I. Dagan, O. Glickman, and B. Magnini, editors. *The PASCAL Recognising Textual Entailment Challenge.*, volume 3944, 2006. Springer-Verlag, Berlin.

D. Das and S. Petrov. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, Portland, OR, June 2011. Association for Computational Linguistics.

D. Das and N. A. Smith. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, August 2009.

H. Daumé and D. Marcu. Learning as search optimization: Approximate large margin methods for structured prediction. In *Proceedings of the International Conference on Machine Learning (ICML)*, Bonn, Germany, 2005. URL `http://pub.hal3.name/#daume05laso`.

Hal Daumé III. Cross-task knowledge-constrained self training. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 680–688, Honolulu, Hawaii, October 2008. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D08-1071`.

R. Dechter and R. Mateescu. Mixtures of deterministic-probabilistic networks and their AND/OR search space. In *Proceedings of AUAI*, pages 120–129, Arlington, Virginia, United States, 2004. AUAI Press. ISBN 0-9749039-0-6.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39:1–38, 1977.

P. Denis and J. Baldridge. Joint determination of anaphoricity and coreference resolution using integer programming. In *Proceedings of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*, 2007.

Q. Do, D. Roth, M. Sammons, Y. Tu, and V. Vydiswaran. Robust, light-weight approaches to compute lexical similarity. Technical report, Computer Science Department, University of Illinois, 2009. URL `http://cogcomp.cs.illinois.edu/papers/DRSTV09.pdf`.

T. Do and T. Artières. Large margin training for hidden markov models with partially observed states. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 265–272, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-516-1.

W. Dolan, C. Quirk, and C. Brockett. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings the International Conference on Computational Linguistics (COLING)*, 2004.

P. Domingos and M. Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29:103–130, 1997.

J. Eisner. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings the International Conference on Computational Linguistics (COLING)*, pages 340–345, Copenhagen, August 1996. URL `http://cs.jhu.edu/~jason/papers/#coling96`.

R.-E. Fan and C.-J. Lin. Libsvm data. URL `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`.

P. Felzenszwalb, D. Mcallester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR) Anchorage, Alaska, June 2008.*, June 2008. URL `http://www.ics.uci.edu/~dramanan/papers/latent.pdf`.

P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(1), 2009.

J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 363–370, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

Y. Freund and R. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999.

K. Ganchev, J. V. Graca, and B. Taskar. Better alignments = better translations. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2008.

K. Ganchev, J. Graça, J. Gillenwater, and B. Taskar. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 2010.

R. Ge and R. Mooney. A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the Annual Conference on Computational Natural Language Learning (CoNLL)*, pages 9–16, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W/W05/W05-0602`.

D. Goldwasser and D. Roth. Active sample selection for named entity transliteration. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, Columbus, OH, USA, Jun 2008. Association for Computational Linguistics. URL `http://l2r.cs.uiuc.edu/~danr/Papers/GoldwasserRo08.pdf`. Short Paper.

J. V. Graca, K. Ganchev, and B. Taskar. Expectation maximization and posterior constraints. In *NIPS*, volume 20, 2007.

T. Grenager, D. Klein, and C. Manning. Unsupervised learning of field segmentation models for information extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2005.

A. Haghighi and D. Klein. Prototype-driven learning for sequence models. In *Proc. of HTL-NAACL*, 2006.

A. Haghighi and D. Klein. Coreference resolution in a modular, entity-centered model. In *Proceedings of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*, pages 385–393, Los Angeles, California, June 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N10-1061.

A. Haghighi, A. Ng, and C. Manning. Robust textual inference via graph matching. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 387–394, Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/H/H05/H05-1049.

K. Hall, R. McDonald, J. Katz-Brown, and Mi. Ringgaard. Training dependency parsers by jointly optimizing multiple objectives. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 1489–1499, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics.

S. Har-Peled, D. Roth, and D. Zimak. Constraint classification: a new approach to multiclass classification. In *Proceedings of the International Workshop on Algorithmic Learning Theory (ALT)*, pages 135–150. Springer-Verlag, 2002. URL http://l2r.cs.uiuc.edu/~danr/Papers/Har-PeledRoZi02.pdf.

G.E. Hinton. Products of experts. In *Proc. of the 9th International Conference on Artificial Neural Networks (ICANN99)*, pages 1–6, 1999. URL citeseer.ist.psu.edu/hinton99product.html.

T. Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 2001.

C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear svm. In *Proceedings of the International Conference on Machine Learning (ICML)*, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4. doi: http://doi.acm.org/10.1145/1390156.1390208.

C.-w. Hsu and C.-j. Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 2002.

L. Huang. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2008.

T. Joachims, T. Finley, and Chun-Nam Yu. Cutting-plane training of structural svms. *Machine Learning*, 77 (1):27–59, 2009.

M. Johnson. Why doesn't EM find good HMM POS-taggers? In *Proceedings of the 2007 Joint Conference of EMNLP-CoNLL*, pages 296–305, 2007. URL http://www.aclweb.org/anthology/D/D07/D07-1031.

T. Kato, H. Kashima, M. Sugiyama, and K. Asai. Multi-task learning via conic programming. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *The Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 737–744. MIT Press, Cambridge, MA, 2008.

J. Katz-Brown, S. Petrov, R. McDonald, F. Och, D. Talbot, H. Ichikawa, M. Seno, and H. Kazawa. Training a parser for machine translation reordering. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 183–192, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics.

J. Kazama and K. Torisawa. A new perceptron algorithm for sequence labeling with non-local features. In *Proceedings of the 2007 Joint Conference of EMNLP-CoNLL*, pages 315–324, 2007. URL http://www.aclweb.org/anthology/D/D07/D07-1033.

M. Kearns, Y. Mansour, and A. Y. Ng. An information-theoretic analysis of hard and soft assignment methods for clustering. In *Proceedings of the NATO Advanced Study Institute on Learning in graphical models*, pages 495–520, Norwell, MA, USA, 1998. Kluwer Academic Publishers.

J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 1952.

J. Kivinen and M. K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. In *ACM Symp. of the Theory of Computing*, 1995.

J. Kivinen and M. K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 1997.

D. Klein and C. D. Manning. Fast exact inference with a factored model for natural language parsing. In *The Conference on Advances in Neural Information Processing Systems (NIPS)*, volume 15. MIT Press, 2003.

D. Klein and C. D. Manning. Corpus-based induction of syntactic structure: models of dependency and constituency. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

A. Klementiev and D. Roth. Named entity transliteration and discovery in multilingual corpora. In Cyril Goutte, Nicola Cancedda, Marc Dymetman, and George Foster, editors, *Learning Machine Translation*. MIT Press, 2008. URL `http://l2r.cs.uiuc.edu/~danr/Papers/KlementievRo08.pdf`.

P. Koehn. Europarl: A multilingual corpus for evaluation of machine translation. 2002.

T. Koo, A. M. Rush, M. Collins, T. Jaakkola, and D. Sontag. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 1288–1298, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2001.

P. Lambert, A. De Gispert, R. Banchs, and J. Marino. Guidelines for word alignment evaluation and manual alignment. *Language Resources And Evaluation*, 2006.

K. Lang. NewsWeeder: learning to filter netnews. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 331–339, 1995.

Y. Le Cun, L. Bottou, Y. Bengio, and P. Haffner. Gradient based learning applied to document recognition. *Proceedings of IEEE*, 86(11):2278–2324, 1998. URL `http://leon.bottou.org/papers/lecun-98h`.

P. Liang and D. Klein. Analyzing the errors of unsupervised learning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2008.

P. Liang, B. Taskar, and D. Klein. Alignment by agreement. In *Proceedings of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*, pages 104–111, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

P. Liang, M. I. Jordan, and D. Klein. Learning from measurements in exponential families. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2009.

P. Liang, M. I. Jordan, and D. Klein. Learning dependency-based compositional semantics. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2011.

B. MacCartney, M. Galley, and C. D. Manning. A phrase-based alignment model for natural language inference. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 802–811, Honolulu, Hawaii, October 2008. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D08-1084`.

G. Mann and A. McCallum. Generalized expectation criteria for semi-supervised learning of conditional random fields. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, number 870 - 878, 2008.

T. Marciniak and M. Strube. Beyond the pipeline: Discrete optimization in NLP. In *Proceedings of the Annual Conference on Computational Natural Language Learning (CoNLL)*, pages 136–143, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W05/W05-0618.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*.

A. Martins, N. A. Smith, and E. Xing. Concise integer linear programming formulations for dependency parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, August 2009a.

A. F. T. Martins, N. A. Smith, and E. P. Xing. Polyhedral outer approximations with application to natural language parsing. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 713–720, New York, NY, USA, 2009b. ACM. ISBN 978-1-60558-516-1. doi: http://doi.acm.org/10.1145/1553374.1553466.

A. McCallum, D. Freitag, and F. Pereira. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2000.

A. McCallum, K. Bellare, and F. Pereira. A conditional random field for discriminatively-trained finite-state string edit distance. In *Proceedings of Uncertainty in Artificial Intelligence (UAI)*, 2005.

Andrew K. McCallum, Ronald Rosenfeld, Tom M. Mitchell, and Andrew Y. Ng. Improving text classification by shrinkage in a hierarchy of classes. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 359–367, 1998.

D. McClosky, E. Charniak, and M. Johnson. Effective self-training for parsing. In *Proceedings of HLT-NAACL*, 2006.

R. McDonald, S. Petrov, and K. Hall. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 62–72, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D11-1006.

B. Merialdo. Tagging text with a probabilistic model. *Computational Linguistics*, 20(2):155–172, 1994.

M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2009.

R. M. Neal and G. E. Hinton. A new view of the EM algorithm that justifies incremental, sparse and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. Kluwer Academic Publishers, 1998.

A. Ng and M. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Proceedings of NIPS 14*, 2002.

A. Y. Ng and M. I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *The Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 841–848, 2001.

K. Nigam, A. Mccallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.

N. J. Nilsson. *Learning Machines*. McGraw-Hill, New York, 1965.

F. J. Och and H. Ney. Improved statistical alignment models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2000.

J. Pasternack and D. Roth. Learning Better Transliterations. In *The 18th ACM Conference on Information and Knowledge Management (CIKM)*, November 2009. URL http://l2r.cs.uiuc.edu/~danr/Papers/PasternackRo09a.pdf.

J. C. Platt. Fast training of support vector machines using sequential minimal optimization. *Advances in kernel methods: support vector learning*, 1999.

V. Punyakanok, D. Roth, and W. Yih. The necessity of syntactic parsing for semantic role labeling. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1117–1123, 2005a. URL http://l2r.cs.uiuc.edu/~danr/Papers/PunyakanokRoYi05.pdf.

V. Punyakanok, D. Roth, W. Yih, and D. Zimak. Learning and inference over constrained output. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1124–1129, 2005b. URL http://l2r.cs.uiuc.edu/~danr/Papers/PRYZ05.pdf.

V. Punyakanok, D. Roth, and W. Yih. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287, 2008. URL http://l2r.cs.uiuc.edu/~danr/Papers/PunyakanokRoYi07.pdf.

L. Qiu, M.-Y. Kan, and T.-S. Chua. Paraphrase recognition via dissimilarity significance classification. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 18–26, 2006.

C. Quirk, C. Brockett, and W. Dolan. Monolingual machine translation for paraphrase generation. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 142–149, 2004.

L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, 1989.

L. R. Rabiner and B. H. Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, 3(1):4–16, January 1986.

M. Richardson and P. Domingos. Markov logic networks. *Machine Learning Journal*, 62(1-2):107–136, 2006. ISSN 0885-6125. doi: http://dx.doi.org/10.1007/s10994-006-5833-1.

S. Riedel and J. Clarke. Incremental integer linear programming for non-projective dependency parsing. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 129–137, Sydney, Australia, 2006. URL http://aclweb.org/anthology-new/W/W06/W06-1616.pdf.

S. Riedel, L. Yao, and A. McCallum. Modeling relations and their mentions without labeled text. In *Proceedings of the European Conference on Machine Learning (ECML)*, pages 148–163, 2010.

N. Rizzolo and D. Roth. Modeling Discriminative Global Inference. In *Proceedings of the First International Conference on Semantic Computing (ICSC)*, pages 597–604, Irvine, California, September 2007. IEEE. URL http://l2r.cs.uiuc.edu/~danr/Papers/RizzoloRo07.pdf.

K. Rose. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. In *Proceedings of the IEEE*, pages 2210–2239, 1998.

K. Rose, E. Gurewwitz, and G. Fox. A deterministic annealing approach to clustering. *Pattern Recogn. Lett.*, 11:589–594, September 1990. ISSN 0167-8655.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psych. Rev.*, 65:386–407, 1958. (Reprinted in *Neurocomputing* (MIT Press, 1988).).

F. Rosenblatt. *Principles of Neurodynamics*. Spartan, New York, 1962.

D. Roth. Learning in natural language. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 898–904, 1999. URL http://l2r.cs.uiuc.edu/~danr/Papers/ijcai99r.pdf.

D. Roth and W. Yih. A linear programming formulation for global inference in natural language tasks. In Hwee Tou Ng and Ellen Riloff, editors, *Proceedings of the Annual Conference on Computational Natural Language Learning (CoNLL)*, pages 1–8. Association for Computational Linguistics, 2004. URL http://l2r.cs.uiuc.edu/~danr/Papers/RothYi04.pdf.

D. Roth and W. Yih. Integer linear programming inference for conditional random fields. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 737–744, 2005. URL http://l2r.cs.uiuc.edu/~danr/Papers/RothYi05.pdf.

D. Roth and W. Yih. Global inference for entity and relation identification via a linear programming formulation. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2007. URL http://l2r.cs.uiuc.edu/~danr/Papers/RothYi07.pdf.

D. Roth, M. Sammons, and V.G. Vydiswaran. A framework for entailed relation recognition. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, Singapore, August 2009. Association for Computational Linguistics. URL http://L2R.cs.uiuc.edu/~danr/Papers/RothSaVy09.pdf.

A. M. Rush, D. Sontag, M. Collins, and T. Jaakkola. On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, 2010.

S. Sarawagi and W. Cohen. Semi-markov conditional random fields for information extraction. In *The Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 1185–1192, 2004.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: primal estimated sub-gradient solver for SVM. In Zoubin Ghahramani, editor, *Proceedings of the International Conference on Machine Learning (ICML)*, pages 807–814. Omnipress, 2007.

N. Smith and J. Eisner. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2005.

N. A. Smith and J. Eisner. Annealing techniques for unsupervised statistical language learning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2004.

N. A. Smith and J. Eisner. Annealing structural bias in multilingual weighted grammar induction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, ACL-44, pages 569–576, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

R. Snow, B. O'Connor, D. Jurafsky, and A. Ng. Cheap and fast – but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 254–263, Honolulu, Hawaii, October 2008. Association for Computational Linguistics. URL http://www.aclweb.org/anthology-new/D/D08/D08-1027.bib.

D. Sontag. *Approximate Inference in Graphical Models using LP Relaxations*. PhD thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 2010.

V. I. Spitkovsky, H. Alshawi, D. Jurafsky, and C. D. Manning. Viterbi training improves unsupervised dependency parsing. In *Proceedings of the Annual Conference on Computational Natural Language Learning (CoNLL)*, 2010.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. Punctuation: Making a point in unsupervised dependency parsing. In *Proceedings of the Annual Conference on Computational Natural Language Learning (CoNLL)*, 2011.

O. Tackstrom and R. McDonald. Discovering fine-grained sentiment with latent variable structured prediction models. In *Proceedings of the 33rd European conference on Advances in information retrieval*, pages 368–374, Berlin, Heidelberg, 2011. Springer-Verlag.

L. R. Tang and R. J. Mooney. Using multiple clause constructors in inductive logic programming for semantic parsing. In *Proceedings of the European Conference on Machine Learning (ECML)*, 2001.

B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *The Conference on Advances in Neural Information Processing Systems (NIPS)*, 2004.

M. Thelen and E. Riloff. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, 2002.

S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, November 2001. Full version of paper in ICML 2000.

K. Toutanova, A. Haghighi, and C. D. Manning. Joint learning improves semantic role labeling. In *Proceedings of ACL 2005*, 2005.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, September 2005.

N. Ueda and R. Nakano. Deterministic annealing em algorithm. *Neural Network*, 1998. ISSN 0893-6080.

A. F. Veinott and G. B. Dantzig. Integral extreme points. Technical report, Stanford University, 1968. Also in SIAM Review 10, pp. 371372.

S. Wan, M. Dras, R. Dale, and C. Paris. Using dependency-based features to take the p̈ara-farceöut of paraphrase. In *Proc. of the Australasian Language Technology Workshop (ALTW)*, 2006.

P. Wolfe. Convergence conditions for ascent methods. *Siam Review*, 11, 1969. doi: 10.1137/1011036.

Y.-W. Wong and R. Mooney. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 960–967, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P/P07/P07-1121`.

Z. Wu and M. Palmer. Verb semantics and lexicon selection. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 133–138, 1994.

D. Yarowsky. Unsupervised word sense disambiguation rivaling supervied methods. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 1995.

A. Yessenalina, Y. Yue, and C. Cardie. Multi-level structured models for document-level sentiment classification. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, 2010.

C. Yu and T. Joachims. Learning structural svms with latent variables. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2009.

A. L. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 15(4), 2003.

F. M. Zanzotto and A. Moschitti. Automatic learning of textual entailments with cross-pair similarities. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, July 2006.

J. M. Zelle and R. J. Mooney. Learning to parse database queries using inductive logic proramming. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 1050–1055, 1996.

L. Zettlemoyer and M. Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of Uncertainty in Artificial Intelligence (UAI)*, 2005.

L. Zettlemoyer and M. Collins. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference of EMNLP-CoNLL*, pages 678–687, 2007. URL http://www.aclweb.org/anthology/D/D07/D07-1071.

T. Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the International Conference on Machine Learning (ICML)*, page 116, New York, NY, USA, 2004. ACM. ISBN 1-58113-828-5. doi: http://doi.acm.org/10.1145/1015330.1015332.

A. Zien, U. Brefeld, and T. Scheffer. Transductive support vector machines for structured variables. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2007.

Author's Biography

Ming-Wei Chang is a Ph.D. candidate in University of Illinois at Urbana-Champaign. He enjoys the research area of machine learning and its application to natural language processing and information retrieval. In 2009 and 2010, he has co-presented tutorials on combining human knowledge with statistical models in EACL and NAACL, respectively. Before becoming a Ph.D. student, Ming-Wei has worked on several support vector machine projects, and has won the first place in two international machine learning competitions.