Entity Discovery by Exploiting Contextual Structures

CHAN, Shing Kit

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Doctor of Philosophy
in
Systems Engineering and Engineering Management

The Chinese University of Hong Kong
August 2011

Thesis/Assessment Committee

Professor Wai Lam (Thesis Supervisor)
Professor Kai Pui Lam (Chair)
Professor Jeffrey Yu (Committee Member)

Abstract

In text mining, being able to recognize and extract named entities, e.g. Locations, Persons, Organizations, is very useful in many applications. This is usually referred to named entity recognition (NER). This thesis presents a cascaded framework for extracting named entities from text documents. We automatically derive features on a set of documents from different feature templates. To avoid high computational cost incurred by a single-phase approach, we divide the named entity extraction task into a segmentation task and a classification task, reducing the computational cost by an order of magnitude.

To handle cascaded errors that often occur in a sequence of tasks, we investigate and develop three models: maximum-entropy margin-based (MEMB) model, isomeric conditional random field (ICRF) model, and online cascaded reranking (OCR) model. MEMB model makes use of the concept of margin in maximizing log-likelihood. Parameters are trained in a way that they can maximize the "margin" between the decision boundary and the nearest training data points. ICRF model makes use of the concept of joint training. Instead of training each model independently, we design the segmentation and classification models in a way that they can be efficiently trained together under a soft constraint. OCR model is developed by using an online training method to maximize a margin without considering any probability measures, which greatly reduces the training time. It reranks all of the possible outputs from a previous stage based on a total output score. The best output with the highest total score is the final output.

We report experimental evaluations on the GENIA Corpus available from the BioNLP/NLPBA (2004) shared task and the Reuters Corpus available from the CoNLL-2003 shared tasks, which demonstrate the state-of-the-art performance achieved by the proposed models.

摘要

在文本挖掘中，能夠識別和提取實體名稱，例如：地點， 個人，組織，在許多應用中非常有用。這 通常被稱為命名實體識別（ＮＥＲ）。本論文提出 一個梯級框架去從文本中提取實體名稱。 我們會自動用特點 模板去取得文件中的不同特點。為了避免一次性運算所涉及的龐大運算成本， 我們將命名實體提取任務劃分為分割 任務和分類任務，減少了運算成本的數級。

為了處理序列任務中常常發生的梯級錯誤，我們探討和 發展出三種模型：最大熵餘邊基礎（ＭＥＭＢ） 模型，同分異構體條件隨機場（ＩＣＲＦ）模型，在線梯級 重排序（ＯＣＲ）模型。ＭＥＭＢ模型是在最大熵模型中加入了餘邊的概念， 將對數似然最大化。參數進行培訓的目的，是為了將邊界和最近的訓練數據點的餘邊最大化。 ＩＣＲＦ模型是運用聯合訓練的概念， 每個任務的訓練不是獨立的，我們將分割模型 和分類模型的訓練，設計成可以在一個軟約束下一起有效地訓練 。ＯＣＲ模型是運用在線 訓練方法，將餘邊最大化，而不考慮任何或然率， 大大減少了訓練時

間。在某一階段中，它重排所有從上一個階段輸出的可能。 最後，用總得分最高的答案作為最終的輸出。

我們報告了在ＧＥＮＩＡ語料庫作實驗的結果，這一語料庫可 在ＢｉｏＮＬＰ／ＮＬＰＢＡ（２００４年）的共同任務中取得。我們也報告了在路透社語料庫作實驗的結果， 此語料庫可從ＣｏＮＬＬ－２００３的共享任務中取得。實驗結果展現了目前最先進模型的性能表現。

Acknowledgements

First and foremost I place on record my deep sense of gratitude to my supervisor, Professor Wai Lam, who supported me throughout my thesis with his patience and knowledge, and at the same time allowing me the room to work in my own way. He introduced me to the field and helped me in every possible way. I want to thank him for the years full of advice on research and academic matters alike. I really enjoyed the freedom that he gave me; besides text mining, there was opportunity to work on several other topics as well (algorithms, statistics and programming). I attribute the level of my PhD degree to his encouragement and effort; without him, this thesis could not have been completed or written.

IT technicians in the Information Systems (IS) laboratory aided my work over many years, related to software, as well as hardware. I thank Phyllis Leung, a fine computer technician who helped me in many computer-related problems.

I was interested in using machine learning techniques for text mining problems but somehow I was not satisfied with what I had learned in textbooks and other classes. One afternoon I attended Professor Lei Xu's lecture on Computational Intelligence on using data and a probabilistic framework to make the best decision. I was sold immediately on his insights. My sincere thanks to Prof Xu for having imparted clarity to my thinking and understanding.

I am grateful to all professors, staff and my schoolmates in the CUHK. All of them (Gatien, Kaka, Cecia and Wu Di) gave me genuine and helpful advice on every problem that I encountered. I would also like to thank Eric W. L. Ng, a technician in the University of Hong Kong, for help he provided for getting into a PhD programme.

Finally, of all the people who have helped me with my thesis, there were only two who at times really had to suffer: my mother did not understand the topic but remained open and supportive, and my girlfriend hardly complained and remained loving and lovely as ever. I am really grateful to them for that.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In text mining, the vast amount of data and the great variety of induced features are two major bottlenecks when looking for important information from text data. This thesis focuses on the task of extracting information from sequence data. For example, a noun-phrase (NP) chunking task may involve a part-of-speech (POS) tagging task. These two tasks are usually cascaded, i.e. a POS tagger is used to get the POS tag of every word, followed by a noun-phrase chunker to segment and identify a NP chunk.

Named entity recognition involves processing structured or unstructured documents and identifying a sequence of words that refers to named entities such as people, places, and organizations. It is a fundamental task in natural language processing (NLP). NER tasks were first introduced in the sixth Message Understanding Conference (MUC-6) and quickly evoked interests in the NLP community. NER tasks have also found mention in the Conference on Computational Natural Language Learning (CoNLL) in recent years. These conferences provided the benchmark for named entity systems in many information extraction tasks.

NER is considered to be a simple problem for human beings. It is intuitively simple that proper names can be identified with ease. However, it is hard for a machine to recognize proper names in an unstructured text. One may think that words having initial letter in capital are proper nouns; this is right but only to a limited extent. One may also think that using a dictionary one can identify most of the named entities because they are proper nouns. This is wrong because named entities evolve continuously as time passes by; it is impossible to add all named entities or proper nouns into a dictionary. As human beings we do not need a dictionary to recognize an unseen building with a name such as "The Fortune Building".

The difficult issues in NER often involve semantic ambiguity. A proper noun may have different meanings under different contexts. Consider the phrase "The Pentagon". When should it be considered a location, and when is it an organization? When should we treat "May" as a month name, and when as a person name? It is not easy to decide the sense without looking at the whole sentence.

Extracting named entities automatically is useful in many important applications such as information retrieval, question answering, and machine translation systems. In biological text archives, a named entity recognition system can automatically extract important medical named entities such as DNAs, RNAs, and proteins. In a question answering system, named entities are the key to directly answering questions that involve "who" and "where" and helpful in indirectly answering questions that involve "what" and "when".

While the most straightforward and easiest way to build a NER system is to directly process each feature and entity type, it also comes with a huge computational cost. For example, a direct application of a conditional random field (CRF) requires $O(N_a N_c^2)$ time for each iteration in training, where N_c is the number of entity types and N_a is the number of features, which is usually in the range of 1 to 10 millions. Moreover, many probability models use exponential families to estimate the probability for each training instance. A direct use of each feature means that we have to perform this expensive exponent operation for each feature.

Typically a NER system has two sub-tasks: the first is identification of phrases in unstructured text, and the second is classification of these phrases into a set of predefined categories of interest, such as person names, organization names, and locations. Organization names may include companies, committees, and associations. Locations may include cities, countries, streets, and buildings.

Generally speaking, in many real world problems, information extraction tasks are cascaded together to achieve a final goal. Recently there has been a lot of related research into this area [44] [15] [17]. In a large-scale data mining or information extraction system, a major task is divided into two or more sub-tasks. Each task aims at a specific goal to accomplish one part of the major task. For example, an initial task of a noun-phrase chunking system may be to get the part-of-speech (POS) tag of every word. The result of a sub-task is passed on to the next sub-task for processing.

Usually the initial tasks are responsible for generating some low-level features for representing an instance. They may be in the form of attribute-value vectors or classification categories. The reason for doing this is to turn the complex knowledge, which is very rich in its original form, into a simpler form that an information extraction system can handle.

"Cascaded errors" often occur in a sequence of tasks. This refers to an error made in the early stage of a system that is passed on to the next stage for processing. Based on this incorrect result, the next stage of the system produces another incorrect result, which is passed on to the next stage. In this way, an error is amplified in a chain of sub-tasks.

Generally, in order to avoid cascaded errors, one can formulate the problem such that different sub-tasks are solved together. This approach is suitable for small problems where the search space of combined sub-tasks is tractable during training. This totally eliminates the possibility of cascaded errors because it tries to assign values to variables simultaneously. However, this approach is computationally very expensive because usually the search space is exponentially increased. Therefore some approximate inference algorithms [46] [53] [34] have to be applied to reduce the inference time. Other approximate inference algorithms include Markov chains Monte Carlo (MCMC) sampling [16] and Gibbs sampling [14].

1.1 A Motivating Example

In this section, we illustrate the idea of cascaded approach in the context of a noun-phrase chunking problem, modeled as a sequence labeling problem [41]. Suppose the input sentence in IOB2 format is

```
`` `` O O
On IN O O
the DT B-NP B-NP
comptroller NN I-NP I-NP
side NN I-NP I-NP
, , O O
you PRP B-NP B-NP
```

```
're VBP O O

developing VBG O O

and CC O O

making VBG O O

work NN O B-NP

financial JJ B-NP B-NP

controls NNS I-NP I-NP

governing VBG O O

a DT B-NP B-NP

$ $ I-NP I-NP

6 CD I-NP I-NP

billion CD I-NP I-NP

budget NN I-NP I-NP

. . O O
```

In a humanly readable form, this sentence is actually:

```
On [the comptroller side], [you]'re

developing and making work [financial

controls] governing [a $6 billion budget].
```

The words between "[" and "]" indicate a noun-phrase that we need to extract. A useful sub-task for this noun-phrase chunking system is to give each word a part-of-speech (POS) tag by a POS tagger

```
 IN  DT NN NN ,  PRP  VBP
 VBG  CC  VBG NN  JJ
 NNS VBG DT -NP  CD  CD  NN  .
```

With the help of the POS tags, the next stage of the noun-phrase chunking system can extract noun-phrases more easily than without the POS information.

A rather subtle problem is that the POS tagger is not trained to optimize the performance of the noun-phrase chunker in the next stage. In fact, for the above sentence, even when all POS tags are given correctly to the chunker, the chunker would make a mistake on the word "work". The output of the chunker is

```
On [the comptroller side], [you]'re
developing and making [work] [financial
controls] governing [a $6 billion budget].
```

In other words, training a POS tagger to correctly give a POS tag to the word "work" actually does not pay off in this case because it cannot improve the overall system performance. Worse still, this may imply sacrificing accuracy of POS tags of other words.

When this happens, there is a double loss: (1) we fail to extract the noun-phrase correctly because of the chunker performance, and (2) our chunker would have extracted the correct noun-phrase with regard to other words with correct POS tags, but failed to do so because our POS tagger sacrifices this part of accuracy for accuracy in some words that the chunker cannot benefit from. In this way, cascaded errors occur unnecessarily.

1.2 Improvements: where do they come from?

To improve the performance of named entity extraction, we consider the following ideas.

1.2.1 Margin-based improvement

Traditional modeling makes use of probability for parameter estimation. It aims at maximizing the probability over a set of training data. In margin-based training, parameters are trained such that they can maximize the "margin" between the decision boundary and the nearest training data points. In other words, parameters given by a margin-based method are more robust in the sense that it helps improve the performance on the set of data points that are very close to the decision boundary.

We make use of this concept of "margin" in our Maximum Entropy Margin Based (MEMB) model.

1.2.2 Joint-training-based improvement

When each model in a system is separated in a cascaded manner, the models are trained independent of each other. It is hard to predict the performance of the whole system since in the training stage, the goal is not to optimize the performance of the whole system but each of the sub-task instead.

Another major disadvantage is that a certain amount of effort may be wasted in training. A model in an early stage of a system may be spending effort in training to optimize its performance on a subset of training instances. However the model in the next stage of the system may not be able to take advantage of the correct results on this subset from the earlier stages due to imperfect modeling or lack of resources. In other words, some of the effort may be wasted. Worse still, it may harm the performance of another subset that can be handled well in the next stage. In this way, the performance of the whole system is hurt.

We make use of this concept of joint training in our Isomeric Conditional Random Field (ICRF) model.

1.2.3 Reranking-based improvement

Another compromise approach is to rank the possible outputs [42] (e.g. give each possible output a score) from an earlier stage and pass the outputs to the next stage. At the next stage all possible outputs are evaluated and assigned scores. The best output with the highest total score is the final output.

We make use of this concept of reranking in our Online Cascaded Reranking (OCR) model.

1.3 Research Contributions

In this thesis, we formulate the entity extraction problem as a supervised learning task, and propose a cascaded framework for extracting named entities from unstructured text [4]. We develop three models for this framework. One is a "maximum-entropy margin-based" (MEMB) model that introduces the concept of "margin" into a maximum entropy model. The second is an isomeric conditional random field (ICRF) model that links two conditional random fields via a marginal distribution. The last model is an online cascaded reranking (OCR) model that uses online training and reranking to yield the best answer out of a list of candidates.

Essentially, the cascaded framework addresses two main issues in sequence labeling problems:

Long Training Time: Traditional approaches that depend on maximum likelihood training method are slow even with large-scale optimization methods such as L-BFGS. This problem worsens with the sheer volume and growth rate of literature. In this thesis, we propose to use a cascaded framework that greatly reduces training time.

Large Memory Space: The total number of features used to extract named entities from documents is very large. For example, to extract biomedical named entities, we often need to use extra features in addition to those used in general-purpose domains, such as prefix, suffix, punctuations, and more orthographic features. We need a large memory space to store them and that also worsens the first problem. By employing a cascaded framework that divides the named entity recognition task into a segmentation task and a classification task, we can alleviate this problem.

However this framework introduces cascaded errors in practical applications:

Cascaded Errors: While we try to divide a sequence labeling task into cascaded tasks, errors in early tasks may propagate to latter tasks. They amplify the errors in subsequent tasks and result in poor final performance.

In this thesis, we propose three models to solve the above problems:

1.3.1 MEMB Model

Our cascaded framework divides the entity extraction task into a segmentation task and a classification task. We use a unified model, which we call the "maximum-entropy margin-based" (MEMB) model, to solve both tasks. This model considers errors made between a correct and an incorrect output during training. In the segmentation task, we reduce the training cost by grouping named entities of interest into one class. Note that this also helps segmentation of named entity types which are sparse in training data. The same MEMB model is used in the classification task as well.

This approach can be used across domains and requires only an error measure function. The error measure function can be as simple as counting the number of errors made, which usually does not require a domain expert, or only takes a relatively short time for a domain expert to come up with. In particular, we view "error" as a fundamental concept in every domain and, therefore, it can be represented in both the standard representation scheme in a domain and an entity extraction system. Our method does not need to adjust the set of features and the training data does not need to be prepared again. Usually the extra efforts involved in calculating the error measure can be executed by the system automatically.

One of the main contributions of this thesis is as follows. By using the proposed cascaded framework, we can reduce the training time by an order of magnitude - from $O(N_a N_c^2)$ to $O(N_a N_c)$. A unified model, which we term as the "maximum-entropy margin-based" model, is used to boost the extraction performance in this framework. We demonstrate this improvement in named entity extraction tasks using corpuses from shared tasks. In fact, our approach has reached state-of-the-art performance and has outperformed all other systems in the JNLPBA shared task. Details of experimental results and comparisons are also given.

1.3.2 ICRF Model

The second model that we have developed in this thesis is called the isomeric CRF (ICRF) model. Recently, probabilistic graphical models [25] [35] [38] [46] for sequence data have become the predominant formalism for information extraction problems. They deal well with uncertainty and achieve state-of-the-art performance.

Following the current research trend, we put the cascaded framework in a probabilistic context and investigate the possible causes for cascaded errors [3]. We explore a way of joint training, using probabilistic graphical models, without adding huge computation costs. Instead of using a margin in training our model, we develop a new training algorithm for this cascaded framework that considers the relationship of probability distributions in the two tasks. Our proposed approach achieves performance comparable to state-of-the-art systems in real world problems.

To improve efficiency, we also investigate an efficient method for training our model. This method uses a parallel implementation of parameter estimation. The idea is to analyze and divide a parameter estimation problem for the whole model into independent parameter estimation problems for each training instance. Our experiments showed that this could reduce running time by 10x or more.

1.3.3 OCR Model

This online cascaded reranking (OCR) model actually makes use of the concept of margin for parameter estimation and inference. It also uses a reranking method to correct errors in the first task by examining a candidate's score in the second task.

In this model, we use a linear combination of features and use the resulting decision boundary to classify each candidate into a suitable class. We can completely ignore the probability framework but the inference method is exactly the same as in previous models.

Because it does not involve any probability modeling and there is no need to use any gradient descend methods, we can use an online training method [5, 2] for parameter estimation, which is very fast. The advantage of using this online training method is that it can avoid huge computation costs involved in other margin-based methods such as those used in Support Vector Machines (SVM) for structured classification. This online training method improves a parameter value whenever a new data point is available. The updating formula is simple and fast. However due to its algorithmic nature, parallel implementation is not possible.

In other words, the margin-based training help reduce errors in each of the tasks. Furthermore, we generate a list of N-best candidates in the first task, which contains the top-N candidates having the highest score, in descending order. We examine each candidate's score in the second task. The candidate that has the highest total score in the two tasks is chosen as the best output.

Chapter 2

Background and Related Work

This chapter presents some of the related work and background which are necessary for understanding the modeling and algorithms in later sections.

2.1 Background

A general CRF model can be described by

$$\mathrm{P}_\theta(\mathbf{y}|\mathbf{x}) = \frac{\exp[\theta \cdot \mathbf{F}(\mathbf{x},\mathbf{y})]}{Z_\theta(\mathbf{x})} \tag{2.1}$$

where $\theta = (\theta_1, \theta_2, \ldots, \theta_m)$ is the training model parameter, $Z_\theta(\mathbf{x}) = \sum_{\mathbf{y}' \in \mathcal{Y}} \exp[\theta \cdot \mathbf{F}(\mathbf{x}, \mathbf{y}')]$ is the normalization factor.

In fact, this formulation implies that the joint probability distribution $\mathrm{P}_\theta(\mathbf{x},\mathbf{y})$ is proportional to $\exp[\theta \cdot \mathbf{F}(\mathbf{x},\mathbf{y})]$. Thus we can write

$$\mathrm{P}_\theta(\mathbf{x},\mathbf{y}) = \kappa \exp[\theta \cdot \mathbf{F}(\mathbf{x},\mathbf{y})] \tag{2.2}$$

where κ is just a scale factor. κ may also be absorbed into θ.

$$\mathrm{P}_\theta(\mathbf{x}, \mathbf{y}) = \exp[\theta' \cdot \mathbf{F}'(\mathbf{x}, \mathbf{y})] \tag{2.3}$$

where $\theta' =< \theta, \log\kappa >$ and $F' =< F, 1 >$. Bearing this in mind can help understand the isomeric CRF approach in later chapters.

In a typical text mining task, features to be used in a system are usually determined manually. For example, in a noun-phrase chunking task, we observe a sentence $\mathbf{x}$ where each word x_i has a label $y_i \in \{\text{B, I, O}\}$ to indicate a word is the "beginning", "inside", or "out" of a noun-phrase respectively. Thus a feature may be defined as

$$f(\mathbf{x}, \mathbf{y}, i) = \begin{cases} 1 & \text{if } x_i \text{ is the word "boy",} \\ & \text{and } y_i \text{ is the label "B"} \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

A shorthand notation for the above feature is $(x_i = \text{boy}, y_i = \text{B})$. Here, feature value is 1 if a feature is present, and 0 otherwise. Since manually writing all these features is not feasible, a possible solution is to this problem is to define a set of feature templates and let the computer generate the features automatically. Two feature templates are shown below as examples.

$$A1: \quad (x_i, y_i) \tag{2.5}$$

$$A2: \quad (x_{i-1}, x_i, y_i) \tag{2.6}$$

When these two feature templates work on the following labeled sentence,

```
x : I would like to thank my advisor .
y : B O     O    O  O     B  I       O
```

features generated by each template are:

$$A1: \quad (x_i = \text{I}, \quad y_i = \text{B}) \tag{2.7}$$

$$(x_i = \text{would}, \quad y_i = \text{O}) \tag{2.8}$$

$$(x_i = \text{like}, \quad y_i = \text{O}) \tag{2.9}$$

$$\ldots$$

$$A2: \quad (x_{i-1} = \text{START}, \quad x_i = \text{I}, \quad y_i = \text{B}) \tag{2.10}$$

$$(x_{i-1} = \text{I}, \quad x_i = \text{would}, \quad y_i = \text{O}) \tag{2.11}$$

$$(x_{i-1} = \text{would}, \quad x_i = \text{like}, \quad y_i = \text{O}) \tag{2.12}$$

$$\ldots$$

where "START" denotes the start of a sentence. This reduces the problem of exactly defining a set of features to that of defining a set of feature templates. In some cases, users may want to define a threshold to remove some features that occur rarely, in order to reduce the number of features. An alternative to using a threshold is using a χ^2 test on the features as in [39]. This feature template technique is used in programs such as [24], [36], and [32].

2.1.1 Maximum Likelihood Principle

In early works of CRFs, model parameter $\mathbf{w}$ is usually learned by the maximum likelihood (ML) principle [27, 41], which yields a conditional probability distribution for the data being considered. The best decision for the current task is then made with respect to this conditional probability distribution.

To estimate model parameter $\mathbf{w}$, we maximize log-likelihood $\mathcal{L}$ over the set of training data $\mathcal{T} = \{\mathbf{x_i}, \mathbf{y_i}\}$:

$$\begin{aligned}
\mathrm{P_w}(\mathcal{T}) &= \mathrm{P_w}(\mathcal{Y}|\mathcal{X})\mathrm{P}_\theta(\mathcal{X}) && (2.13)\\
&= \prod_i \mathrm{P_w}(\mathbf{y_i}|\mathbf{x_i})\mathrm{P}_\theta(\mathbf{x_i}) && (2.14)\\
\mathcal{L} &= \log \mathrm{P_w}(\mathcal{T}) && (2.15)\\
&= \sum_i \log \mathrm{P_w}(\mathbf{y_i}|\mathbf{x_i}) && (2.16)\\
&\quad + \sum_i \log \mathrm{P}_\theta(\mathbf{x_i})\\
\arg\max_{\mathbf{w}} \mathcal{L} &= \arg\max_{\mathbf{w}} \sum_i \log \mathrm{P_w}(\mathbf{y_i}|\mathbf{x_i}) && (2.17)\\
&= \arg\max_{\mathbf{w}} \sum_i \mathbf{w} \cdot \mathbf{F}(\mathbf{x_i}, \mathbf{y_i}) && (2.18)\\
&\quad - \sum_i \log Z_{\mathbf{w}}(\mathbf{x_i})
\end{aligned}$$

where θ is the parameter for $\mathrm{P}_\theta(\mathcal{X})$, $\mathcal{X} = \{\mathbf{x_i}\}$, and $\mathcal{Y} = \{\mathbf{y_i}\}$. $\mathrm{P}_\theta(\mathcal{X})$ is dropped in the maximization because it does not involve $\mathbf{w}$.

C

In practice, we need to add regularization terms to the log-likelihood to avoid overfitting. Assume a Gaussian prior [45] with mean 0 and covariance $\sigma^2 I$,

$$\begin{aligned}\arg\max_{\mathbf{w}} \mathcal{L} &= \arg\max_{\mathbf{w}} -\frac{\|\mathbf{w}\|^2}{2\sigma^2} \\ &\quad + \sum_i [\mathbf{w} \cdot \mathbf{F}(\mathbf{x_i}, \mathbf{y_i}) - \log Z(\mathbf{x_i})]\end{aligned} \tag{2.19}$$

After a CRF is trained, it can be used for finding the best output sequence $\hat{\mathbf{y}}$ for an input sequence $\mathbf{x}$.

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}'} \mathrm{P}_{\mathbf{w}}(\mathbf{y}'|\mathbf{x}) \tag{2.20}$$

$$= \arg\max_{\mathbf{y}'} \mathbf{w} \cdot \mathbf{F}(\mathbf{x}, \mathbf{y}') \tag{2.21}$$

2.1.2 Max-margin Principle

Another training approach is to estimate model parameters from the mechanism used in the decision phase. Parameters are estimated by the max-margin (MM) principle [47, 33], which chooses parameters that give the largest "margin" to minimize empirical risks on the training data.

Formally speaking, for each training instance $(\mathbf{x}, \mathbf{y})$ and each possible predicted output sequence $\mathbf{y}' \neq \mathbf{y}$ for $\mathbf{x}$, we require that

$$\mathbf{w} \cdot \mathbf{F}(\mathbf{x}, \mathbf{y}) - \mathbf{w} \cdot \mathbf{F}(\mathbf{x}, \mathbf{y}') \geq L \tag{2.22}$$

where L is a margin that we specify to reflect the level of confidence of making a correct decision in this model. Note that in this paper a "margin" means the functional margin [11], not the geometric margin. In training, we try to find $\mathbf{w}$ by

$$\min \frac{\|\mathbf{w}\|}{2} \tag{2.23}$$

such that $\forall(\mathbf{x}, \mathbf{y})$ and $\forall \mathbf{y}' \neq \mathbf{y}$

$$\mathbf{w} \cdot \mathbf{F}(\mathbf{x}, \mathbf{y}) - \mathbf{w} \cdot \mathbf{F}(\mathbf{x}, \mathbf{y}') \geq L \tag{2.24}$$

Another way of looking at this problem is to find $\mathbf{w}$ by minimizing the regularized empirical risk function:

$$\begin{aligned} \arg\min_{\mathbf{w}} \frac{\|\mathbf{w}\|^2}{2} - \sum_i \{ \mathbf{w} \cdot \mathbf{F}(\mathbf{x}_i, \mathbf{y}_i) \\ - \max_{\mathbf{y}' \neq \mathbf{y}_i} [\mathbf{w} \cdot \mathbf{F}(\mathbf{x}_i, \mathbf{y}') + L] \} \end{aligned} \tag{2.25}$$

2.1.3 Discriminative Models

Let us assume that we can generate, store, and add a new feature into our feature space $\mathcal{Z}$, and also can efficiently retrieve all activated features in $\mathcal{Z}$ for an instance. Now our learning model must be able to handle well the enormous number of features and the possible correlations between them. We choose to use a log-linear model with discriminative training to address the difficulties. We aim to learn the conditional distribution:

$$\Pr(\mathbf{y}|\mathbf{x};\mathbf{\Lambda}) = \frac{\exp[\mathbf{\Lambda}\cdot\mathbf{F}(\mathbf{x},\mathbf{y})]}{Z(\mathbf{x};\mathbf{\Lambda})} \tag{2.26}$$

where $\mathbf{\Lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_m)$ is the training model parameter, $\mathbf{F}(\mathbf{x},\mathbf{y}) = (f_1(\mathbf{x},\mathbf{y}), \ldots, f_m(\mathbf{x},\mathbf{y}))$ is the feature vector for instance $(\mathbf{x},\mathbf{y})$, and $Z(\mathbf{x};\mathbf{\Lambda}) = \sum_{\mathbf{y}'\in\mathcal{Y}} \exp[\mathbf{\Lambda}\cdot\mathbf{F}(\mathbf{x},\mathbf{y}')]$ is the normalization factor.

Log-linear models are widely used in maximum entropy modeling. Parameters $\mathbf{\Lambda}$ are optimized such that $E_{\mathbf{\Lambda}}[f(\mathbf{x},\mathbf{y})] = \hat{E}[f(\mathbf{x},\mathbf{y})]$, where $E_{\mathbf{\Lambda}}[\cdot]$ is the expectation parametrized by $\mathbf{\Lambda}$ and $\hat{E}[\cdot]$ is the empirical expectation. The log-linear models can be shown to have maximum entropy over the training instances.

In discriminative training, a conditional distribution $P_{\mathbf{w}}(\mathbf{y}|\mathbf{x})$ is modeled directly. This allows more freedom to the model to adjust its parameter $\mathbf{w}$ and fit training instances $\{(\mathbf{x},\mathbf{y})\}$. By comparing models

$$P_g(\mathbf{x},\mathbf{y};\mathbf{w}) = P_g(\mathbf{y};\mathbf{w})P_g(\mathbf{x}|\mathbf{y};\mathbf{w}) \tag{2.27}$$

$$P_d(\mathbf{x},\mathbf{y};\mathbf{w}) = P_d(\mathbf{x};\mathbf{w}')P_d(\mathbf{y}|\mathbf{x};\mathbf{w}) \tag{2.28}$$

where (2.27) is used in generative training and (2.28) is used in discriminative training, it is easy to see that there are fewer constraints for discriminative training as parameter $\mathbf{w}$ does not need to account for marginal distribution $P_d(\mathbf{x};\mathbf{w}')$, which can be parametrized by another parameter $\mathbf{w}'$.

Another advantage of modeling the conditional distribution directly is the flexibility of adding a large number of possibly correlated and diversified features without degrading the performance as in generative training. For example, conditional random fields (CRFs) have been shown to perform better than hidden Markov models (HMMs) , its generative counterparts.

2.2 Related Work

2.2.1 Feature Selection Methods

The traditional feature selection problem is closely related to our feature abstraction problem. Two main types of feature selection methods are widely used: wrapper methods and filter methods. In wrapper methods, the learning algorithm is used to estimate the performance of a subset of features [22, 1]. This approach can usually generate a small number of features that are suitable for the learning algorithm being used. However, it is slow because the learning algorithm must be used iteratively. In filter methods, the features are selected independent of the learning algorithm being used. They use statistical tests or mutual information to "filter out" redundant features [26, 23]. However, filter methods usually work on discrete classes only.

Feature induction [31] with conditional random fields [27] has been used effectively for text mining. It makes use of a pseudo-likelihood to estimate the impact on performance of adding a new feature. This estimation is done iteratively to select a new feature to be added from a set of proposed new features. The model is then trained by applying a few BFGS iterations before the next iteration.

2.2.2 Parameter Estimation Methods

CRFs [27], since being introduced to solve sequence labeling problems, have shown good performance in learning tasks such as noun-phrase chunking and named entity recognition [41, 31, 49, 40]. CRFs can be extended to semi-CRFs [38], where an element in a sequence (e.g. a word in a sentence) is generalized to a segment in a sequence (e.g. a named entity in a sentence).

Most of the early work followed the maximum likelihood principle in training and used the L-BFGS method, a limited-memory quasi-Newton method for conducting large-scale optimization, in parameter estimation because of its fast convergence rate.

There is another training approach that is based on the concept of "margin". For example, a perceptron-like algorithm is used in training a hidden Markov model (HMM) [8]. For structured classification, [47] presented a margin-based general framework known as the maximum margin Markov network. The parameter estimation problem is solved with an approach similar to SVM, where a dual problem is formulated and optimization methods analogous to sequential minimal optimization (SMO) for SVMs are used. A formal SVM treatment for structured classification is also presented in [50].

A margin-based approach to parsing is presented in [48]. It uses a common approach used in SVMs for solving the dual problem in training. There is another line of research that focuses on online training methods. For example, [33] proposed an online margin-based training method for parsing. This type of training method is fast and has the advantage that it does not need to form the dual problem, though the performance may be inferior. Detailed descriptions of these algorithms, e.g. Margin Infused Relaxed Algorithm (MIRA), and online passive-aggressive algorithms, can be found in [10, 9].

In this thesis we treat the purely margin-based approach as a method for estimating parameters. Actually, we can view the parameters estimated by the margin-based principle as "the parameters that can give a distribution with margin-based characteristics over the training set." As mentioned in [41], the best objective value (i.e. the log-likelihood) in maximum likelihood training does not necessarily give the best learning results. Therefore, we can treat the margin-based approach as an alternative method for training.

2.2.3 Cascaded Methods

To handle multiple tasks simultaneously, Dynamic Conditional Random Fields (DCRFs) [46] use factorized probabilistic models for segmentation and labeling. The model is formulated such that joint training is performed to optimize the overall system performance. However since exact inference is intractable in a DCRF, some approximate inference algorithms have to be used [13] [53]. Our isomeric CRF approach tries to achieve the same joint training without using an approximate inference algorithm. The speed is faster when existing well-developed algorithms for a linear-chain CRF are reused. Because we formulate each sub-task as a linear chain CRF, it is more flexible to give each sub-task its own feature template to best solve the problem.

Another approach is to employ a joint decoding method [42]. In a dual-layer CRFs approach, an N-best list is generated to avoid enumeration of all possible segmentations. This re-ranking technique has been used successfully in many NLP applications, such as speech recognition [43] and NP-bracketing [20]. This approach is more suitable for problems where two separately trained CRFs are to be used together in a cascaded way to solve a single task.

A more complete and interesting study is described in [44], where comparisons between *cascaded training and testing*, *joint training and testing*, and *joint testing with cascaded training* were presented and their performances were evaluated on a dataset of email seminar announcements. The objective of this study was to transfer regularities learned from a well-studied subtask to a new related task. Therefore their focus was on how to perform joint-decoding of separately-trained models.

For joint inference on database records and semi-structured sources, readers can refer to [37], which proposes a joint approach to information extraction, where segmentation of all records and entity resolution are performed together in a single integrated inference process.

Chapter 3

Problem Description

This chapter presents the text mining problem in a machine learning context. We start by describing the general supervised learning setting and then discuss some examples and applications. Then we move on to discuss how a text mining problem is cast as a supervised learning problem.

In many text mining tasks, a set of training data is given and the problem can be cast as a supervised learning problem. We follow [51] to describe the general setting of our learning problem as follows.

Definition 1 *The goal of the general learning problem is to minimize the* *following risk function* $R(w)$.

$$R(w) = \int Q(f(x,y);w)\ \mathrm{d}P(f(x,y)) \qquad (3.1)$$

where

$$Q(f(x,y);w) \text{ is a loss function on } f(x,y) \qquad (3.2)$$

$$f(x,y) \text{ describes the pair } (x,y) \qquad (3.3)$$

$$x \text{ is an observation} \qquad (3.4)$$

$$y \text{ is the answer for } x \qquad (3.5)$$

$$w \text{ is the parameter for } Q(\cdot) \qquad (3.6)$$

$$P(f(x,y)) \text{ is the unknown probability distribution of } f(x,y) \qquad (3.7)$$

and $f(x_1,y_1),\ldots,f(x_n,y_n)$ *from* $P(f(x,y)$ *are given.*

From the above formal definition, we can see that the machine learning problem handles the problem of finding the parameter w that can minimize risk $R(w)$, i.e. to minimize the number of prediction mistakes on $f(x,y)$ from distribution $P(f(x,y))$, under the condition that $P(f(x,y))$ is unknown but a set of samples $\{f(x_i,y_i)\}$ is given. Usually w parametrizes and enables a model to predict the answer for x. Thus loss function $Q(\cdot)$ can be calculated by comparing the predicted answer with the true answer y. That is why $Q(\cdot)$ is also parameterized by w. For example, in regression problem, if we are using a quadratic equation $ax^2 + bx + c$, the loss function is

$$Q((x, y); (a, b, c)) = (y - (ax^2 + bx + c))^2 \tag{3.8}$$

where we use L_2 norm as our loss metrics and represent the pair (x, y) directly by an identical feature function.

This general setting of machine learning problem assumes that the feature function $f(x, y)$, which is used to describe the pair (x, y), is available or given. Technically speaking, even without a proper feature function $f(x, y)$ to describe the pair (x, y), a machine learning problem can still be solved.

However, this is not true in many real world applications. Often there are uncertainties involved in choosing $f(x, y)$. For example, should we represent numerical values in log-scale? Should the values be normalized? If yes, what are the upper and lower bounds? Should we represent discrete values by binary variables? Researchers need to resort to trials and errors to make judgments. Results may be different from time to time for different datasets. This is one of the reasons why many machine learning systems does not give satisfactory performance on real-world data.

Strictly speaking, if the definition of $f(x, y)$ is changed, for example by using a slightly different set of features, then it becomes a different machine learning problem. The old and new problem may or may not have useful relationships between their parameters, performance, or convergence rate. This

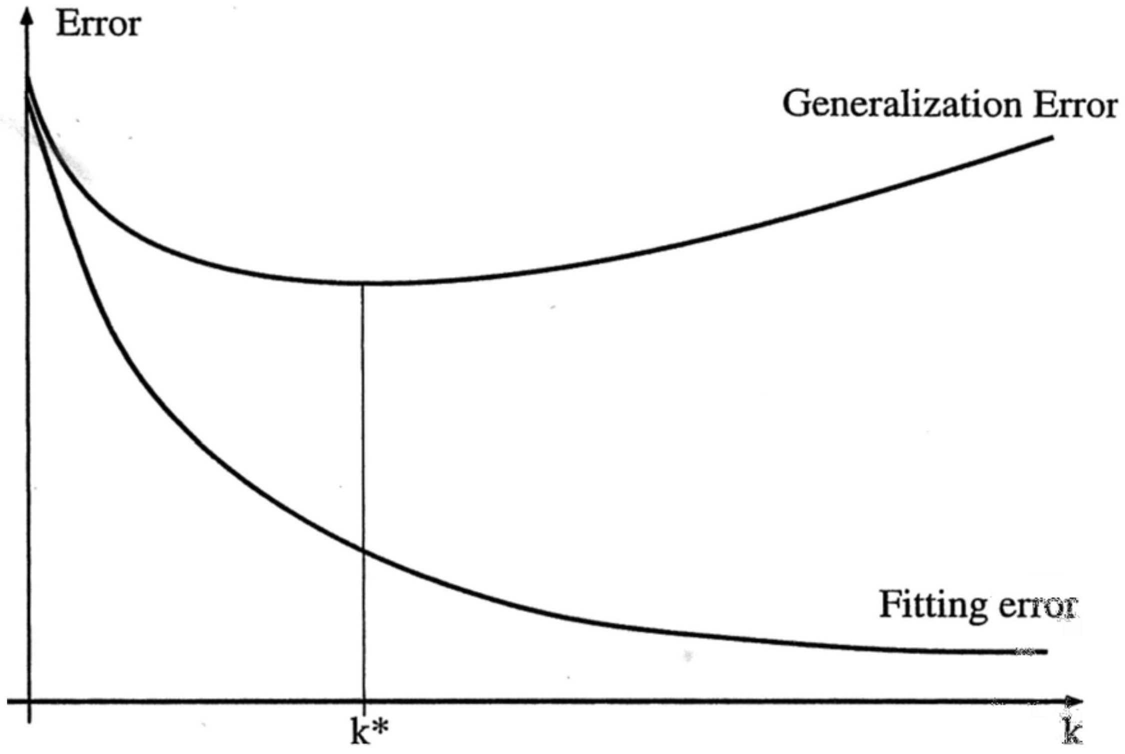

Figure 3.1: Generalization error versus fitting error

makes comparisons among different mining systems difficult because they may differ in machine learning algorithms and also in terms of features used. This scenario is further complicated if there are more training parameters used in the systems.

In general, the feature selection problem affects the learning performance in two ways.

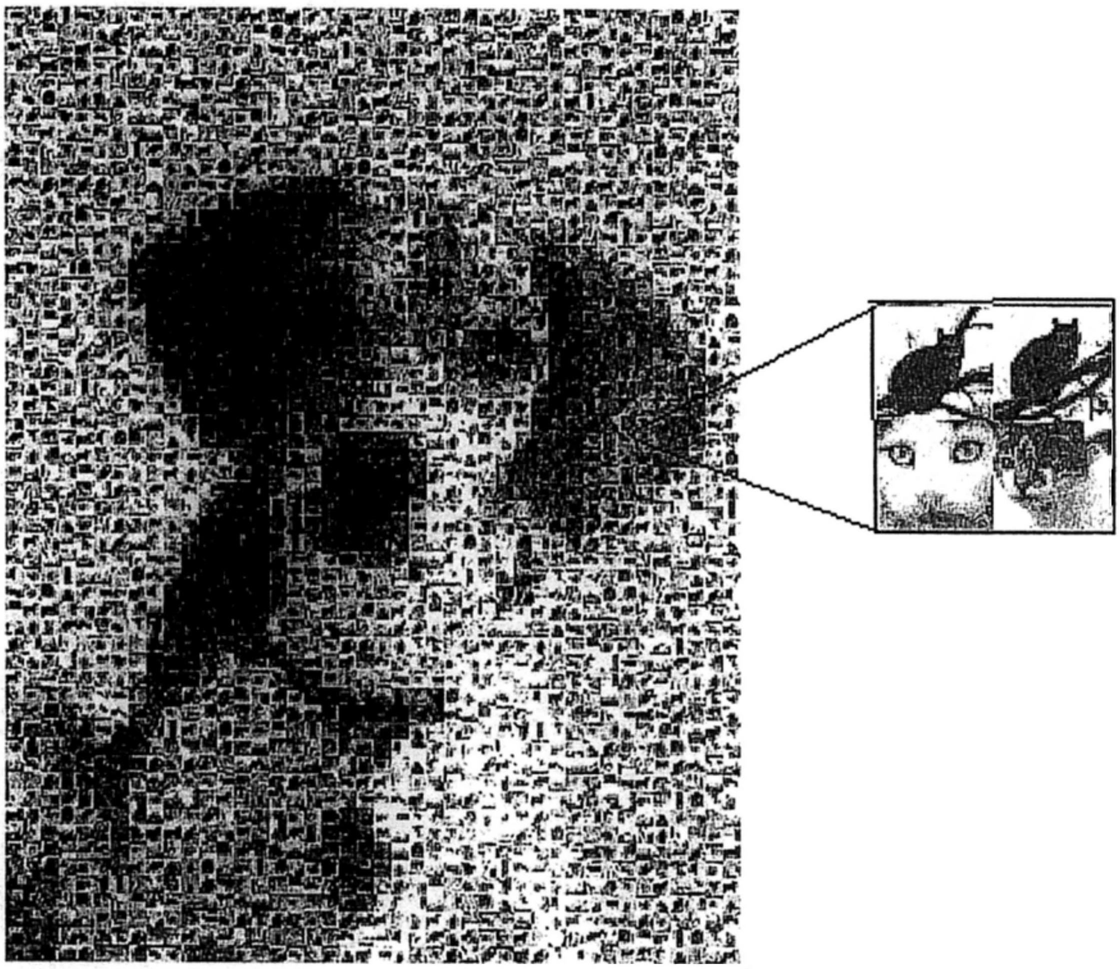

Figure 3.2: A dog or many cats?

3.1 Linear Combination of Features

Linear combinations of features are widely used in many learning algorithms. For example, in support vector machines (SVMs) and maximum entropy

(MaxEnt) model, and their variants.

$$\text{SVM}: \quad \min \frac{1}{2}||\mathbf{w}||^2 \tag{3.9}$$

$$\text{subject to} \quad c_i(\mathbf{w} \cdot \mathbf{f}(\mathbf{x_i}) - b) \geq 1, \tag{3.10}$$

$$1 \leq i \leq n \tag{3.11}$$

$$\text{MaxEnt}: \quad P(\mathbf{x}|\mathbf{w}) = \frac{1}{Z(\mathbf{w})} e^{-\mathbf{w}\cdot\mathbf{f}(\mathbf{x})} h(\mathbf{x}) \tag{3.12}$$

In these approaches, part of the models involve a linear combination of features, as follows.

$$w_1 f_1 + w_2 f_2 + \cdots + w_n f_n \tag{3.13}$$

where f_i represents the presence of feature i.

In the following, we give an example to show why this linear combination of features is inadequate for text mining problems. Suppose we are using only word features and need to distinguish that "Carnegie Mellon" is a person name, and "Carnegie Mellon University" is an organization name. We use normalized weights to represent the probability of a word to be a person name, i.e. $0 \leq w_i \leq 1$. Suppose further that from the training set, we can see that f_1 = "Carnegie" and f_2 = "Mellon" appear more often as a person name and their weights are : $w_1 = 0.8$ and $w_2 = 0.7$. f_3 = "University" can be a place name, or an organization name but can never be a person name:

so its weight $w_3 = 0$. The score for these two named entities are

$$\text{Score(Carnegie Mellon)} \tag{3.14}$$

$$= w_1 f_1 + w_2 f_2 = 0.7 + 0.8 \tag{3.15}$$

$$= 1.5 \tag{3.16}$$

$$\text{Score(Carnegie Mellon University)} \tag{3.17}$$

$$= w_1 f_1 + w_2 f_2 + w_3 f_3 = 0.7 + 0.8 + 0 \tag{3.18}$$

$$= 1.5 \tag{3.19}$$

They both get the same score and are regarded as person names (or NOT person names, depending on the decision criteria or threshold). This problem emanates from the fact that the sum of individual features does not necessarily reflect properties of conjunction of the features. A more extreme example would be the word "not" that can negate the meaning of a phrase.

One remedy to this problem is to use conjunction of features $f_{i,j}$ to represent that features i and j appear together. In the above example, the features become $f_{1,2}$ = (Carnegie, Mellon) with $w_{1,2} = 0.9$ and $f_{1,2,3}$ = (Carnegie, Mellon, University) with $w_{1,2,3} = 0$. Given the above features and a proper representation of the named entities by these features, i.e. use only $f_{1,2}$ for "Carnegie Mellon" and only $f_{1,2,3}$ for "Carnegie Mellon University", the scores become

$$\text{Score(Carnegie Mellon)} \tag{3.20}$$

$$= w_{1,2} f_{1,2} \tag{3.21}$$

$$= 0.9 \tag{3.22}$$

$$\text{Score(Carnegie Mellon University)} \tag{3.23}$$

$$= w_{1,2,3} f_{1,2,3} \tag{3.24}$$

$$= 0 \tag{3.25}$$

3.2 Model Selection

The model selection problem in machine learning tries to solve the problem of over-fitting. The linear combination of features also contributes to this problem. More details of the model selection problem can be found in [51, 54].

Suppose a family of models $\mathcal{M}$ can be specified by its structure with a scale parameter k. For example, a model can be described as "k-th order polynomial", i.e. $a_k x^k + \cdots + a_1 x + a_0$. The scale parameter k gives an order of complexity of different models under the same structure. This complexity can be roughly regarded as a measure of the number of possible mappings that a model can give. A model of larger scale k includes models of smaller scale k', denoted by $M_{k'} \prec M_k$. As shown in Figure 3.1 (adapted from [54]), when k is increased, the fitting error on the training set decreases while the generalization error decreases to a minimum point and then increases again. Our goal is to select the model with the best k^* so as to minimize the generalization error but what we can observe is the fitting error on the training set. There are many approaches to choose k^*, including regularization, cross-validation, minimum description length (MDL) principle and Akaike's information criterion (AIC) criterion, etc.. One notable approach is the structural risk minimization (SRM) strategy that makes use of the concept of VC-dimension [51].

It is tempting to think that the number of parameters in a model can represent its complexity. For example, a linear model has a lower complexity than a higher order polynomial. However, with the fact that a simple sinusoidal $y = A \sin Tx$ with only two parameters A and T can fit any set of data points, the number of parameters in a model clearly is not the best indicator of the model's complexity.

This problem is further complicated in text mining tasks because of the rich feature sets that are available for text. These features may or may not be fully derived from the text document in the given data. For example, part-of-speech (POS) information, prefix, suffix, lemma form, base type, and domain lexicon. Even if we fix our model to be a linear model, when we add more features to our model, we may not be sure how much complexity we

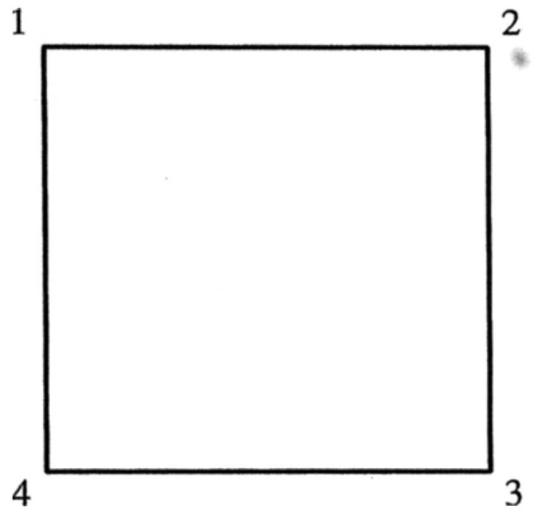

Figure 3.3: A square with four lines

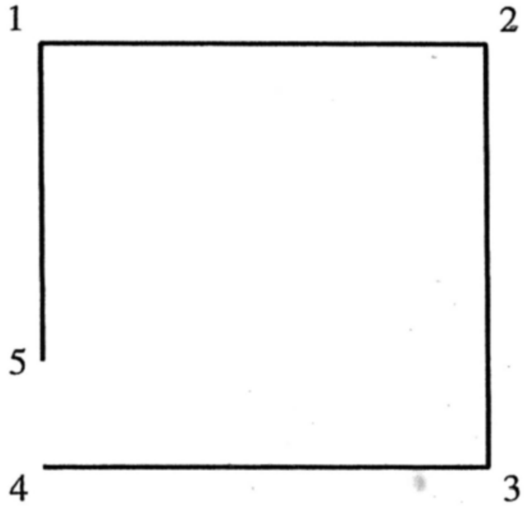

Figure 3.4: Four lines but not a square

are "adding" to the model. As shown in Figure 3.1, the over-fitting problem may easily occur if we add too much complexity but we also face the risk of under-fitting if we do not add enough complexity. Because of the nature of text (as compared to other learning problems that involve numerical values only) and that some features may come from sources (e.g. a domain lexicon) out of the scope of the given text documents, it is harder to evaluate the complexity of the model, i.e. choosing k^* becomes a harder problem.

3.3 Our Model Requirements

In this section, we outline the requirements for our learning model:

Large number of features. Because there can be an enormously number of conjunctions of features generated, the learning model we use must be able to handle the large number of features, say, at least millions of features.

Correlated features. Since a basic feature may appear many times in other conjunctions of features, there are correlations between these conjunctions of features. Our learning model needs to handle the correlations well and not to let these correlations degrade the performance.

Parameter estimation for conjunction of features. Each conjunction of features is expected to occur less frequently than its constituent basic features. According to the law of large numbers, in general reliability of conjunctions of features is less than that of basic features. The parameter estimation process is required to address this difference.

Chapter 4

Modeling

4.1 Maximum-entropy Margin-based (MEMB) Model

4.1.1 Proposed Framework

We propose a cascaded framework for extracting named entities in scientific text. The framework consists of two modules: a segmentation module (SEG) and a classification module (CLASS). Each module uses the same general model that considers the probability and the margin for the data being considered, which we term the "maximum-entropy margin-based" (MEMB) model.

In the SEG module, we segment the text and identify entity candidates

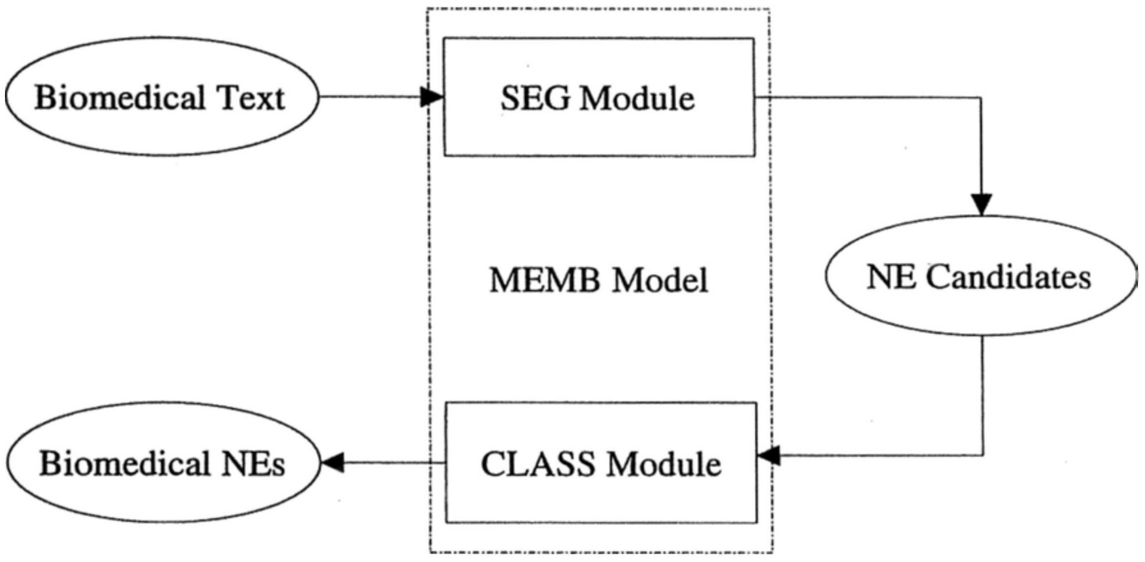

Figure 4.1: A schematic representation of our cascaded framework for extracting biomedical named entities (NEs). Both the SEG and CLASS modules use the MEMB model.

without identifying their types. The SEG module is performed by grouping all entity types of interest into one super-type . In the CLASS module, each entity candidate is classified into one of the entity types. One advantage of using this cascaded approach is that segmenting the text and identifying entity candidates are relatively simpler problems, as different types of named entities share common features in the context of scientific text. Another advantage is that we can select features that are suitable for each module. While combining the two modules into a single phase is possible, the large number of features involved may sometimes degrade the performance. This phenomenon has been observed in many tasks. In particular, in the context of named entities, [39] reported that a conditional random field (CRF) using only orthographic features gave better performance than using a complete set of features that included semantic features. The third advantage is that with the reduced number of features due to both the selected features for each module and the reduced number of output labels, the computational cost is substantially lower and thus the training time for our approach is significantly shorter than when combining the two sets of features.

In the following, we first describe our notations and present the general MEMB model in our framework. Then we present the details of its application to extracting named entities in scientific text, which includes the training algorithm and the decision algorithm with analysis in later chapters.

4.1.2 Notations

In the SEG module, each sentence in text is represented by a sequence $\mathbf{x}$. The named entities in $\mathbf{x}$ can be segmented with the help of a segmentation sequence $\mathbf{y}$. We assume there is only one correct segmentation $\mathbf{y}$ for each sentence $\mathbf{x}$. The SEG module is required to give the correct output sequence $\mathbf{y}$ upon an input sequence $\mathbf{x}$.

There is a feature vector $\mathbf{F_s} = (F_1, F_2, \ldots, F_m)$, where m is the number of features, that associates each sentence $\mathbf{x}$ and each segmentation sequence $\mathbf{y}$, i.e. $\mathbf{F_s} = \mathbf{F_s}(\mathbf{x}, \mathbf{y})$. This is commonly known as joint feature representation or combined feature representation of inputs and outputs in the support vector machine (SVM) literature [50]. We denote the inner product of two vectors $\mathbf{a}$ and $\mathbf{b}$ as $\mathbf{a} \cdot \mathbf{b}$.

In segmenting named entities in text, $\mathbf{x}$ is a sequence of words that forms a sentence, while $\mathbf{y}$ is represented as a sequence of labels. Each label is an element in a label set Ω. We use the IOB2 notation to identify a segment in a sentence, i.e. $\Omega = \{\texttt{I}, \texttt{O}, \texttt{B}\}$, where `B` represents a word that is the start of a named entity, `I` represents a word that is inside a named entity (but not the start), and `O` represents a word that is not part of a named entity. For example, an RNA named entity is in the following sentence:

```
MTIIa mRNA levels increased significantly .
B     I    O      O         O             O
```

We combine the words with the labels `B` and `I` together and identify "`MTIIa mRNA`" as a segment of a named entity. Note that in this IOB2 notation, a label `O` can be followed only by `O` or `B` but not `I`.

In general, there are far more `O` tags in training data. Therefore if there are no special clues on a word, our model will give a `O` tag to the word.

In the CLASS module, each input sequence $\mathbf{x}$ is an entity candidate obtained from the SEG module. For example, $\mathbf{x}$ = "`EBV genome`". The output

$\mathbf{y}$ is reduced to a single label y that represents the type of the named entity $\mathbf{x}$. The feature vector becomes $\mathbf{F_c} = \mathbf{F_c}(\mathbf{x}, y)$ accordingly.

4.1.3 A General Model for Segmentation and Classification

Our general model is a maximum entropy model with a margin incorporated. For ease of exposition, we restrict our discussion to the SEG module. This analysis can be similarly extended to the CLASS module with each sentence $\mathbf{x}$ and each segmentation sequence $\mathbf{y}$ being replaced by a named entity candidate and an entity type respectively. The general model can be described by

$$\Pr(\mathbf{y'}|\mathbf{x_i}; \mathbf{\Lambda}) = \frac{\exp[\mathbf{\Lambda} \cdot \mathbf{F}(\mathbf{x_i}, \mathbf{y'}) + E(\mathbf{y_i}, \mathbf{y'})]}{Z(\mathbf{x_i}; \mathbf{\Lambda})} \tag{4.1}$$

where $\mathbf{\Lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_m)$ is the training model parameter, $Z(\mathbf{x_i}; \mathbf{\Lambda}) = \sum_{\mathbf{y'} \in \mathcal{Y}} \exp[\mathbf{\Lambda} \cdot \mathbf{F}(\mathbf{x_i}, \mathbf{y'}) + E(\mathbf{y_i}, \mathbf{y'})]$ is the normalization factor, and

$$E(\mathbf{y_i}, \mathbf{y'}) = \begin{cases} Err(\mathbf{y_i}, \mathbf{y'}) & \text{in training} \\ 0 & \text{in decision} \end{cases} \tag{4.2}$$

$Err(\mathbf{y_i}, \mathbf{y'}) \geq 0$ is an error measure function of $\mathbf{y_i}$ and a possible output $\mathbf{y'}$ with equality when $\mathbf{y'} = \mathbf{y_i}$. A traditional conditional random field (CRF) [27] model can be obtained by setting $E(\mathbf{y_i}, \mathbf{y'}) \equiv 0$.

4.1.4 Weighted Error Measure Function

In order to allow the flexibility to reflect the importance of the error measure function, we use a weighted error measure function as follows:

$$Err(\mathbf{y}, \mathbf{y}') = \kappa \sum_{i=1}^{|\mathbf{y}|} I(y_i \neq y_i') \tag{4.3}$$

where κ is the weight for the error measure function. From our empirical results, a weighted error measure function usually gives better results compared to a non-weighted one (i.e. $\kappa = 1$).

4.1.5 Training: Optimization Problem

Training the parameter $\mathbf{\Lambda}$ is intractable in general. Consider a sentence $\mathbf{x}$ of length p with IOB2 notation where the number of possible labels for a word is 3. The total number of possible outputs $\mathbf{y}'$ for sentence $\mathbf{x}$ is 3^p, where p is usually in the range of 10 to 20. Note that if one sentence is long, the total training time will be dominated by that sentence because of the exponential increase in the number of possible $\mathbf{y}'$ for that sentence. Enumerating all possible $\mathbf{y}'$ is computationally expensive and generally impractical. Thus we follow the assumptions that are commonly made in a sequence labeling model and use a modified version of the forward-backward algorithm for the MEMB model. We assume first-order independence on the output $\mathbf{y}$ for the set of features $\mathbf{F}(\mathbf{x}, \mathbf{y})$ and write $\mathbf{F}(\mathbf{x}, \mathbf{y})$ as the sum of features at all positions of sentence $\mathbf{x}$.

$$\mathbf{F}(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{|\mathbf{x}|} \mathbf{f}(\mathbf{x}, y_{i-1}, y_i, i) \tag{4.4}$$

where $|\mathbf{x}|$ is the length of sentence $\mathbf{x}$, and $\mathbf{f} = (f_1, f_2, \ldots, f_m)$. Each f_k is a feature function defined at position i of sentence $\mathbf{x}$. More precisely, it is a binary function of the whole sentence $\mathbf{x}$, the previous label y_{i-1}, the current (i.e. the i-th) label, and the position i. For example, it can be defined as

$$f_k(\mathbf{x}, y_{i-1}, y_i, i) = \begin{cases} 1 & \text{if } x_i \text{ is the word } \texttt{genome}, \\ & \text{and } y_{i-1} \text{ is the label } \texttt{B} \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$

In our experiments, we observed that some features appeared only a few times in the whole training corpus. After testing on smaller training sets, we decided to filter out these features to speed up the training process. We set our cutoff number to be 3. Finally nearly 3 million features were used.

We further assume that the error measure function $Err(\mathbf{y}, \mathbf{y}')$ is decomposable. This measures the error between two segmentation sequences $\mathbf{y}$ and $\mathbf{y}'$ by directly counting the number of mismatched labels.

$$Err(\mathbf{y}, \mathbf{y}') = \sum_{i=1}^{|\mathbf{y}|} I(y_i \neq y'_i) \tag{4.6}$$

where $I(x) = 1$ if x is true, and 0 otherwise.

We formulate the parameter estimation problem as an optimization problem. Assuming each sentence $\mathbf{x_i}$ and its segmentation sequence $\mathbf{y_i}$ is independently drawn from an underlying unknown distribution, we estimate the parameter $\mathbf{\Lambda}$ by maximizing the conditional likelihood over a set of training data $\mathcal{T} = \{(\mathbf{x_i}, \mathbf{y_i})\}$.

$$P^T(\mathcal{T}) = \prod_i P^T_{\mathbf{\Lambda}}(\mathbf{y_i}|\mathbf{x_i}) P_\theta(\mathbf{x_i}) \tag{4.7}$$

where θ is the parameter for the marginal distribution $\mathrm{P}_\theta(\mathbf{x_i})$. (See [45] for a detailed explanation and the advantages of using a different parameter θ to parameterize the marginal distribution of $\mathbf{x_i}$). The probability is typically represented as a log-likelihood $\mathcal{L}(\mathbf{\Lambda})$ for optimization [45]. The convexity of the form is helpful in parameter estimation.

$$\mathcal{L}(\mathbf{\Lambda}) = \log P^T_{\mathbf{\Lambda}}(\mathcal{T}) \tag{4.8}$$

$$= \sum_i \log P^T_{\mathbf{\Lambda}}(\mathbf{y_i}|\mathbf{x_i}) + \sum_i \log P_\theta(\mathbf{x_i}) \tag{4.9}$$

The problem is now transformed to searching for the parameter $\mathbf{\Lambda}$ that can give the maximum log-likelihood over the set of training data.

$$\arg\max_{\mathbf{\Lambda}} \mathcal{L}(\mathbf{\Lambda}) = \arg\max_{\mathbf{\Lambda}} \sum_i \mathbf{\Lambda} \cdot \mathbf{F}(\mathbf{x_i}, \mathbf{y_i}) - \log Z^T_{\mathbf{\Lambda}}(\mathbf{x_i})$$

Note that $\mathrm{P}_\theta(\mathbf{x_i})$ is dropped in the maximization because it does not involve $\mathbf{\Lambda}$. The above formulation is similar to that in a linear-chain CRF except that the normalization factor $Z^T_{\mathbf{\Lambda}}(\mathbf{x_i})$ contains an additional term $Err(\mathbf{y_i}, \mathbf{y}')$ in it, i.e. if $Err(\mathbf{y_i}, \mathbf{y}') \equiv 0$, it is reduced to a linear-chain CRF. We need to modify the CRF training procedure to account for this additional term. To perform maximization, we need to get the gradient of $\mathcal{L}(\mathbf{\Lambda})$.

$$\begin{aligned} & \nabla\mathcal{L}(\mathbf{\Lambda}) \\ = & \sum_i \mathbf{F}(\mathbf{x_i}, \mathbf{y_i}) - \frac{\sum_{\mathbf{y'}\in\mathcal{Y}} \mathbf{F}(\mathbf{x_i}, \mathbf{y'}) e^{\mathbf{\Lambda}\cdot\mathbf{F}(\mathbf{x_i},\mathbf{y'})+Err(\mathbf{y_i},\mathbf{y'})}}{Z^T_{\mathbf{\Lambda}}(\mathbf{x_i})} \\ = & \sum_i \mathbf{F}(\mathbf{x_i}, \mathbf{y_i}) - \mathcal{E}_{P^T_{\mathbf{\Lambda}}(\mathbf{Y}|\mathbf{x_i})}\mathbf{F}(\mathbf{x_i}, \mathbf{Y}) \end{aligned} \tag{4.10}$$

where $\mathbf{Y}$ is a random variable that takes a value in the output space $\mathcal{Y}$, and $\mathcal{E}_{P^T_{\mathbf{\Lambda}}(\mathbf{Y}|\mathbf{x_i})}\mathbf{F}(\mathbf{x_i}, \mathbf{Y})$ is the expected $\mathbf{F}(\mathbf{x_i}, \mathbf{Y})$ under the distribution $P^T_{\mathbf{\Lambda}}(\mathbf{Y}|\mathbf{x_i})$. The major difficulty in our optimization process is the computation of the gradient $\nabla\mathcal{L}(\mathbf{\Lambda})$ that leads to the problem of finding the expectation $\mathcal{E}_{P^T_{\mathbf{\Lambda}}(\mathbf{Y}|\mathbf{x_i})}\mathbf{F}(\mathbf{x_i}, \mathbf{Y})$. To efficiently calculate the expectation, we make use of the first-order independence assumption in Equation (4.4) and the decomposability of the error function $Err(\mathbf{y}, \mathbf{y'})$ in Equation (4.6). A variant of the forward-backward algorithm can be employed.

4.1.6 Forward-backward Algorithm

We first define two sets of variables that will be useful in computing the expectation: forward variables $\alpha_p(q)$ to represent the joint probability that the word at position p is labeled as q, and the words from position 1 to p appear; backward variables $\beta_p(q)$ to represent the conditional probability that given the word at position p is labeled as q, the words from position

$(p+1)$ to the end appear. Let y_p be the correct label at position p. Each variable can be recursively calculated by the following equations:

$$\alpha_p(q) = \sum_{y' \in \Omega} \alpha_{p-1}(y') e^{\mathbf{\Lambda} \cdot \mathbf{f}(\mathbf{x}, y', q, p) + I(y_p \neq q)}$$

$$\beta_p(q) = \sum_{y' \in \Omega} e^{\mathbf{\Lambda} \cdot \mathbf{f}(\mathbf{x}, q, y', p+1) + I(y_{p+1} \neq y')} \beta_{p+1}(y')$$

where $\alpha_0(\cdot)$ and $\beta_{|\mathbf{x}|}(\cdot)$ are both initialized to 1. With the two sets of variables, we can efficiently compute the normalization factor $Z^T_{\mathbf{\Lambda}}(\mathbf{x})$ and the expectation $\mathcal{E}_{P^T_{\mathbf{\Lambda}}(\mathbf{Y}|\mathbf{x_i})}\mathbf{F}(\mathbf{x_i}, \mathbf{Y})$.

$$Z^T_{\mathbf{\Lambda}}(\mathbf{x}) = \sum_{q=1}^{|\Omega|} \alpha_{|\mathbf{x}|}(q) \tag{4.11}$$

$$\begin{aligned} &\mathcal{E}_{P^T_{\mathbf{\Lambda}}(\mathbf{Y}|\mathbf{x_i})} F_k(\mathbf{x_i}, \mathbf{Y}) \\ =& \sum_{\mathbf{y}' \in \mathcal{Y}} \frac{F_k(\mathbf{x_i}, \mathbf{y}') e^{\mathbf{\Lambda} \cdot \mathbf{F}(\mathbf{x_i}, \mathbf{y}') + Err(\mathbf{y_i}, \mathbf{y}')}}{Z^T_{\mathbf{\Lambda}}(\mathbf{x_i})} \\ =& \sum_{p=1}^{|\mathbf{x}|} \sum_{q', q \in \Omega} \frac{f_k(\mathbf{x_i}, q', q, p) \alpha_{p-1}(q') e^{\mathbf{\Lambda} \cdot \mathbf{f}(\mathbf{x_i}, q', q, p) + I(y_p \neq q)} \beta_p(q)}{Z^T_{\mathbf{\Lambda}}(\mathbf{x_i})} \end{aligned} \tag{4.12}$$

Note that with the help of the forward and backward variables, the number of operations is changed from being exponential in the length of a sentence $\mathbf{x}$ (i.e. $|\Omega|^{|\mathbf{x}|}$) to being linear in $|\mathbf{x}|$. The objective value $\mathcal{L}(\mathbf{\Lambda})$ and the gradient $\nabla \mathcal{L}(\mathbf{\Lambda})$ can be calculated efficiently with the normalization factor $Z^T_{\mathbf{\Lambda}}(\mathbf{x})$ and the expectation $\mathcal{E}_{P^T_{\mathbf{\Lambda}}(\mathbf{Y}|\mathbf{x_i})}\mathbf{F}(\mathbf{x_i}, \mathbf{Y})$ provided. With these two pieces of information, we can use the L-BFGS algorithm [28], which is a limited-memory quasi-Newton method for conducting large-scale nonlinear optimization, to estimate the parameter $\mathbf{\Lambda}$. The L-BFGS algorithm has a fast convergence rate and has been shown to achieve good performance in estimating parameters for maximum entropy models [29].

In practice, we need to add regularization terms to the log-likelihood to avoid overfitting. Assume a Gaussian prior with mean 0 and covariance $\sigma^2 I$,

$$\begin{aligned}
\mathcal{L}(\mathbf{\Lambda}) &= \sum_i \mathbf{\Lambda} \cdot \mathbf{F}(\mathbf{x_i}, \mathbf{y_i}) - \log Z^T_{\mathbf{\Lambda}}(\mathbf{x_i}) - \frac{\|\mathbf{\Lambda}\|^2}{2\sigma^2} \\
\nabla\mathcal{L}(\mathbf{\Lambda}) &= \sum_i \mathbf{F}(\mathbf{x_i}, \mathbf{y_i}) - \mathcal{E}_{P^T_{\mathbf{\Lambda}}(\mathbf{Y}|\mathbf{x_i})}\mathbf{F}(\mathbf{x_i}, \mathbf{Y}) - \frac{\|\mathbf{\Lambda}\|}{\sigma^2}
\end{aligned}$$

The role of regularization terms is particularly important in our model. If we consider Equation (4.22), it is easier to satisfy these constraints with large values of λ_i. That means a parameter $\mathbf{\Lambda}$ with a larger norm is preferred without regularization, which usually gives an overfitting model. In our experiments, a smaller σ (i.e. a heavier penalty for a larger norm) tends to give better performance.

4.1.7 Grid Search for σ and κ

In order to give appropriate values for the regularization parameter σ and the error weight κ, we propose a grid search algorithm to find the appropriate parameter values. The idea is to split the training data into a training set and a validation set and divide the two-dimensional search space for σ and κ

into large grids. In each grid, we use the center of the grid to train the model on the training set and test it on the validation set. We choose the grid that gives the best performance and continue to divide it into smaller grids. This process is repeated until there is no improvement on the performance or the performance is satisfied. The algorithm is presented in Algorithm 1

Algorithm 1 Grid-search for σ and κ.

```
1: INPUT: search range for $\sigma$: $[\sigma_a, \sigma_b]$ and $\kappa$: $[\kappa_a, \kappa_b]$
2: while true do
3:     set $\sigma_m = \frac{\sigma_a+\sigma_b}{2}, \quad \kappa_m = \frac{\kappa_a+\kappa_b}{2}$
4:     set $\sigma_{m1} = \frac{\sigma_a+\sigma_m}{2}, \quad \sigma_{m2} = \frac{\sigma_m+\sigma_b}{2}$
5:     set $\kappa_{m1} = \frac{\kappa_a+\kappa_m}{2}, \quad \kappa_{m2} = \frac{\kappa_m+\kappa_b}{2}$
6:     $MaxP = -\infty$
7:     for $p = 1$ to 2 do
8:         for $q = 1$ to 2 do
9:             Train the model on the training set with parameter $(\sigma_{mp}, \kappa_{mq})$
10:            Test the model on the validation set and get performance $P$
11:            if $P > MaxP$ then
12:                $MaxP = P$
13:                $\sigma = \sigma_{mp}$ and $\kappa = \kappa_{mq}$
14:            end if
15:        end for
16:    end for
17:    terminate if no improvement
18:    $\sigma_a = \min(\sigma_a, \sigma), \quad \sigma_b = \max(\sigma_b, \sigma)$
19:    $\kappa_a = \min(\kappa_a, \kappa), \quad \kappa_b = \max(\kappa_b, \kappa)$
20: end while
```

4.1.8 Inference

We divide the named entity recognition task into a segmentation task and a classification task. In the segmentation task, a sentence $\mathbf{x}$ is segmented and possible segments of named entities are identified. In the classification task, the identified segments are classified into one of the possible named entity types or rejected.

In other words, in the segmentation task, the segments in a sentence $\mathbf{x}$ are identified by

$$\hat{\mathbf{y}}_{\mathbf{s}} = \arg\max_{\mathbf{y}'} \mathbf{w}_{\mathbf{s}} \cdot \mathbf{F}_{\mathbf{s}}(\mathbf{x}, \mathbf{y}') \tag{4.13}$$

where $\mathbf{F}_{\mathbf{s}}(\cdot)$ is the set of segment features and $\mathbf{w}_{\mathbf{s}}$ is the parameter for segmentation.

In the classification task, the segments (that can be identified by $\mathbf{y}_{\mathbf{s}}$) in a sentence $\mathbf{x}$ are classified by

$$\hat{\mathbf{y}}_{\mathbf{c}} = \arg\max_{\mathbf{y}'} \mathbf{w}_{\mathbf{c}} \cdot \mathbf{F}_{\mathbf{c}}(\mathbf{x}, \mathbf{y}_{\mathbf{s}}, \mathbf{y}') \tag{4.14}$$

where $\mathbf{F}_{\mathbf{c}}(\cdot)$ is the set of classification features and $\mathbf{w}_{\mathbf{c}}$ is the parameter for classification.

First-order Independence

In this cascaded framework, the number of possible labels in the segmentation task is N_s. For example, $N_s = 3$ in the IOB2 notation. In the classification task, the number of possible labels is N_c which is the number of entity types. Following the first-order independence assumption, the maximum total number of features in the two tasks is $O(\max(N_s^2, N_c^2))$, which is much smaller than the single-phase approach where the total number of features is $O((N_s N_c)^2)$.

Another potential advantage of dividing the NER task into two tasks is that it allows greater flexibility in choosing an appropriate set of features for each task. In fact, adding more features may not necessarily increase the performance. [39] reported a system using a subset of features outperformed that using a full set of features.

Adapted Decision Algorithm for MEMB Model

In MEMB model, the decision algorithm is an adapted training algorithm that reflects the unavailability of the true output $\mathbf{y_i}$; in other words, we do not know the true output $\mathbf{y_i}$ and thus no information between $\mathbf{y_i}$ and a possible output $\mathbf{y}'$ is known.

$$\Pr_{decision}(\mathbf{y}'|\mathbf{x_i};\mathbf{\Lambda}) = \frac{\exp[\mathbf{\Lambda}\cdot\mathbf{F}(\mathbf{x_i},\mathbf{y}')]}{Z_{decision}(\mathbf{x_i};\mathbf{\Lambda})} \tag{4.15}$$

where $\mathbf{\Lambda}$ is the model parameter that we get from the training stage, $Z_{decision}(\mathbf{x_i};\mathbf{\Lambda}) = \sum_{\mathbf{y}'\in\mathcal{Y}} \exp[\mathbf{\Lambda}\cdot\mathbf{F}(\mathbf{x_i},\mathbf{y}')]$ is the normalization factor.

For convenience, we write $\Pr_{training}(\mathbf{y}'|\mathbf{x_i};\mathbf{\Lambda})$ as $P_{\mathbf{\Lambda}}^{T}(\mathbf{y}'|\mathbf{x_i})$ and $\Pr_{decision}(\mathbf{y}'|\mathbf{x_i};\mathbf{\Lambda})$

as $P^D_{\mathbf{\Lambda}}(\mathbf{y'}|\mathbf{x_i})$, and similarly for the normalization factors $Z_{train}(\mathbf{x_i};\mathbf{\Lambda})$ and $Z_{decision}(\mathbf{x_i};\mathbf{\Lambda})$ as $Z^T_{\mathbf{\Lambda}}(\mathbf{x_i})$ and $Z^D_{\mathbf{\Lambda}}(\mathbf{x_i})$ respectively. In the decision stage, we extract the named entities in the input sentence $\mathbf{x_i}$ by finding the output $\hat{\mathbf{y}}$ with the highest probability $P^D_{\mathbf{\Lambda}}(\mathbf{y'}|\mathbf{x_i})$ with parameter $\mathbf{\Lambda}$.

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y'}} P^D_{\mathbf{\Lambda}}(\mathbf{y'}|\mathbf{x_i}) \tag{4.16}$$

In case there is a tie, we assume there is an ordering in $\mathcal{Y}$ for choosing $\hat{\mathbf{y}}$, such as the number or the maximum length of named entities. The same operation can be performed in the log space. Equation (4.16) can be written as

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y'}} \log P^D_{\mathbf{\Lambda}}(\mathbf{y'}|\mathbf{x_i}) \tag{4.17}$$

$$= \arg\max_{\mathbf{y'}} \mathbf{\Lambda}\cdot\mathbf{F}(\mathbf{x_i},\mathbf{y'}) \tag{4.18}$$

Now we are ready to investigate the design and significance of the error measure function $Err(\mathbf{y_i},\mathbf{y'})$ in the training model. Suppose in training, for a training instance $(\mathbf{x_i},\mathbf{y_i})$ the model gives the highest probability correctly to $\mathbf{y_i}$, i.e. $\hat{\mathbf{y}} = \mathbf{y_i}$,

$$\max_{\mathbf{y'}} \log P^T_{\mathbf{\Lambda}}(\mathbf{y'}|\mathbf{x_i}) \tag{4.19}$$

$$= \mathbf{\Lambda}\cdot\mathbf{F}(\mathbf{x_i},\mathbf{y_i}) + Err(\mathbf{y_i},\mathbf{y_i}) \tag{4.20}$$

$$= \mathbf{\Lambda}\cdot\mathbf{F}(\mathbf{x_i},\mathbf{y_i}) \tag{4.21}$$

where $Err(\mathbf{y_i}, \mathbf{y_i}) = 0$ by definition. We can conclude $\forall \mathbf{y}'$:

$$\mathbf{\Lambda} \cdot \mathbf{F}(\mathbf{x_i}, \mathbf{y_i}) \geq \mathbf{\Lambda} \cdot \mathbf{F}(\mathbf{x_i}, \mathbf{y}') + Err(\mathbf{y_i}, \mathbf{y}') \tag{4.22}$$

$$\mathbf{\Lambda} \cdot \mathbf{F}(\mathbf{x_i}, \mathbf{y_i}) - \mathbf{\Lambda} \cdot \mathbf{F}(\mathbf{x_i}, \mathbf{y}') \geq Err(\mathbf{y_i}, \mathbf{y}') \tag{4.23}$$

This is analogous to training the parameter $\mathbf{\Lambda}$ with margin $Err(\mathbf{y_i}, \mathbf{y}')$ in margin-based training (although $\mathbf{\Lambda}$ may not necessarily give the maximum margin in our case). When the training model gives the highest probability to the output $\mathbf{y_i}$ for the input $\mathbf{x_i}$, it also gives at least a margin defined by $Err(\mathbf{y_i}, \mathbf{y}')$ between all possible $\mathbf{y}' \in \mathcal{Y}$ and $\mathbf{y_i}$ for the input $\mathbf{x_i}$.

This analysis can be applied to the CLASS module in a similar manner.

4.2 Isomeric CRF (ICRF) Model

Based on the cascaded modeling framework, we further investigate the cause of cascaded errors. To simplify our discussion, suppose we have two sub-tasks to perform. (The following discussion can be extended to N sub-tasks trivially.) In the following discussion, we also change our notation to indicate that we are now focusing on the ICRF model when necessary.

The first sub-task aims at predicting Y based on X, and the second sub-task is to predict Z based on X and Y. We can put the above problems into a probabilistic framework as follows.

The foundation of cascaded approach comes from the chain rule of probability:

$$\begin{aligned} \mathrm{P}(Y,Z|X) &= \frac{\mathrm{P}(X,Y,Z)}{\mathrm{P}(X)} && (4.24)\\ &= \frac{\mathrm{P}(X,Y,Z)}{\mathrm{P}(X,Y)}\frac{\mathrm{P}(X,Y)}{\mathrm{P}(X)} && (4.25)\\ &= \mathrm{P}(Z|X,Y)\mathrm{P}(Y|X) && (4.26) \end{aligned}$$

(Note: in some problems, $\mathrm{P}(Z|X,Y) = \mathrm{P}(Z|Y)$) In practice, the probability distributions for $\mathrm{P}(Z|X,Y)$ and $\mathrm{P}(Y|X)$ in a cascaded approach will be modeled by different probabilistic models. They may differ in the model families used or the feature representations. Furthermore, each model will have its own parameters. Therefore, a more precise description will be:

$$\begin{aligned} \mathrm{P}(Y,Z|X) &= \frac{\mathrm{P}(X,Y,Z)}{\mathrm{P}(X)} && (4.27)\\ &= \frac{\mathrm{P}(X,Y,Z)}{\mathrm{P}(X,Y)}\frac{\mathrm{P}(X,Y)}{\mathrm{P}(X)} && (4.28)\\ &\simeq \frac{\mathrm{P}_\theta(X,Y,Z)}{\mathrm{P}_\theta(X,Y)}\frac{\mathrm{Q}_\phi(X,Y)}{\mathrm{Q}_\phi(X)} && (4.29)\\ &= \mathrm{P}_\theta(Z|X,Y)\mathrm{Q}_\phi(Y|X) && (4.30) \end{aligned}$$

where $P(\cdot|\cdot)$ is a model family for the relationship between Z and (X,Y) with θ being its parameter, and $Q(\cdot|\cdot)$ is a model family for the relationship between Y and X with ϕ being its parameter.

The above transformation actually imposes a constraint :

$$\mathrm{P}_\theta(X, Y) = \mathrm{Q}_\phi(X, Y) \tag{4.31}$$

under the parameter values (θ, ϕ) for the two probability distributions. The difference between a separately-trained approach and a jointly-trained approach is presented schematically in Figure 4.3 and Figure 4.2.

From here, we will need a specification for the model families in order to proceed. A natural choice for the conditional probability distributions above will be a conditional random field (CRF) model [41], where X is an observation and (Y, Z) is the answers that we are going to predict.

In a sequence labeling problem, Y and Z correspond to different tags that we are interested in. For example, X is a sentence consisting of a sequence of words, Y is a sequence of POS tags corresponding to every word in the sentence, and Z is a sequence of NP-chunking tags for identifying NP chunks in the sentence.

4.2.1 Markov Properties

To see why we want to model two conditional distributions instead of the original conditional distribution, we turn our attention to the practical implementation. One typical requirement for a CRF model $\mathrm{P}(U|V)$ to be tractable in its inference stage is to assume a first-order [12] independence of U, i.e. if

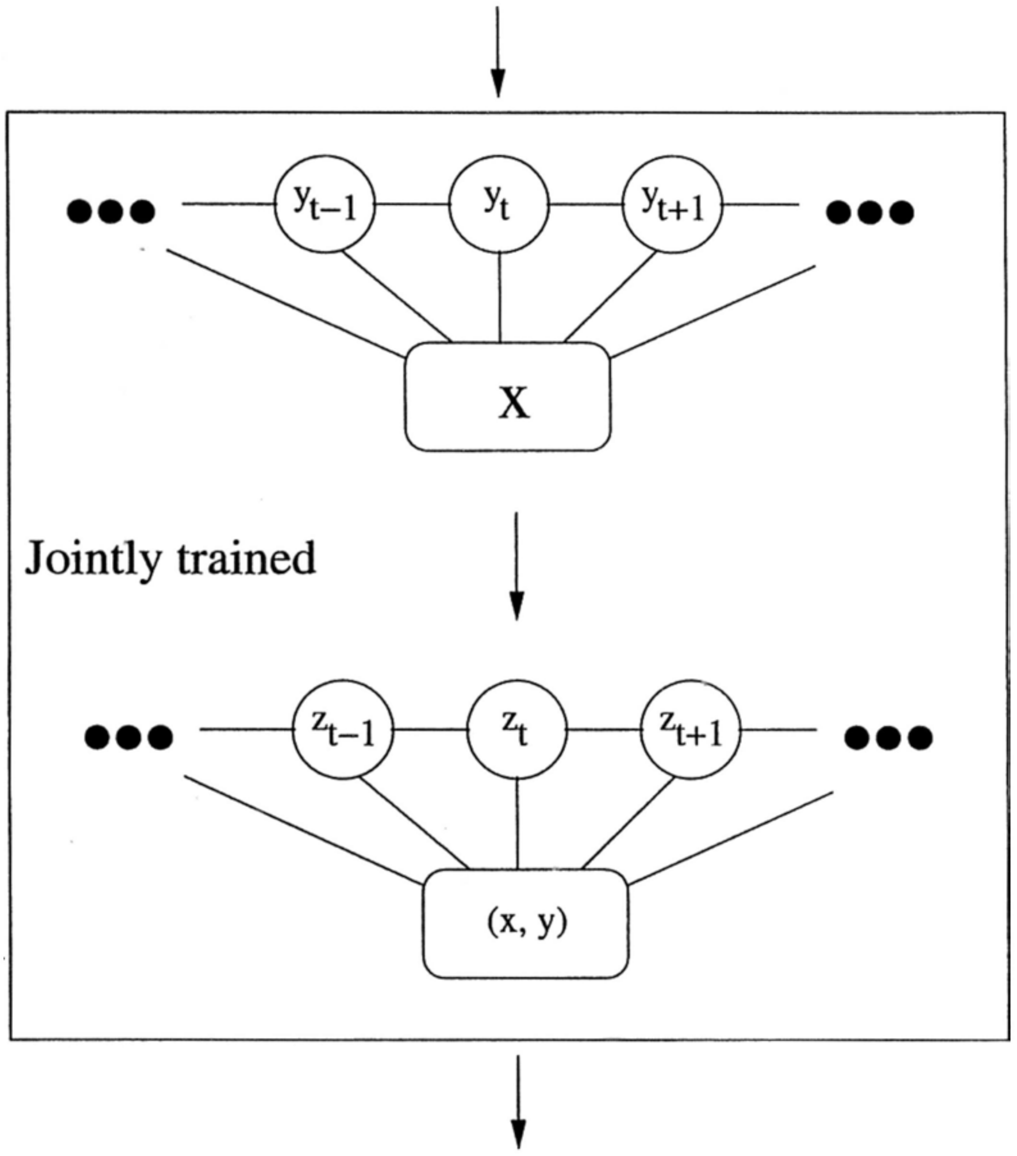

Figure 4.2: A schematic representation of a jointly-trained framework for information extraction. The two models are jointly trained. Each model is optimized in a way that the overall system performance is maximized.

$U =< u_1, u_2, \ldots, u_n >$, then $P(u_t|V, u_{t-1}, \ldots, u_1) = P(u_t|V, u_{t-1})$. Theoretically speaking, our model does not prohibit the use of higher-order CRFs.

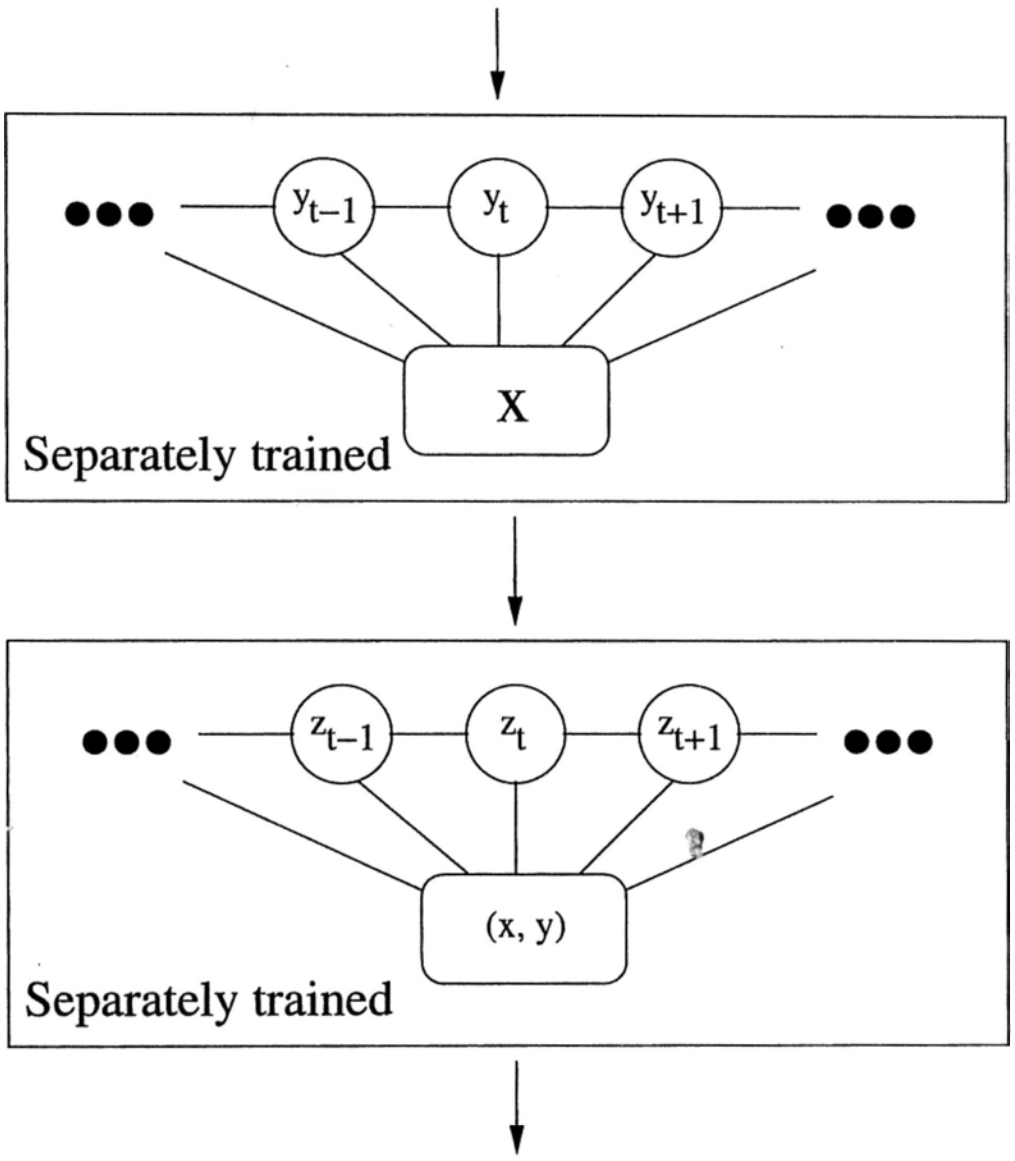

Figure 4.3: A schematic representation of a separately-trained framework for information extracting. The two models are separately trained and cascaded together to solve a single task. The performance of each model is not optimized to benefit each other. Cascaded errors can easily occur which leads to poor overall system performance.

However in practice, a higher-order model needs a lot more data to estimate its parameters and a bigger cost for training (e.g. longer training time and larger memory space).

If we model the conditional distribution $P(Y, Z|X)$ directly, the random variables (Y, Z) together have to satisfy this Markov property. However, if we model the two conditional distributions separately, in each model only one random variable is required to satisfy this first-order independence assumption. This implies that we can make use of more information from Y in $P_\theta(Z|X, Y)$ to predict Z without adding too much computation load.

4.2.2 Template Specification

When we specify a CRF for a sequence labeling problem, we need to specify different feature functions to be evaluated on an instance X. However, manually specifying each feature function for a large-scale problem is time-consuming and not scalable. This is usually done by specifying a template that will be turned into a set of feature functions.

By modeling two conditional probability distributions instead of one, it will be more flexible for each model to choose its own specific template for its own modeling goal. The number of features also grows with the square of the number of values for a random variable. In other words, if $y \in \mathcal{Y}$, $z \in \mathcal{Z}$, and there are $m|\mathcal{Y}|^2$ features for y and $m|\mathcal{Z}|^2$ for z, then in one single CRF there will be approximately $m(|\mathcal{Y}| + |\mathcal{Z}|)^2$ features.. This actually means there will be fewer total number of parameters in a cascaded approach.

4.2.3 Training Time

Two important efficient algorithms that are used as inference algorithms in a CRF are the forward-backward and viterbi algorithms. The time complexities of both grow linearly with the square of the number of states. As stated in the previous section, the greater number of states in a single CRF approach will suffer a longer training time in a cascaded approach.

4.2.4 Joining two CRF Models

Following the description in the previous section, suppose we have two CRFs defined by

$$\mathrm{P}_\theta(z|x, y) = \frac{\exp[\theta \cdot \mathbf{F}(\mathbf{x}, \mathbf{y}, \mathbf{z})]}{Z_\theta(\mathbf{x}, \mathbf{y})} \tag{4.32}$$

$$\mathrm{Q}_\phi(y|x) = \frac{\exp[\phi \cdot \mathbf{G}(\mathbf{x}, \mathbf{y})]}{Z_\phi(\mathbf{x})} \tag{4.33}$$

We can then define an isomeric CRF to be

$$\mathrm{R}_\lambda(y, z|x) = \frac{\exp[\lambda \cdot \mathbf{H}(\mathbf{x}, \mathbf{y}, \mathbf{z})]}{Z_\lambda(\mathbf{x}, \mathbf{y})} \tag{4.34}$$

$$= \mathrm{P}_\theta(z|x, y)\mathrm{Q}_\phi(y|x) \tag{4.35}$$

where $\lambda =< \theta, \phi >$, $\mathbf{H} =< \mathbf{F}, \mathbf{G} >$, and $Z_\lambda(\mathbf{x}, \mathbf{y}) = Z_\theta(\mathbf{x}, \mathbf{y}) Z_\phi(\mathbf{x})$.

Claim: Under the assumption that the term $\exp[\phi \cdot \mathbf{G}(\mathbf{x}, \mathbf{y})]$ and the normalization factor $Z_\theta(\mathbf{x}, \mathbf{y})$ are equal, the distribution $\mathrm{R}_\lambda(\mathbf{y}, \mathbf{z}|\mathbf{x})$ recovers a conditional distribution for $(\mathbf{y}, \mathbf{z})$ conditioned on $\mathbf{x}$.

Proof: When $\exp[\phi \cdot \mathbf{G}(\mathbf{x}, \mathbf{y})] = Z_\theta(\mathbf{x}, \mathbf{y})$, the distribution $\mathrm{R}_\lambda(\mathbf{y}, \mathbf{z}|\mathbf{x})$ can be written as

$$\begin{aligned}
\frac{\exp[\lambda \cdot \mathbf{H}(\mathbf{x}, \mathbf{y}, \mathbf{z})]}{Z_\lambda(\mathbf{x}, \mathbf{y})} &= \frac{\exp[\theta \cdot \mathbf{F}(\mathbf{x}, \mathbf{y}, \mathbf{z})] \exp[\phi \cdot \mathbf{G}(\mathbf{x}, \mathbf{y})]}{Z_\theta(\mathbf{x}, \mathbf{y}) Z_\phi(\mathbf{x})} && (4.36)\\
&= \frac{\exp[\theta \cdot \mathbf{F}(\mathbf{x}, \mathbf{y}, \mathbf{z})] Z_\theta(\mathbf{x}, \mathbf{y})}{Z_\theta(\mathbf{x}, \mathbf{y}) Z_\phi(\mathbf{x})} && (4.37)\\
&= \frac{\exp[\theta \cdot \mathbf{F}(\mathbf{x}, \mathbf{y}, \mathbf{z})] Z_\theta(\mathbf{x}, \mathbf{y})}{Z_\theta(\mathbf{x}, \mathbf{y}) \sum_{\mathbf{y}' \in \mathcal{Y}} Z_\theta(\mathbf{x}, \mathbf{y}')} && (4.38)\\
&= \frac{\exp[\theta \cdot \mathbf{F}(\mathbf{x}, \mathbf{y}, \mathbf{z})]}{\sum_{\mathbf{y}' \in \mathcal{Y}} \sum_{\mathbf{z}' \in \mathcal{Z}} \exp[\theta \cdot \mathbf{F}(\mathbf{x}, \mathbf{y}', \mathbf{z}')]} && (4.39)\\
&= \mathrm{R}(\mathbf{y}, \mathbf{z}|\mathbf{x}) && (4.40)
\end{aligned}$$

where in the second last line we cancel out the factor $Z_\theta(\mathbf{x}, \mathbf{y})$ and substitute for the definition of $Z_\theta(\mathbf{x}, \mathbf{y}')$ in the summation term.

4.2.5 Joint Training for an isomeric CRF

As in the case of a general CRF, we define the log-likelihood $\mathcal{L}(\lambda)$ of $\mathrm{R}_\lambda(\mathbf{y},\mathbf{z}|\mathbf{x})$ to be

$$\mathcal{L}(\lambda) \tag{4.41}$$

$$= \log \mathrm{R}_\lambda(\mathbf{y},\mathbf{z}|\mathbf{x}) \tag{4.42}$$

$$= \log \mathrm{P}_\theta(\mathbf{z}|\mathbf{x},\mathbf{y})\mathrm{Q}_\phi(\mathbf{y}|\mathbf{x}) \tag{4.43}$$

$$= \theta \cdot \mathbf{F}(\mathbf{x},\mathbf{y},\mathbf{z}) - \log Z_\theta(\mathbf{x},\mathbf{y}) + \phi \cdot \mathbf{G}(\mathbf{x},\mathbf{y}) - \log Z_\phi(\mathbf{x}) \tag{4.44}$$

$$= \lambda \cdot \mathbf{H}(\mathbf{x},\mathbf{y},\mathbf{z}) - \log Z_\lambda(\mathbf{x},\mathbf{y}) \tag{4.45}$$

If we look at the last line, the log-likelihood for an isomeric CRF looks exactly the same as a general CRF. That is why we term it an isomeric CRF. We also have to add the constraint $\exp[\phi \cdot \mathbf{G}(\mathbf{x},\mathbf{y})] = Z_\theta(\mathbf{x},\mathbf{y})$ for each training instance in the optimization process. However as in the case of other machine learning algorithms (e.g. support vector machines (SVMs)), a hard constraint is rarely satisfied in real-world problems. Therefore we use a soft constraint approach to implement the constraint $\exp[\phi \cdot \mathbf{G}(\mathbf{x},\mathbf{y})] = Z_\theta(\mathbf{x},\mathbf{y})$ by adding a penalty term $\frac{D^2}{2} = \frac{|\phi \cdot \mathbf{G}(\mathbf{x},\mathbf{y})] - \log Z_\theta(\mathbf{x},\mathbf{y})|^2}{2}$ to the log-likelihood

$$\mathcal{L}(\lambda) \tag{4.46}$$

$$= \theta \cdot \mathbf{F}(\mathbf{x},\mathbf{y},\mathbf{z}) - \log Z_\theta(\mathbf{x},\mathbf{y}) + \phi \cdot \mathbf{G}(\mathbf{x},\mathbf{y}) - \log Z_\phi(\mathbf{x}) \tag{4.47}$$

$$+ \frac{C}{2}|\phi \cdot G(x,y)] - \log Z_\theta(\mathbf{x},\mathbf{y})|^2 \tag{4.48}$$

where C is the penalty cost for the violation of the constraint. Note that by this constraint, the two CRF models P and Q are linked together. For example, if we try to maximize solely the log-likelihood of P without considering Q, the resulting inconsistency between P and Q will show up as a non-zero difference D. Therefore, the optimization process has to adjust the CRF model Q to match the CRF model P in order to reduce the difference D. When the two CRF models are "consistent", the log-likelihood is also maximized with the difference D being minimized. The penalty cost C controls the tolerance for this inconsistency and usually depends on the characteristics of the data.

We seek to maximize the log-likelihood $\mathcal{L}(\lambda)$ over training instances $(\mathbf{x_i}, \mathbf{y_i}, \mathbf{z_i})$ by performing a gradient ascent. The gradient is broken down into two parts, corresponding to the parameter θ and ϕ.

$$\nabla_\theta \mathcal{L}(\lambda) \tag{4.49}$$

$$= \sum_i \mathbf{F}(\mathbf{x_i}, \mathbf{y_i}, \mathbf{z_i}) - (1 + CD)\mathcal{E}_{P_\theta(\mathbf{Z}|\mathbf{x_i},\mathbf{y_i})}\mathbf{F}(\mathbf{x_i}, \mathbf{y_i}, \mathbf{Z}) \tag{4.50}$$

$$\nabla_\phi \mathcal{L}(\lambda) \tag{4.51}$$

$$= \sum_i (1 + CD)\mathbf{G}(\mathbf{x_i}, \mathbf{y_i}) - \mathcal{E}_{Q_\phi(\mathbf{Y}|\mathbf{x_i})}\mathbf{G}(\mathbf{x_i}, \mathbf{Y}) \tag{4.52}$$

where $\mathcal{E}(\cdot)$ is the conditional expectation:

$$\mathcal{E}_{P_\theta(\mathbf{Z}|\mathbf{x_i},\mathbf{y_i})}\mathbf{F}(\mathbf{x_i},\mathbf{y_i},\mathbf{Z}) = \frac{\sum_{\mathbf{z'}\in\mathcal{Z}}\mathbf{F}(\mathbf{x_i},\mathbf{y_i},\mathbf{z'})e^{\theta\cdot\mathbf{F}(\mathbf{x_i},\mathbf{y_i},\mathbf{z'})}}{Z_\theta(\mathbf{x_i},\mathbf{y_i})} \tag{4.53}$$

$$\mathcal{E}_{Q_\phi(\mathbf{Y}|\mathbf{x_i})}\mathbf{G}(\mathbf{x_i},\mathbf{Y}) = \frac{\sum_{\mathbf{y'}\in\mathcal{Y}}\mathbf{G}(\mathbf{x_i},\mathbf{y'})e^{\phi\cdot\mathbf{G}(\mathbf{x_i},\mathbf{y'})}}{Z_\phi(\mathbf{x_i})}\nabla_\phi\mathcal{L}(\lambda) \tag{4.54}$$

The expectation term $\mathcal{E}(\cdot)$ can be calculated efficiently by the forward-backward algorithm, in the case of a linear-chain CRF. The total gradient can be obtained by concatenating the two gradients $\nabla_\lambda\mathcal{L}(\lambda) =< \nabla_\theta\mathcal{L}(\lambda),\ \nabla_\phi\mathcal{L}(\lambda) >$.

We use a Gaussian prior for the parameters θ and ϕ. Thus we also need to add the regularization terms $\frac{\|\theta\|^2}{-2\sigma_\theta}$ and $\frac{\|\phi\|^2}{-2\sigma\phi}$ to the log-likelihood $\mathcal{L}(\lambda)$. As such, we also need to add the term $\frac{\|\theta\|}{-\sigma_\theta}$ and $\frac{\|\phi\|}{-\sigma\phi}$ to the total gradient $\nabla_\lambda\mathcal{L}(\lambda)$, which corresponds to the gradients of the regularization terms.

For each training instance $(\mathbf{x_i},\mathbf{y_i},\mathbf{z_i})$, we calculate the normalization term Z, the conditional expectation $\mathcal{E}(\cdot)$ and thus the log-likelihood $\mathcal{L}$. After we get D for this instance, we scale the corresponding parts of the gradients in $\nabla_\theta\mathcal{L}(\lambda)$ and $\nabla_\phi\mathcal{L}(\lambda)$. We use the L-BFGS algorithm for doing this large-scale optimization because it has been proved to perform well and achieve state-of-the-art performance in many real-world problems.

In this formulation, we achieve joint training without the need of using an approximation inference algorithm such as belief propagation. All the existing efficient algorithms (e.g. forward-backward algorithm, viterbi, L-BFGS) for a linear-chain CRF can be reused. Because joint training with the soft constraint implies that both CRF models have to output its predicted values with the highest score in order to get maximum system performance, we can simply use viterbi algorithm on both models in the testing stage.

4.2.6 Parallel Training for an isomeric CRF

To further improve the speed of training an isomeric CRF, we propose a parallel training algorithm to do parameter estimation for an isomeric CRF. The main idea is to structure the problem of estimating parameters for the whole training set into a sub-problem of estimating parameters for one training instance. By making use of the multi-core technology on a CPU or the GPU architecture on a graphics card, we can easily achieve a speedup of 10x or more.

If we look at the derivation of the gradient of the log-likelihood for an isomeric CRF,

$$\nabla_\theta \mathcal{L}(\lambda) \tag{4.55}$$

$$= \sum_i \mathbf{F}(\mathbf{x_i}, \mathbf{y_i}, \mathbf{z_i}) - (1 + CD)\mathcal{E}_{P_\theta(\mathbf{Z}|\mathbf{x_i},\mathbf{y_i})}\mathbf{F}(\mathbf{x_i}, \mathbf{y_i}, \mathbf{Z}) \tag{4.56}$$

$$\nabla_\phi \mathcal{L}(\lambda) \tag{4.57}$$

$$= \sum_i (1 + CD)\mathbf{G}(\mathbf{x_i}, \mathbf{y_i}) - \mathcal{E}_{Q_\phi(\mathbf{Y}|\mathbf{x_i})}\mathbf{G}(\mathbf{x_i}, \mathbf{Y}) \tag{4.58}$$

we can see that for each training instance $\mathbf{x_i}$ the calculations are independent of each other. This insight is valuable and it suggests that a parallel training algorithm is possible. We present a parallel training algorithm for an isomeric CRF as follows.

For each training instance $(\mathbf{x_i}, \mathbf{y_i}, \mathbf{z_i})$, we need to calculate the following values:

1. the weighted sum of feature functions in P: $S_\theta = \theta \cdot \mathbf{F}(\mathbf{x_i}, \mathbf{y_i}, \mathbf{z_i})$
2. the weighted sum of feature functions in Q: $S_\varphi = \phi \cdot \mathbf{G}(\mathbf{x_i}, \mathbf{y_i})$
3. the normalization factor Z in P: $Z_\theta(\mathbf{x_i}, \mathbf{y_i})$
4. the normalization factor Z in Q: $Z_\phi(\mathbf{x_i}, \mathbf{y_i})$
5. the expectation with respect to P: $\mathcal{E}_{P_\theta}\mathbf{F}$
6. the expectation with respect to Q: $\mathcal{E}_{Q_\theta}\mathbf{G}$

After we divide the calculations in the above manner, we can assign one thread to each of the training instance and perform calculations in parallel. The details of this parallel training algorithm for an isomeric CRF is presented in Algorithm 2. The keyword **pardo** means a dedicated thread is to run the enclosed statements sequentially.

Algorithm 2 Parallel Training for an isomeric CRF

```
1: INPUT:  $N$ training instances $\{(\mathbf{x_i}, \mathbf{y_i}, \mathbf{z_i})\}$, no. of threads $M$
2: for $i = 0$ to $N - 1$ do
3:    Assign training instance $\{(\mathbf{x_i}, \mathbf{y_i}, \mathbf{z_i})\}$ to thread $(i \mod M)$
4: end for
5: while not converged do
6:    while not all threads done do
7:       pardo
8:          Calculate $S_\theta, S_\phi, Z_\theta, Z_\phi, \mathcal{E}_{P_\theta}\mathbf{F}, \mathcal{E}_{Q_\theta}\mathbf{G}$
9:          Calculate $\frac{D^2}{2} = \frac{(S_\phi - \log Z_\theta)^2}{2}$
10:         Calculate $\nabla_\theta = \mathbf{F} - (1 + CD)\mathcal{E}_{P_\theta}\mathbf{F}$
11:         Calculate $\nabla_\phi = (1 + CD)\mathbf{G} - \mathcal{E}_{Q_\phi}\mathbf{G}$
12:         Calculate $\mathcal{L}_i = S_\theta + S_\phi - Z_\theta - Z_\phi + C \cdot \frac{D^2}{2}$
13:         Calculate $\nabla_i =< \nabla_\theta, \nabla_\phi >$
14:      end pardo
15:   end while
16:   Calculate $\nabla = \sum_i \nabla_i$
17:   Calculate $\mathcal{L} = \sum_i \mathcal{L}_i$
18:   Perform L-BFGS optimization with objective value $\mathcal{L}$ and gradient $\nabla$
19: end while
```

4.3 Analysis of the constraints on the probability distributions in MEMB and ICRF

In both MEMB and ICRF models, our knowledge from a set of training data is represented by a probability distribution. If we can form this distribution, then our system can use decision theory to arrive at rational decisions. In other words, we need to assign a probability to each possible event. It is impossible to get the probabilities of all "possible" events in many real world problems. Some probabilities can be inferred from the training data or past experience but for many of them there may not be enough data or past experience to determine the "real" probabilities. The idea behind MEMB and ICRF models is to construct a single probability distribution over all possible outcomes, given certain constraints that we know from training data or past experience.

4.3.1 Constraint Sets and Distributions

To begin our analysis, let $\mathcal{E}$ be a sample space and $\psi_1, \psi_2, \ldots, \psi_n$ be a set of functions on $\mathcal{E}$. The set of constraints in our model can be expressed as:

$$\mathcal{C} = \{E[\psi_1(X)] = \mu_1, \tag{4.59}$$

$$E[\psi_2(X)] = \mu_2, \tag{4.60}$$

$$\ldots, \tag{4.61}$$

$$E[\psi_m(X)] = \mu_m\} \tag{4.62}$$

where μ_i is a value depending on the training data.

We want to find a probability distribution P such that it makes the best possible prediction of future data. That is, this probability distribution P should be the "most likely" to happen given our constraints. To find such a P, we would adopt the principle of maximum entropy. The entropy of a probability distribution P is defined as $-\log P(X)$. In our case, the P that maximizes the entropy given our constraints would be the "most likely" distribution. The entropy measures the inherent randomness in P and $\log P(x)$ actually describes the goodness-of-fit of a data instance x under P.

In the most simplest form, there are no constraints in our model P. Therefore $\mathcal{E} = x_1, x_2, \ldots, x_k$ and the P with maximum entropy is given by $P(x_i) = 1/k$. for all x_i. It is easy to see that the resulting entropy is $\log k$.

From this uniform distribution, we can infer that all other more skewed distributions give less entropy. In other words, data drawn from $\mathcal{E}$ with a more skewed distribution tends to be more regular than the uniform distribution. In the limiting case, $P(x_i) = 1$ and $P(x_j) = 0$ where $i \neq j$, the entropy of P is 0. When a skewed distribution is more far away from the uniform distribution $1/k$, the entropy decreases.

As another example, if we change our sample space $\mathcal{E}$ to $(-\infty, \infty)$ and add two constraints as follows:

$$E[X] = \mu \tag{4.63}$$

$$Var[X] = \sigma^2 \tag{4.64}$$

the distribution that gives the maximum entropy becomes the normal distribution with mean μ and variance σ^2.

4.3.2 Constraints in Sequence Labeling Problems

For a sequence $\mathbf{x}$ we can define a set of feature functions $f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})$. We can express our constraints in the following form:

$$\mathcal{C} = \{E[f_1(\mathbf{X})] = \mu_1, \tag{4.65}$$

$$E[f_2(\mathbf{X})] = \mu_2, \tag{4.66}$$

$$\ldots, \tag{4.67}$$

$$E[f_m(\mathbf{X})] = \mu_m\} \tag{4.68}$$

or simply in vector form:

$$\mathcal{C} = \{E[\mathbf{f}(\mathbf{X})] = \mathbf{u}\} \tag{4.69}$$

Therefore we can represent a constraint set for a set of sequence data as $\mathcal{C} = \mathcal{C}(\mathbf{f}, \mathbf{u})$. The set of probability distributions that satisfies our constraints in a sequence labeling problem is $\mathcal{P} = \{P \mid E_P[\mathbf{f}(\mathbf{X}) = \mathbf{u}]\}$.

However in real-world problems, we do not have the luxury to know the exact expected value for every constraint. Instead we use the empirical average values of the feature functions observed on a large training set of data. Therefore we assume that

$$E[f_i(\mathbf{X})] = \mu_i \approx \frac{1}{n}\sum_{j=1}^{n} f_i(\mathbf{x}_\mathrm{j}) \tag{4.70}$$

To be more precise, in real-world problems, we need to deal with the set of empirical constraints $\mathcal{C}_e(\mathbf{f}, \mathbf{u})$. Our goal now becomes getting the distribution

P_m that maximize the entropy subject to the constraints $\mathcal{C}_e(\mathbf{f}, \mathbf{u})$ which can be derived from a training data set.

We argue in the following that the set $\mathcal{P}$ is in the following form

$$P_m(x) = \frac{1}{Z(\mathbf{w})} \exp[\mathbf{w} \cdot \mathbf{f}(\mathbf{x})] \tag{4.71}$$

where $\mathbf{w} = (w_1, w_2, \ldots, w_m)$ is the parameters for a P_m, and $Z(\mathbf{w})$ is a normalization factor $Z(\mathbf{w}) = \sum_{\mathbf{x}} \exp[\mathbf{w} \cdot \mathbf{f}(\mathbf{x})]$.

Let Φ be a distribution that satisfies the constraint $\mathcal{C}_e(\mathbf{f}, \mathbf{u})$ and $\Phi \neq P_m$. The entropy of Φ is

$$H(\Phi) = E_\Phi[-\log \Phi(X)] \tag{4.72}$$

$$< E_\Phi[-\log P_m(X)] \tag{4.73}$$

$$= E_\Phi[-\mathbf{w} \cdot \mathbf{f}(\mathbf{x}) - \log Z(\mathbf{w})] \tag{4.74}$$

$$= -\mathbf{w} \cdot E_\Phi[\mathbf{f}(\mathbf{x})] - \log Z(\mathbf{w}) \tag{4.75}$$

$$= -\mathbf{w} \cdot \mathbf{u} - \log Z(\mathbf{w}) \tag{4.76}$$

However we also notice that

$$H(P_m) = E_{P_m}[-\log P_m(X)] \tag{4.77}$$

$$= E_{P_m}[-\mathbf{w} \cdot \mathbf{f}(\mathbf{x}) - \log Z(\mathbf{w})] \tag{4.78}$$

$$= -\mathbf{w} \cdot E_{P_m}[\mathbf{f}(\mathbf{x})] - \log Z(\mathbf{w}) \tag{4.79}$$

$$= -\mathbf{w} \cdot \mathbf{u} - \log Z(\mathbf{w}) \tag{4.80}$$

Therefore we can conclude $H(P_m) > H(\Phi)$ and P_m is the distribution that satisfies the constraints $\mathcal{C}_e(\mathbf{f}, \mathbf{u})$ with maximum entropy.

4.3.3 Stirling's Approximation

In this section we further analyze the relationship between the number of features and the set of probability distributions under our constraints.

Let us define frequency distribution α

$$\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_k) \text{ if } \sum_i^k \alpha_i = 1 \tag{4.81}$$

Let the number of training instances to be n and n_i is the number of $\mathbf{y}_i$ occurs in the training set $\{\mathcal{X}, \mathcal{Y}\}$. Clearly we can define $\alpha_i = n_i/n$ so that α is a frequency distribution. For convenience, we denote $\alpha_i(\mathcal{X}, \mathcal{Y}) = n_i/n$. We can see that the constraint $\mathcal{C}_e(\mathbf{f}, \mathbf{u})$ is satisfied if

$$\alpha_1 f_1(\mathbf{X}, \mathbf{Y}) + \alpha_2 f_1(\mathbf{X}, \mathbf{Y}) + \cdots + \alpha_k f_1(\mathbf{X}, \mathbf{Y}) = \mu_1 \tag{4.82}$$

$$\alpha_1 f_2(\mathbf{X}, \mathbf{Y}) + \alpha_2 f_2(\mathbf{X}, \mathbf{Y}) + \cdots + \alpha_k f_2(\mathbf{X}, \mathbf{Y}) = \mu_2 \tag{4.83}$$

$$\ldots \tag{4.84}$$

$$\alpha_1 f_m(\mathbf{X}, \mathbf{Y}) + \alpha_2 f_m(\mathbf{X}, \mathbf{Y}) + \cdots + \alpha_k f_m(\mathbf{X}, \mathbf{Y}) = \mu_m \tag{4.85}$$

Now we can define the set of training data with frequency distribution α

to be

$$\Upsilon^n(\alpha) = \Upsilon^n(\alpha_1, \alpha_2, \dots, \alpha_k) \tag{4.86}$$

$$= \{(\mathcal{X}, \mathcal{Y}) \mid \alpha(\mathcal{X}, \mathcal{Y}) = \alpha\} \tag{4.87}$$

We can count the number of distinct possible sets of training data by multinomial combinations:

$$\Upsilon^n(\alpha) = \binom{n}{n_1, n_2, \dots, n_k} \tag{4.88}$$

$$= \frac{n!}{n_1!n_2!\dots n_k!} \tag{4.89}$$

By using Stirling's approximation below:

$$\ln n! \approx n \ln n - n + \ln \sqrt{2\pi n} + O(\frac{1}{n}) \tag{4.90}$$

We can transform the multinomial combination :

$$\ln \Upsilon^n(\alpha) = \ln \frac{n!}{n_1! n_2! \ldots n_k!} \tag{4.91}$$

$$\approx n \ln n - n + \ln \sqrt{2\pi n} + O(\frac{1}{n}) \tag{4.92}$$

$$- (\ln n_1! + \ln n_2! + \cdots + \ln n_k!) \tag{4.93}$$

$$= n \ln n - n + \ln \sqrt{2\pi n} + O(\frac{1}{n}) \tag{4.94}$$

$$- (n_1 \ln n_1 - n_1 + \ln \sqrt{2\pi n_1} + O(\frac{1}{n_1})) \tag{4.95}$$

$$- (n_2 \ln n_2 - n_2 + \ln \sqrt{2\pi n_2} + O(\frac{1}{n_2})) \tag{4.96}$$

$$\ldots \tag{4.97}$$

$$- (n_k \ln n_k - n_k + \ln \sqrt{2\pi n_k} + O(\frac{1}{n_k})) \tag{4.98}$$

$$= (n_1 + n_2 + \cdots + n_k) \ln n - (n_1 \ln n_1 + n_2 \ln n_2 + \cdots + n_k \ln n_k) + c \tag{4.99}$$

$$= n_1 \ln \frac{n}{n_1} + n_2 \ln \frac{n}{n_2} + \cdots + n_k \ln \frac{n}{n_k} + c \tag{4.100}$$

$$= -n \sum_{i=1}^{k} \alpha_i \ln \alpha_i + \frac{c}{n} \tag{4.101}$$

$$= nH(\alpha) + c \tag{4.102}$$

$$\Upsilon^n(\alpha) \approx \exp[nH(\alpha) + \frac{c}{n}] \tag{4.103}$$

Let the set of all training data sets that satisfy constraint $C(\mathbf{f}, \mathbf{u})$ be Γ. If α_1 and α_2 are two frequency distributions such that $\Upsilon^n(\alpha_1)$ and $\Upsilon^n(\alpha_2)$ satisfy $C(\mathbf{f}, \mathbf{u})$, then $\Upsilon^n(\alpha_1)$ and $\Upsilon^n(\alpha_2)$ are subsets of Γ. When we look at the number of elements in each set and compare them, we find that the ratio is:

$$\frac{\|\Upsilon^n(\alpha_1)\|}{\|\Upsilon^n(\alpha_2)\|} = \frac{\exp[nH(\alpha_1) + \frac{c_1}{n}]}{\exp[nH(\alpha_2) + \frac{c_2}{n}]} \tag{4.104}$$

$$\approx \exp(-nc) \text{ as n increases} \tag{4.105}$$

where we assume $H(\alpha_1) < H(\alpha_2)$ and $c > 0$.

As n increases, the number of elements in set $\Upsilon^n(\alpha_2)$ is exponentially more than than that in set $\Upsilon^n(\alpha_1)$. If α_2 is the frequency distribution that satisfies constraint $\mathcal{C}(\mathbf{f}, \mathbf{u})$ and have the maximum entropy, there will be exponentially more training data sets corresponding to this frequency distribution. In other words, almost all possible sets $(\mathcal{X}, \mathcal{Y})$ have frequencies α_2 which is very close to the maximum entropy frequencies. This phenomenon is also termed "concentration phenomenon". Because almost all possible data sets are "typical" for this frequency distribution, it is reasonable to take it as our "best guess" for the model of our data without any other knowledge.

4.4 Online Cascaded Reranking Model

4.4.1 Online Learning

Most of the early work followed the maximum likelihood principle in training and used the L-BFGS method, a limited-memory quasi-Newton method for conducting large-scale optimization, in parameter estimation for its fast convergence rate.

There is another training approach that is based on the concept of "margin". For example, a perceptron-like algorithm is used in training a hidden Markov model (HMM) [8]. For structured classification, [47] presented a margin-based general framework known as maximum margin Markov networks. The parameter estimation problem is solved in an approach similar to that in an SVM, where a dual problem is formulated and optimization methods analogous to the sequential minimal optimization (SMO) for SVMs are used. A formal SVM treatment for structured classification is also presented in [50].

A margin-based approach to parsing is presented in [48]. It uses a common approach that is used in SVMs for solving the dual problem in training. There is another line of research that focuses on online training methods. For example, [33] proposed an online margin-based training method for parsing. This type of training method is fast and has the advantage that it does not need to form the dual problem though the performance may be a bit inferior. A detailed description of these algorithms, e.g. Margin Infused Relaxed Algorithm (MIRA), and online passive-aggressive algorithms, can be found in [10, 9].

We wish to point out that in this thesis we treat the margin-based approach as a method for estimating parameters. Actually, we can view the parameters estimated by the margin-based principle as "the parameters that can give a distribution with margin-based characteristics over the training set". As mentioned in [41], the best objective value (i.e. the log-likelihood) in maximum likelihood training does not necessarily give the best learning results. Therefore, we can treat the margin-based approach as an alternative method for training.

A similar analysis exists in the simple classification case for SVMs that draws the connection between log-likelihood and margin loss in standard textbooks. Our work is different in that we focus on the context of CRFs and how we make use of this connection to *improve* the performance of existing methods with a length-adaptive margin.

4.4.2 Updating Parameters

We propose to estimate the parameter $\mathbf{w}$ in an online manner. In this OCR model, parameters are estimated by margin-based training, which chooses the set of parameters that attempts to make the "margin" on each training instance greater than a predefined value. In the online training setting, the parameter $\mathbf{w}$ is updated iteratively. In each iteration, we draw a training instance from the training set and update the parameter $\mathbf{w}$. In this thesis, we consider the online passive-aggressive (PA) algorithm [9]. In the t-th iteration with the parameter $\mathbf{w_t}$ and the training instance $\mathbf{x_t}$, this algorithm tries to solve the following problem:

$$\mathbf{w_{t+1}} = \arg\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w} - \mathbf{w_t}\|^2 \tag{4.106}$$

$$\text{such that} \quad \ell(\mathbf{w}; (\mathbf{x_t}, \mathbf{y_t})) = 0$$

where $\ell(\mathbf{w}; \mathbf{x_t})$ is a *hinge loss* function defined as follows:

$$\ell(\mathbf{w}; \mathbf{x_t}) = \begin{cases} 0 & \text{if } \gamma_t \geq \gamma \\ \gamma - \gamma_t & \text{otherwise} \end{cases} \tag{4.107}$$

When the training data is not *linearly-separable*, we need to introduce a slack term and a user-defined aggressive parameter $C > 0$ in Equation (4.106). C controls the penalty of the slack term and the *aggressiveness* of each update step. A larger C implies a more aggressive update and hence a higher tendency to overfit.

In other words, this user-defined aggressive parameter C also accounts for the case where the training data is not *linearly-separable.* Finally the parameter $\mathbf{w_t}$ is updated by:

$$\mathbf{w_{t+1}} = \mathbf{w_t} - \tau_t[\mathbf{F}(\mathbf{x_t}, \mathbf{y_t}) - \mathbf{F}(\mathbf{x_t}, \hat{\mathbf{y}}_\mathbf{t})] \tag{4.108}$$

$$\text{where} \quad \tau_t = \min\left\{C, \frac{\ell(\mathbf{w_t}; (\mathbf{x_t}, \mathbf{y_t}))}{\|\mathbf{F}(\mathbf{x_t}, \mathbf{y_t}) - \mathbf{F}(\mathbf{x_t}, \hat{\mathbf{y}}_\mathbf{t})\|^2}\right\} \tag{4.109}$$

This algorithm is "passive" in a sense that the parameter $\mathbf{w_t}$ is not updated when the margin γ_t is larger than γ. The performance is justified by its relative loss bound on the training data that also bounds the number of prediction mistakes on the training data. The main idea is that the performance cannot be much worse than the best fixed parameter chosen in hindsight. See [9] for a detailed proof.

In this method, it is important to choose an appropriate value for γ. Following most of the work in margin-based training, we choose it to be a function of the correct output sequence $\mathbf{y}$ and the predicted output sequence $\hat{\mathbf{y}}$.

$$\gamma(\mathbf{y}, \hat{\mathbf{y}}) = \begin{cases} 0 & \text{if } \mathbf{y} = \hat{\mathbf{y}} \\ \sum_{i=1}^{|\mathbf{y}|}[[y_i \neq \hat{y}_i]] & \text{otherwise} \end{cases} \tag{4.110}$$

where $[[z]]$ is 1 if z is true, 0 otherwise.

The major computation difficulty in this online training comes from finding the best output $\hat{\mathbf{y}}$. It is in general an intractable task. We follow the usual first-order independence assumption made in a linear-chained CRF model and calculate the best score using the Viterbi algorithm.

This approach has two major characteristics:

- It does not take the average over all parameters in the end as in the averaged perception algorithm.

- Shrinking technique that is commonly used in SVMs can be easily added to reduce training time.

Because of the online setting and the simple update formula, the training time can be substantially reduced. In our experiments, the training time of the online algorithm is generally at least three times faster than that of the L-BFGS algorithm.

4.4.3 Inference: Viterbi Algorithm

We apply the parameter $\mathbf{\Lambda}$ that is determined in the training stage to the decision model. In probability context, we get the best output sequence $\mathbf{y}$ for an input sequence $\mathbf{x}$ by maximizing the conditional probability $P_{\mathbf{\Lambda}}^{D}(\mathbf{y}|\mathbf{x})$. In general it is an intractable task to enumerate all possible $\mathbf{y}$. We make use of the first-order independence assumption and get the output in a dynamic programming fashion. The decision procedure is a Viterbi decoding algorithm. Note that because the exponential function is a monotonic increasing function, there is no need to take the exponential function when searching for the maximum probability. Instead, comparing the indexes is enough (i.e. $e^x > e^y$ iff $x > y$). This can save decoding time and allow post-processing to be incorporated into the decision stage if needed. The algorithm is outlined in Algorithm 3.

Algorithm 3 Viterbi decoding

```
1: $BestS[p][q] = -\infty \quad \forall p, q$
2: $BestPrevLabel[p][q] = -\emptyset \quad \forall p, q$
3: for $p = 1$ to $|\mathbf{x}|$ do
4:     for $q \in \Omega$ do
5:         $MaxS = -\infty$
6:         $MaxPrevLabel = \emptyset$
7:         for $q' \in \Omega$ do
8:             $S = BestS[p-1][q'] + \mathbf{\Lambda} \cdot \mathbf{f}(\mathbf{x_i}, q', q, p)$
9:             if $S > MaxS$ then
10:                $MaxS = S$
11:                $MaxPrevLabel = q'$
12:            end if
13:        end for
14:        $BestPrevLabel[p][q] = MaxPrevLabel$
15:        $BestS[p][q] = MaxS$
16:    end for
17: end for
18: $Label[|\mathbf{x}|] = \arg\max_q BestS[|\mathbf{x}|][q]$
19: for $p = |\mathbf{x}|$ to 2 do
20:     $Label[p-1] = BestPrevLabel[p][Label[p]]$
21: end for
22: return $Label$
```

4.4.4 Reranking

To avoid the errors that are propagated in the cascaded framework, we use the classification results to correct the segmentation results. Our derivation is similar to [42]. However, we avoid dealing with the probability directly. Supposing the probability can be expressed in a maximum entropy model, we can decompose the conditional joint probability into two components:

$$\begin{aligned}
(\hat{\mathbf{y}}_{\mathbf{c}}, \hat{\mathbf{y}}_{\mathbf{s}}) &= \arg\max_{(\mathbf{y}'_{\mathbf{c}}, \mathbf{y}'_{\mathbf{s}})} P(\mathbf{y}'_{\mathbf{c}}, \mathbf{y}'_{\mathbf{s}} | \mathbf{x}) && (4.111) \\
&= \arg\max_{(\mathbf{y}'_{\mathbf{c}}, \mathbf{y}'_{\mathbf{s}})} P(\mathbf{y}'_{\mathbf{c}}, | \mathbf{y}'_{\mathbf{s}}, \mathbf{x}) P(\mathbf{y}'_{\mathbf{s}} | \mathbf{x}) && (4.112) \\
&= \arg\max_{(\mathbf{y}'_{\mathbf{c}}, \mathbf{y}'_{\mathbf{s}})} \mathbf{w}_{\mathbf{c}} \cdot \mathbf{F}_{\mathbf{c}}(\mathbf{x}, \mathbf{y}'_{\mathbf{s}}, \mathbf{y}'_{\mathbf{c}}) + \mathbf{w}_{\mathbf{s}} \cdot \mathbf{F}_{\mathbf{s}}(\mathbf{x}, \mathbf{y}'_{\mathbf{s}})
\end{aligned}$$

Searching for the best pair $(\hat{\mathbf{y}}_{\mathbf{c}}, \hat{\mathbf{y}}_{\mathbf{s}})$ among all of the possible configurations for a sentence $\mathbf{x}$ is an intractable task. In practice, searching among the top-N segmentations is enough, where N is small, say $N \leq 10$. Usually, the scores for the segmentation candidates near the end of the N-best list are so low that the classification scores cannot change their ranks to the top. The N-best list can be obtained efficiently via an A* search [6]. This greatly reduces the computational cost, but approximates well the performance of joint training and decoding.

Chapter 5

Experiments

5.1 Text Mining Tasks

Text mining datasets are usually large in scale and the features are not fixed nor given. For example, in a noun-phrase chunking task, we observe a sentence $\mathbf{x}$ where each word x_i has a label $y_i \in \{\text{B, I, O}\}$ to indicate that a word is the "beginning", "inside", or "out" of a noun-phrase respectively.

Consider the following sentence:

```
time x(t)    y(t)

0    I        B

1    would    O

2    like     O

3    to       O
```

4	see	O
5	a	B
6	movie	I
7	.	O

If we want to do sequence labeling, e.g. in the form $(x(t), x(t-1)) \Rightarrow y(t)$, we need to add a temporary auxiliary attribute $x(t-1)$.

time	x(t)	y(t)	x(t-1)
0	I	B	START
1	would	O	I
2	like	O	would
3	to	O	like
4	see	O	to
5	a	B	see
6	movie	I	a
7	.	O	movie

where START denotes the start of a sequence.

For example, if $(x(t) = \texttt{movie}, x(t-1) = \texttt{a}) \Rightarrow y(t) = \texttt{I}$ is a useful template rule to our classifier, then it will be transformed into a feature:

$$f(\mathbf{x}, \mathbf{y}, t) = \begin{cases} 1 & \text{if } x(t) \text{ is the word ``movie''}, \\ & \text{and } x(t-1) \text{ is the word ``a''}, \\ & \text{and } y(t) \text{ is the label ``I''} \\ 0 & \text{otherwise} \end{cases}$$

A shorthand notation for the above feature is $(x_t = \texttt{movie}, x_{t-1} = \texttt{a}, y_t = \texttt{I})$. Here the presence of a feature is indicated by the feature value of 1, and 0 otherwise. That is, it is a binary feature.

Our experimental results are evaluated by the F_1-measure defined as

$$F_1 = \frac{2PR}{P+R} \tag{5.1}$$

where P is the precision and R is the recall. This exact-match scoring method doubly penalizes incorrect boundaries for an output as false negatives and false positives. To allow comparisons with the results in previous shared tasks, we use the same evaluation script from each of the shared task, which reports on the precision, recall, and the F_1-measure on the evaluation data.

In this thesis, we apply our MEMB, ICRF, and OCR models on the following text mining tasks.

5.2 Biomedical Named Entity Recognition Task

In this biomedical named entity recognition task, we are required to recognize five types of biomedical named entities: DNA, RNA, cell line, cell type and protein.

5.2.1 Dataset Description: BioNLP-2004

Types	Training	%	Evaluation	%
DNA	9,534	18.58	1,056	12.19
RNA	951	1.85	118	1.36
cell_line	3,830	7.47	500	5.77
cell_type	6,718	13.10	1,921	22.18
protein	30,269	59.00	5,067	58.50
Total	51,302	100.00	8,662	100.00

Table 5.1: Number of different biomedical entities in the corpus for the Coling 2004 shared task (JNLPBA).

In this experiment, we use the GENIA corpus from the COLING workshop (BioNLP/NLPBA 2004)[1]. The GENIA corpus consists of 2000 MEDLINE abstracts of the GENIA version 3 corpus with named entities in IOB2 format. There are 18,546 sentences (492,551 words) in the training set and

[1]`http://research.nii.ac.jp/~collier/workshops/JNLPBA04st.htm`

3,856 sentences (101,039 words) in the test set. In this real dataset, the distribution of entity types are not even. The performance of the less evenly entity types is less accurate in general.

Each word is tagged in a way as described in the previous section, except that the tags are for biomedical named entity classes instead of phrase chunks. The named entity classes are: DNA, RNA, cell line, cell type and protein. There is no POS information provided in this corpus. Our task is to extract all the required named entities in the test set. The evaluation is done by the evaluation script provided by the COLING workshop (BioNLP/NLPBA 2004) shared task, which reports on the precision, recall, and the F_1-measure on the test data.

5.2.2 Experimental Setup for BioNLP-2004

Experimental Setup: MEMB for BioNLP-2004

We try to reproduce the features described in [39] and use them as our domain knowledge attributes. The POS attributes are added by the GENIA tagger[2]. Two lexicons for cell lines and genes are drawn from two online public databases: Cell Line Database[3] and BBID[4].

[2]`http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/tagger/`

[3]`http://www.biotech.ist.unige.it/cldb/cname-tz.html`

[4]`http://bbid.grc.nia.nih.gov/bbidgene.html`

To help our discussion, we define "atomic features" as those features that are based on the input sentence $\mathbf{x}$ only. By the first-order independence assumption, the final set of features is a combination of those atomic features with at most two class labels $y_{i-1}, y_i \in \Omega$ where y_{i-1} is the previous label for the i-th word, and y_i is the label for the current word.

Unigram	(w_{-2}), (w_{-1}), (w_0), (w_1), (w_2)
Bigram	$(w_{-2}\ w_{-1})$, $(w_{-1}\ w_0)$, $(w_0\ w_1)$, $(w_1\ w_2)$
Trigram	$(w_{-2}\ w_{-1}\ w_0)$, $(w_{-1}\ w_0\ w_1)$, $(w_0\ w_1\ w_2)$

Table 5.2: Word features used in the experiment (JNLPBA): w_0 is the current word, w_{-1} is the previous word, etc.

Our features are commonly used in text mining tasks. A simple feature selection was used where features appeared less than 3 times in the training corpus were removed.

SEG Module Features

The atomic features used in the SEG module include word features, orthographic features, part-of-speech (POS), and two lexicons. The word features include unigram, bigram, and trigram (e.g. the previous word, the next word, and the previous two words), while the orthographic features [39] include capital letter, dash, punctuation, and word length. *Word class* (WC) features similar to [7] are also added, which replace a capital letter with "A", a lower case letter with "a", a digit with "0", and all other characters with "_". Similar *brief word class* (BWC) features are added by collapsing all consecutive identical characters in *word class* features into one character. For example, for the word `NF-kappa`, $WC =$ `AA_aaaaa`, and $BWC =$ `A_a`. These are listed in Tables 5.2 and 5.3. The POS features are added by the GENIA tagger[5]. All these features except the prefix/suffix features are applied to the neighborhood window $[i-1, i+1]$ for every word. Two lexicons for cell lines and genes are drawn from two online public databases: Cell Line Database[6] and BBID[7]. The prefix/suffix and the lexicon features are applied to position i only. All the above features are combined with the previous label $y_{i-1} \in \Omega$, and the previous and current labels $(y_{i-1}, y_i) \in \Omega \times \Omega$ to form the final set of features $\mathbf{F_s}(\mathbf{x}, \mathbf{y})$. Because only three labels (i.e. `B, I, O`) are needed, the total number of features is $O(3^2 N_a)$ or $O(N_a)$ where N_a is the number of atomic features.

[5]`http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/tagger/`

[6]`http://www.biotech.ist.unige.it/cldb/cname-tz.html`

[7]`http://bbid.grc.nia.nih.gov/bbidgene.html`

We remove features that occur fewer than five times in the SEG module to reduce the computational cost. The total number of features is about 2.3 million.

Word features	as in Table 5.2
Prefix/suffix	Up to a length of 5
Word Class	WC
Brief Word Class	BWC

Capital Letter	`^[A-Z][a-z]`
	`[A-Z]{2,}`
	`[a-z]+[A-Z]+`

Digit	`[0-9]+`
	`^[^0-9]*[0-9][^0-9]*$`
	`^[^0-9]*[0-9][0-9][^0-9]*$`
	`^[0-9]+$`
	`[0-9]+[,.][0-9,.]+`
	`[A-Za-z]+[0-9]+`
	`[0-9]+[A-Za-z]+`

Dash	`[-]+`
	`^[-]+`
	`[-]+$`

Punctuation	`[,;:?!-+'"\/]+`
Word length	length of the current word x_i

Table 5.3: Features used in the JNLPBA experiment. The features for *Capital Letter, Digit, Dash,* and *Punctuation* are represented as regular expressions.

CLASS Module Features

In the CLASS module, the model only needs to determine the correct entity type for a given named entity candidate. The output sequence consists of one element only, i.e. $y \in \{\texttt{protein, cell\_type, cell\_line, DNA, RNA}\}$. The atomic features mentioned are combined with each possible label y to form the final set of features $\mathbf{F_c}(\mathbf{x}, y)$. These features are applied to each named entity candidate $\mathbf{x}$ from the SEG module. The total number of features is $O(N_c N_a)$ where N_c is the number of entity types. The total number of possible features in the CLASS module is less than that in the SEG module. We can afford to remove fewer features. Features that occur fewer than three times are removed, with a resulting total of about one million features.

Experimental Setup: ICRF for BioNLP-2004

In this experiment, we treat Y as the segmentation tags and Z as the named-entity tags. In other words, the CRF model $\mathrm{Q}_\phi(Y|X)$ predicts segmentation tags Y based on the input sentence X and $\mathrm{P}_\theta(Z|X,Y)$ predicts the named-entity tags Z based on the input sentence X and the predicted segmentation tags Y.

In the first task, we need to identify all segments that are named entity candidates. We pre-process the original training data and replace all named entity types with type ENT (i.e. Entity). In the second task, we needed to categorize each segment into 5 different entity types: DNA, RNA, cell_line, cell_type, protein. We also used a richer set of features to capture the characteristics of the relationships between segments and entity types because there were more different segments and named entity types. Because our isomeric CRF is flexible in choosing different feature templates in different stages, we experimented with different feature templates. The features we used in this experiment are presented in Table 5.2 and Table 5.3.

Experimental Setup: OCR for BioNLP-2004

The setup for this experiment is very similar to that in MEMB. The features used are shown in Table 5.2 and 5.3. We decompose an entity tag into a segmentation tag and a classification tag. For example, "B-DNA" is decomposed into "B-ENT" in the segmentation task and "DNA" in the classification task.

Because we cannot enumerate the results for all N, we choose N to be 10 and performed online training to generate our models in the two tasks for this BioNLP-2004 dataset. After that we use re-ranking to choose the best candidate which has the best overall score.

Our focus in this set of experiments is the use of online training method to increase speed and reranking method to increase accuracy. The programs were developed based on the same basic framework. All of the experiments were run on a Unix machine with a 2.8 GHz CPU and 16 GB RAM.

5.2.3 Experimental Results and Discussions: BioNLP-2004

Types	Train. Data %	Eval. Data %	CRF+CRF F_1	MEMB F_1	ICRF F_1	OCR F_1
DNA	18.58	12.19	66.13	70.15	67.61	69.08
RNA	1.85	1.36	67.48	69.08	69.92	69.88
cell_line	7.47	5.77	56.33	60.15	58.14	59.04
cell_type	13.10	22.18	72.90	75.06	73.43	75.30
protein	59.00	58.50	73.08	74.13	73.84	73.65
overall	100	100	71.09	72.94	72.00	72.54

Table 5.4: Comparisons (in %) of our models with the baseline CRF+CRF model on BioNLP experiments. The MEMB model has the best performance in extracting biomedical named entities.

Experimental Results and Discussions: MEMB for BioNLP-2004

In the SEG module, our approach achieves an F_1 of 77.56%. The recall is of particular importance in the SEG module because it directly limits the number of correct answers that our approach can give. The main results of our experiments are summarized in Tables 5.5, 5.6, and 5.9.

When we compare the SEG module recall (80.45%) with the results re-

Experiments	Recall	Precision	F_1
DNA	82.95	96.16	89.07
RNA	87.29	93.64	90.35
cell_line	77.80	96.53	86.16
cell_type	78.81	98.50	87.57
protein	83.96	96.27	89.69
overall	80.45	74.86	77.56

Table 5.5: Results (in %) of the SEG module

Experiments	Recall	Precision	F_1
DNA	85.04	91.63	88.21
RNA	81.36	91.43	86.10
cell_line	79.20	73.88	76.45
cell_type	87.61	95.19	91.24
protein	97.89	94.06	95.94
overall	92.77	92.77	92.77

Table 5.6: Results (in %) of the CLASS module on fully correct segmentation data

ported in the JNLPBA shared task in Table 5.7, it is clear that subsequent good classification results will yield a good overall F_1. We also compare the segmentation results with a CRF that uses the same set of features in Table 5.8. Generally, the MEMB model shows a greater relative loss reduction on F_1 for sparse entity types such as RNA. Because the fraction of different entity types differs in the training data and the evaluation data, some discrepancies are also found in DNA and cell_type.

System	Recall	Precision	F_1
Cascaded MEMB	75.66	70.40	72.94
Zhou and Su, 2004	75.99	69.42	72.55
OCR	73.83	71.30	72.54
ICRF	74.39	69.76	72.00
Okanohara et al., 2006	72.65	70.35	71.48
Kim et al., 2005	72.77	69.68	71.19
Finkel et al., 2004	68.56	71.62	70.06
Settles, 2004	70.30	69.30	69.80

Table 5.7: Comparisons (in %) with other systems on the overall performance. All other systems except (Okanohara et al., 2006) employ different deep knowledge resources.

To test the performance of the CLASS module, we prepared a set of testing data with all the entities correctly segmented and used it to evaluate the module. The overall accuracy is 92.77%, which is higher than the reported classification results (90.54%) by a simple maximum entropy model in [21].

Types	Train. Data %	Eval. Data %	CRF+CRF F_1	MEMB F_1	rel. loss red. on F_1
DNA	18.58	12.19	88.32	89.07	6.42
RNA	1.85	1.36	88.89	90.35	13.14
cell_line	7.47	5.77	85.94	86.16	1.56
cell_type	13.10	22.18	87.21	87.57	2.81
protein	59.00	58.50	89.59	89.69	0.96

Table 5.8: Comparisons (in %) with CRF model on the segmentation performance. Generally, the MEMB model shows a higher relative loss reduction on F_1 for sparse named entities. Small discrepancies are found when there is a large difference between the fraction of the named entities in the training data and that in the evaluation data , for instance, DNA and cell_type.

The detailed results are depicted in Table 5.6. With the performance in the two modules now available, we can get a rough estimate of the final extraction performance as shown in Table 5.9. Note that if the CLASS module accuracy is 100%, the final extraction performance is identical to the segmentation performance, i.e. $F_1 = 77.56\%$.

			MEMB			ICRF			OCR
Exp.	R	P	F_1	R	P	F_1	R	P	F_1
Segment.	80.45	74.86	77.56	N/A	N/A	N/A	80.13	73.68	76.77
Class.	92.77	92.77	92.77	N/A	N/A	N/A	92.76	92.76	92.76
Est. overall	74.63	69.45	71.95	N/A	N/A	N/A	74.32	68.34	71.21
DNA	69.98	70.31	70.15	67.99	67.23	67.61	68.66	69.51	69.08
RNA	72.88	65.65	69.08	72.88	67.19	69.92	73.73	66.41	69.88
cell_line	64.60	56.27	60.15	63.60	53.54	58.14	60.40	57.74	59.04
cell_type	69.34	81.82	75.06	67.05	81.16	73.43	70.07	81.38	75.30
protein	80.40	68.76	74.13	79.61	68.84	73.84	77.66	70.04	73.65
overall	75.66	70.40	72.94	74.39	69.76	72.00	73.83	71.30	72.54

Table 5.9: Overall results (in %) of our models. The first two rows show the performance of the individual module. The SEG module (Segment.) achieves $F_1 = 77.56\%$. The CLASS module (Class.) performance (92.77%) is based on the fully correct segmented testing data. The following rows show the actual extraction performance (segmentation followed by classification) for each type of entity, achieving an overall $F_1 = 72.94\%$.

The results show that our proposed approach outperforms all the systems in the JNLPBA shared task. Their approaches include the Support Vector

Machine (SVM), the Hidden Markov Model (HMM), the Maximum Entropy Markov Model (MEMM) and the Conditional Random Field (CRF), which use deep knowledge resources with extra costs in pre-processing and post-processing. For example, the best system, (Zhou and Su, 2004), used name alias resolution, cascaded entity name resolution, abbreviation resolution and an open dictionary (around 700,000 entries). Their major framework is a SVM framework. Besides a simple set of features, including word and orthographic, they also use a very rich features set that includes semantic and morphological features, part-of-speech, dictionaries, and so forth. In-domain POS information was also used to extract biomedical named entities. (Settles, 2004) used 17 lexicons that include Greek letters, amino acids, and so forth. (Finkel et al., 2004) used gazetteers and web-querying. Although our system outperforms the best system, (Zhou and Su, 2004), we believe it can be further improved, since our cascaded approach does not prohibit the use of these deep knowledge resources.

We also compare our results with a recent work in [35] that uses a semi-CRF based on a feature forest without using deep knowledge resources such as gazetteers or post-processing. Their system performance is only slightly lower than that of (Zhou and Su, 2004) and our system, and also outperforms the other systems. Another recent work by [21] uses a similar cascaded approach as ours by using two different models in the two phases. They use additional information from the GENIA 3.02p version corpus to train their CRF. A simple maximum entropy model is used in classification. Post-processing is needed to correct the final results. Their reported classification performance is 90.54%, which is lower than our approach. Since they did not report the segmentation performance, we can only compare the final results in Table 5.7.

One interesting point to note is that our actual extraction performance is higher than the estimated performance as shown in Table 5.9. We believe the reason for this is that a large number of same features are used in both the SEG and CLASS modules. Therefore the entity candidates from the SEG module are those that are "sensitive" to the given features, and the same is also likely to happen in the CLASS module, giving a better final performance than estimated.

Experimental Results and Discussions: ICRF for BioNLP-2004

As shown in Table 5.9, our ICRF model performance on BioNLP data is slightly worse than that of MEMB model. However, we can see in Table 5.4 that it is significantly higher than that of the baseline model CRF+CRF.

We notice that the performance on entity types that show big differences in distributions in the training and evaluation data tend to be worse. For example, the entity type DNA has about a 6% difference in the distributions of the training data and the evaluation data and its F_1-measure is about

2.54% worse than that of MEMB model. The same also happens for the entity types cell_type and cell_line that show a difference of about 9% and 2% respectively and their performance is about 2% worse than that of the MEMB model.

We also compare our results with other published work in Table 5.7. ICRF model is ranked third among all models. It is better than all other models except the MEMB model and (Zhou and Su, 2004). In the case of (Zhou and Su, 2004), its performance is only 0.55% lower in F_1-measure.

Experimental Results and Discussions: OCR for BioNLP-2004

Table 5.10 shows the F_1-measure in our experiments. [8] Our performance of the single-phase CRF with maximum likelihood training is 69.44, which agrees with [39] who also uses similar settings. The single-phase online method increases the performance to 71.17. By employing a cascaded framework, the performance is further increased to 72.16. Finally, with reranking, some errors are corrected, and the performance reaches 72.54, which can be regarded as on par with the best system in the JNLPBA shared task.

The experimental evidence for using a small value of N in reranking can

[8]We are aware of the high F_1 in [52]. We contacted the author and found a serious bug in their feature template. We reported it to the author in June 2007. Later Professor Chun-Nan Hsu helped report that in NIPS 2009 [18] and Machine Learning Journal 2009 [19].

Experiments		no. of features	training time	F_1	rel. err. red. on F_1
single-phase	CRF + ML	8,004,392	1699 mins	69.44	–
	CRF + Online	8,004,392	116 mins	71.17	5.66%
two-phase	Online + Cascaded	seg: 2,356,590 class: 8,278,794	14 + 15 = 29 mins	72.16	8.90%
	OCR	seg: **2,356,590** class: **8,278,794**	14+15 **= 29 mins**	**72.54**	**10.14%**

Table 5.10: The number of features, training time, and F_1 that are used in our experiments. The cutoff thresholds for the single-phase CRFs are set to 20, whereas that of the OCR approach is set to 5 in both segmentation and classification. The last column shows the relative error reductions on F_1 (compared to CRF+ML).

be seen from Figure 5.1. We try to look for the correct segmentation from the N-best segmentation list. If it exists in the N-best list, then we treat it as a true positive. Our results show that for $N = 11$, the segmentation performance reaches $F_1 = 91.58$. A further increase of N does not gain much in performance.

We also experimented with different values of N in reranking. The results are plotted in Figure 5.2. The best performance occurs at $N = 2$. Larger values of N give more opportunities to correct the segmentation errors, but sometimes additional errors are also introduced. Therefore, the increase in performance fluctuates when N is increased.

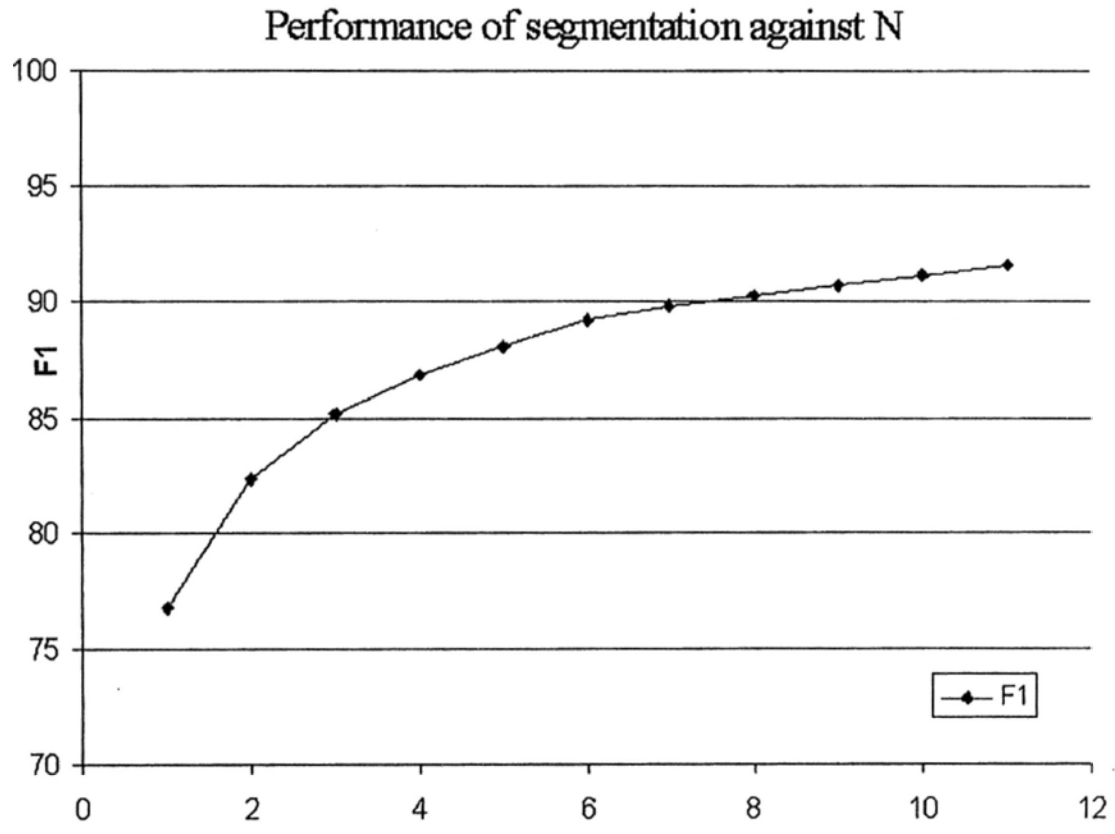

Figure 5.1: Performance of different values of N in segmentation. The F_1-measure is calculated based on the N-best list from segmentation.

Training Time: Referring to Table 5.10, the training time of the OCR approach is substantially shorter than that of all of the other approaches. In the single-phase approach, training a CRF by maximum likelihood (ML)

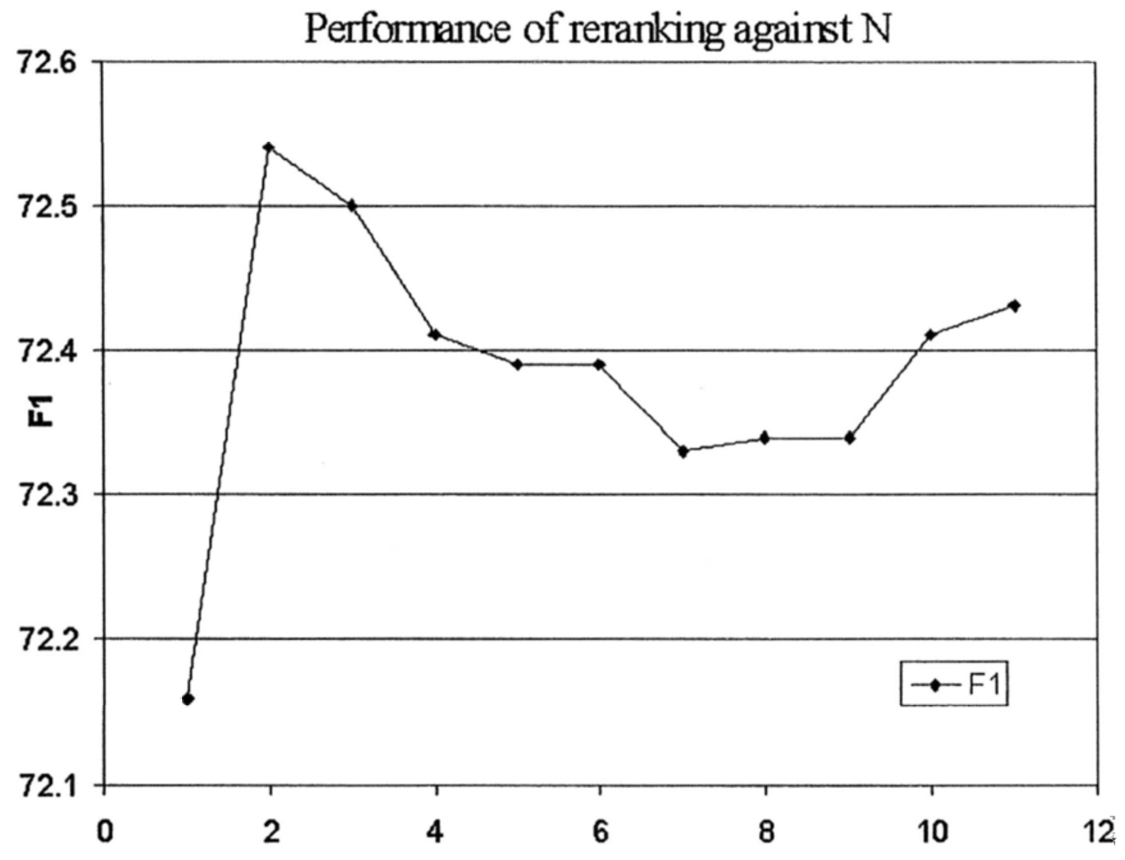

Figure 5.2: Performance of different values of N in reranking.

using the L-BFGS algorithm is the slowest and requires around 28 hours. The online method greatly reduces the training time to around two hours, which is 14 times faster. By employing a two-phase approach, the training time is further reduced to half an hour. In the OCR approach, the reranking time during testing is negligible up to $N = 11$, so we do not report it.

Memory Requirement: Table 5.10 shows the number of features that are required by the different methods. For methods that use the single-phase approach, because the full set of features is too big for practical experiments on our machine, we need to set a higher cutoff value to reduce the number of features. With a cutoff of 20 (i.e. only features that occur more than 20 times are used), the number of features can still go up to about 8 million. However, in the two-phase approach, even with a smaller cutoff of 5, the number of features can still remain at about 8 million.

5.3 General Named Entity Recognition Task

In this general named entity recognition task, we are required to recognize four types of named entities: persons (PER), locations (LOC), organizations (ORG), and names of miscellaneous entities (MISC) that do not belong to the previous three groups.

5.3.1 Dataset Description: CoNLL-2003

In this experiment, we used the dataset in the CoNLL-2003 shared task[9]. This dataset comes from the Reuters Corpus CD[10]. There are 14,987 sentences (204,567 words) in the training set and 3,466 sentences (51,578 words) in the test set (`testa`). The named entity tags are in the form of "B-X", where "X" is a named entity type. For example, the tags include "B-PER",

[9]`http://www.cnts.ua.ac.be/conll2003/ner/`

[10]`http://trec.nist.gov/data/reuters/reuters.html`

"I-MISC", "O", etc. The distribution of each entity types in the training data and the evaluation data is shown in Table 5.11. Part-of-speech (POS) tags and noun-phrase (NP) tags are also provided in this dataset. Each word is labeled with its POS tag and its NP tag. The auxiliary attributes used in the experiments are given in Table 5.12 and 5.13 . The evaluation is done by the evaluation script provided by the CoNLL-2003 shared task that reports on the precision, recall, and the F_1-measure on the test data. It was used in the feature induction experiments in [30].

Types	Training	%	Evaluation	%
LOC	7,141	30.47	1,837	30.91
MISC	3,461	14.77	923	15.53
ORG	6,231	26.59	1341	22.56
PER	6,600	28.17	1,842	30.99
Total	23,433	100.00	5,943	100.00

Table 5.11: Number of different general named entities in the corpus for the CoNLL-2003.

Word features	w_{-2}, w_{-1} w_0 w_1, w_2
Prefix/suffix	from length of 2 to 4

Capital Letter	`^[A-Z][a-z]+`
	`^[A-Z]+`
	`^[A-Z]+[a-z]+[A-Z]+[a-z]`

Digit	`.*[0-9].*`
POS	part-of-speech tags from Brill's POS tagger (provided along with the CoNLL-2003 dataset)

Table 5.12: Features used in the first sub-task of CoNLL-2003 experiment. The first sub-task was to predict the correct phrase tag (including verb phrase, adjective phrase, conjunction phrase, etc) for every word. Therefore more diversified features were used in this sub-task. The features for *Capital Letter, Digit* are presented as regular expressions.

5.3.2 Experimental Setup for CoNLL-2003

Experimental Setup: MEMB for CoNLL-2003

We used different auxiliary features in our experiments to represent the domain knowledge that we need to capture. All features that we used in this

Word features	w_{-2}, w_{-1} w_0 w_1, w_2
Prefix/suffix	from length of 2 to 4
Capital Letter	`^[A-Z][a-z]+`
Digit	`.*[0-9].*`
POS	part-of-speech tags from Brill's POS tagger (provided along with the CoNLL-2003 dataset)
Phrase	phrase tags

Table 5.13: Features used in the second task of CoNLL-2003 experiment. We deliberately tried to use a different feature template for the second sub-task. With the phrase tags given from the previous sub-task and fewer entity types (MISC, LOC, PER, ORG), we reduced our feature templates to allow faster training. The features for *Capital Letter* are presented as regular expressions.

experiment are shown in Table 5.12 and Table 5.13.

All these features serve as our "atomic features" which involve only the input sentence $\mathbf{x}$. They will be combined with the class labels y_{i-1}, y_i to from the final feature set that will be used in our model.

Following our previous description of our MEMB model, we will describe

SEG Module Features and **CLASS Module Features** below respectively. The number of features that should be included actually depends on our computation power.

SEG Module Features

The SEG module features that we used in the experiment include word features, prefix/suffix, capital letter, digit, and part-of-speech(POS). The word features include unigram and bigram (e.g. the previous word and the next word), while the orthographic features include capital letter and digit. The POS features are provided in the CoNLL-2003 data.

All the above features are combined with the previous label $y_{i-1} \in \Omega$, and the previous and current labels $(y_{i-1}, y_i) \in \Omega \times \Omega$ to form the final set of features $\mathbf{F_s}(\mathbf{x}, \mathbf{y})$. Because only nine labels (for the 9 different phrases) are needed, the total number of features is $O(9^2 N_a)$ or $O(N_a)$ where N_a is the number of atomic features.

We remove features that occur fewer than five times in the SEG module to reduce the computational cost. The total number of features is about 1.3 million.

CLASS Module Features

In this module, for a given named entity candidate, we need to determine the correct entity type. Our model needs to choose an entity type out of the 4 possible choices LOC, MISC, ORG, and PER. The output sequence consists of one element only. The atomic features mentioned in the previous section are combined with each possible label y to form the final set of features $\mathbf{F_c}(\mathbf{x}, y)$. These features are applied to each named entity candidate $\mathbf{x}$ from the SEG module. The total number of features is $O(N_c N_a)$ where N_c is the number of entity types. The total number of possible features in the CLASS module is less than that in the SEG module. We can afford to include all features , with a resulting total of about 1.4 million features.

Experimental Setup: ICRF for CoNLL-2003

In this experiment setup, we treat Y as the phrase tags and Z as the named-entity tags. In other words, the CRF model $\mathrm{Q}_\phi(Y|X)$ predicts phrase tags Y based on the input sentence X and $\mathrm{P}_\theta(Z|X, Y)$ predicts the named-entity tags Z based on the input sentence X and the predicted phrase tags Y.

This is a large scale experiment with many sentences in both the training and testing sets. In the first task, we not only needed to identify noun-phrases (NP) but also other phrases such as verb phrases (VP), adjective phrases (ADJP), etc. In the second task, we needed to categorize each phrase into 4 different entity types. We also used a richer set of features to capture the characteristics of the relationships between phrases and entity types because there were more different phrases and named entity types. Because our isomeric CRF is flexible in choosing different feature templates in different stages, we experimented with different feature templates. The features we used in this experiment are presented in Table 5.12 and Table 5.13.

Experimental Setup: OCR for CoNLL-2003

Similar to ICRF, we treat Y as the phrase tags and Z as the named-entity tags in this experiment setup. However the score given by a N-best candidate in the first task will be combined with the score given by the best candidate in the second task to give the overall score. The features used were similar to those used in the ICRF experiment except that in the second task the phrase tag results from the first task would be used as features too.

Because we cannot enumerate the results for all N, we choose N to be 10 and performed online training to generate our models in the two tasks for this CoNLL-2003 dataset. After that we use re-ranking to choose the best candidate which has the best overall score.

Note that in order to perform reranking, we need to assign scores to candidate answers in the first and second tasks. In other words, we get a score for the phrase tag assignments for the whole sentence in the first task and also a score for the named-entity tag assignments for the whole sentence in the second task. As such, the overall score is also on a sentence level.

5.3.3 Experimental Results and Discussions: CoNLL-2003

Experimental Results and Discussions: MEMB for CoNLL-2003

Experiments	Recall	Precision	MEMB F_1	ICRF F_1	OCR F_1
Segment.	93.84	94.71	94.28	N/A	93.41
Class.	89.86	89.86	89.86	N/A	87.87
Est. overall	84.32	85.11	84.72	N/A	82.07
LOC	89.38	92.09	90.72	88.90	91.17
MISC	85.16	90.45	87.72	86.31	85.70
ORG	82.10	86.69	84.34	81.59	82.67
PER	92.02	89.73	90.86	90.01	90.27
overall	87.90	89.90	88.89	87.23	88.17

Table 5.14: Overall results (in %) of our model. The first two rows show the performance of the individual module. The SEG (Segment.) module achieves $F_1 = 94.28\%$. The CLASS module (Class.) performance (89.86%) is based on the fully correct phrase testing data. The following rows show the actual extraction performance (phrase extraction followed by classification) for each type of entity, achieving an overall $F_1 = 88.89\%$.

The main results of the experiments are summarized in Table 5.14.

In the SEG module, our approach achieves an F_1 of 94.28%. Generally speaking, if the SEG module achieves a good result, the overall performance will then depend on the performance of the CLASS module.

To test the performance of the CLASS module, we prepared a set of testing data with all the phrases correctly identified and used it to evaluate the module. The overall accuracy is 89.86%. With the performance in the two modules now available, we can get a rough estimate of the final extraction performance as shown in Table 5.14. Note that if the CLASS module accuracy is 100%, the final extraction performance should be very close to the SEG module performance, i.e. $F_1 = 94.28\%$.

We also note that our actual extraction performance is higher than the estimated performance. We believe the reason for this is that in the SEG module the phrases (instead of the actual segmentation) are being extracted. When a phrase cannot be recognized by the SEG module, it is also likely that it cannot be recognized by the CLASS module either. Therefore the performance is not hurt substantially.

Experimental Results and Discussions: ICRF for CoNLL-2003

The results of this experiment is presented in Table 5.15 and Table 5.16. The phrase tag accuracy were on par with a difference of only 0.02%. The General Entity Recognition tags is slightly better in the isomeric CRF ap-

Types	CRF+CRF	MEMB	Diff	isom. CRF	Diff	OCR	Diff
P Acc	96.38	96.74	+0.36	96.40	+0.02	96.34	-0.04
GER Acc	97.39	97.86	+0.47	97.57	+0.18	97.78	+0.39
LOC	88.09	90.72	+2.63	88.90	+0.81	91.17	+3.08
MISC	85.04	87.72	+2.68	86.31	+1.27	85.70	+0.66
ORG	79.80	84.34	+4.54	81.59	+1.79	82.67	+2.87
PER	89.71	90.86	+1.15	90.01	+0.30	90.27	+0.56
Overall	86.27	88.89	+2.62	87.23	+0.96	88.17	+1.90

Table 5.15: Comparisons with the traditional *CRF+CRF* approach on the CoNLL-2003 dataset. The top 2 rows indicate the phrase (P) tag accuracy and the General Entity Recognition (GER) tag accuracy in percentage (%). The next four rows show the F_1 measures for different entity types: Location (LOC), Miscellaneous (MISC), Organization (ORG), and Person (PER). The last row gives the overall GER F_1 measure and repreents the final rank of a system.

proach, with an accuracy of 97.57% which is 0.18% better than the cascaded approach. In this case we would expect that the F_1 measures would be nearly the same. However the overall F_1 measure is about 1% higher in the isomeric CRF model. Furthermore, if we look at the F_1 measures of the individual named entity types, we find a greater difference between the CRF and ICRF

Types	CRF+CRF	MEMB	Diff	isom. CRF	Diff	OCR	Diff
P Acc	96.38	96.74	+0.36	96.40	+0.02	96.34	-0.04
ADJP	65.66	68.92	+3.26	66.29	+0.63	64.15	-1.51
ADVP	82.68	84.76	+2.08	82.38	-0.30	81.30	-1.38
CONJP	30.77	66.67	+35.90	42.86	+12.09	40.00	+9.23
INTJ	85.19	93.10	+7.91	89.29	+4.10	86.21	+1.02
NP	92.32	93.79	+1.47	92.27	+0.45	92.63	+0.32
PP	98.35	98.11	-0.24	98.18	-0.17	98.01	-0.34
PRT	94.28	92.36	-1.92	94.28	0.00	93.60	-0.68
SBAR	87.79	87.32	-0.47	84.96	-2.83	86.26	-1.53
VP	95.08	95.33	+0.25	95.23	+0.15	95.18	+0.10
Overall	93.36	94.28	+0.92	93.30	-0.06	93.41	+0.05

Table 5.16: Comparisons with the traditional *CRF+CRF* approach on the phrase extraction performance. The top row indicates the phrase (P) tag accuracy in percentage (%). The next 9 rows show the F_1 measures for different entity types. The last row gives the overall extraction performance on the first sub-task. Note that in the isomeric CRF approach the overall phrase extraction performance is lower than the serparately trained model. However the final system performance of the isomeric CRF approach is better.

C

approaches. For the named entity types "MISC" and "ORG", both F_1 measures were more than 1% higher in the isomeric CRF approach. It shows that the two sub-tasks in the isomeric CRF approach are better "integrated" so that the overall system performance is improved.

With reference to Table 5.15, a greater difference can be seen in the first task which is the phrase recognition task. We see that the phrase tag accuracy was about the same with a difference of only 0.02%. However when we break down the individual F_1 measures, we can see that the F_1 measure for "CONJP" and "INTJ" were exceptionally higher in the isomeric CRF approach, with a +12.09% and a +4.10% increase respectively. We conjecture that these two phrase types are helpful in doing general named entity recognition tasks.

McCallum conducted an experiment with manually-prepared templates with lexicons and also one with single word features. Surprisingly, the manually-prepared templates with lexicons got the lowest performance 73.3% F1 while in the experiment using simple single word features, the performance was 80% F1. Our ICRF gets 87.23% F1, which are better than that of the experiment using manually-prepared templates or the single word features. However our performance is worse than the feature induction method that used a lexicon for the named entities. We think that the difference in performance may be due to the use of lexicons that provides additional domain knowledge.

Our training time using a single-threaded approach was 106 minutes and that using a mutli-threaded approach with 16 threads was 9 minutes.

Experimental Results and Discussions: OCR for CoNLL-2003

In this experiment, we get $F_1 = 88.17\%$. In the segmentation task, the phrase tag accuracy is 96.34%, which is slightly lower than the baseline (*CRF+CRF*) approach. In the classification task, the GER tag accuracy is 97.78%, which is better than the baseline approach.

When we analyzed the segmentation results, we found that the segmentation result for CONJP was 9.23% higher than that of the baseline results, which agreed with the results of MEMB and ICRF. However the segmentation result for INTJ does not show a sharp improvement as other approaches do.

We noted that the re-ranking stage of OCR did not help improve the segmentation results as we expected. After we analyze the 10-best segmentation results, we found that many of the correct segmentation occurred at the 1-best or 7-best to 10-best segmentation candidates. We believe that the re-ranking stage could not correct segmentation results in which the correct segmentation occurs near the end of a N-best list.

The training time for the segmentation task was 93s and that for the classification task was 381s. In total it was 474s. It achieved the fastest training time among all models.

5.4 Other Experiments

5.4.1 Parallel Training Experiments

We also tested the speed up gained by the parallel implementation of the isomeric CRF approach. The parallel framework can be implemented on a multi-core CPU which supports multi-threading or a GPU graphics card. The speed test was done on the CoNLL-2000 dataset with different number of training instances. The same set of feature templates used in the previous experiments were used. The test was done with a single-threaded implementation and a multi-threaded implementation with 16 threads. The experiments were run on a machine with 2x Xeon X5570 2.93GHz and 48G Ram. The results are shown in Figure 5.3.

Generally the training time grew linearly with the number of training instances. The average speedup was 12.67x over this dataset. The termination condition was that the objective did not improve more than 0.5% for three times. We noted that there were cases that the training time was shorter when the training size was bigger. However the speedup stayed constantly between 12x and 13x.

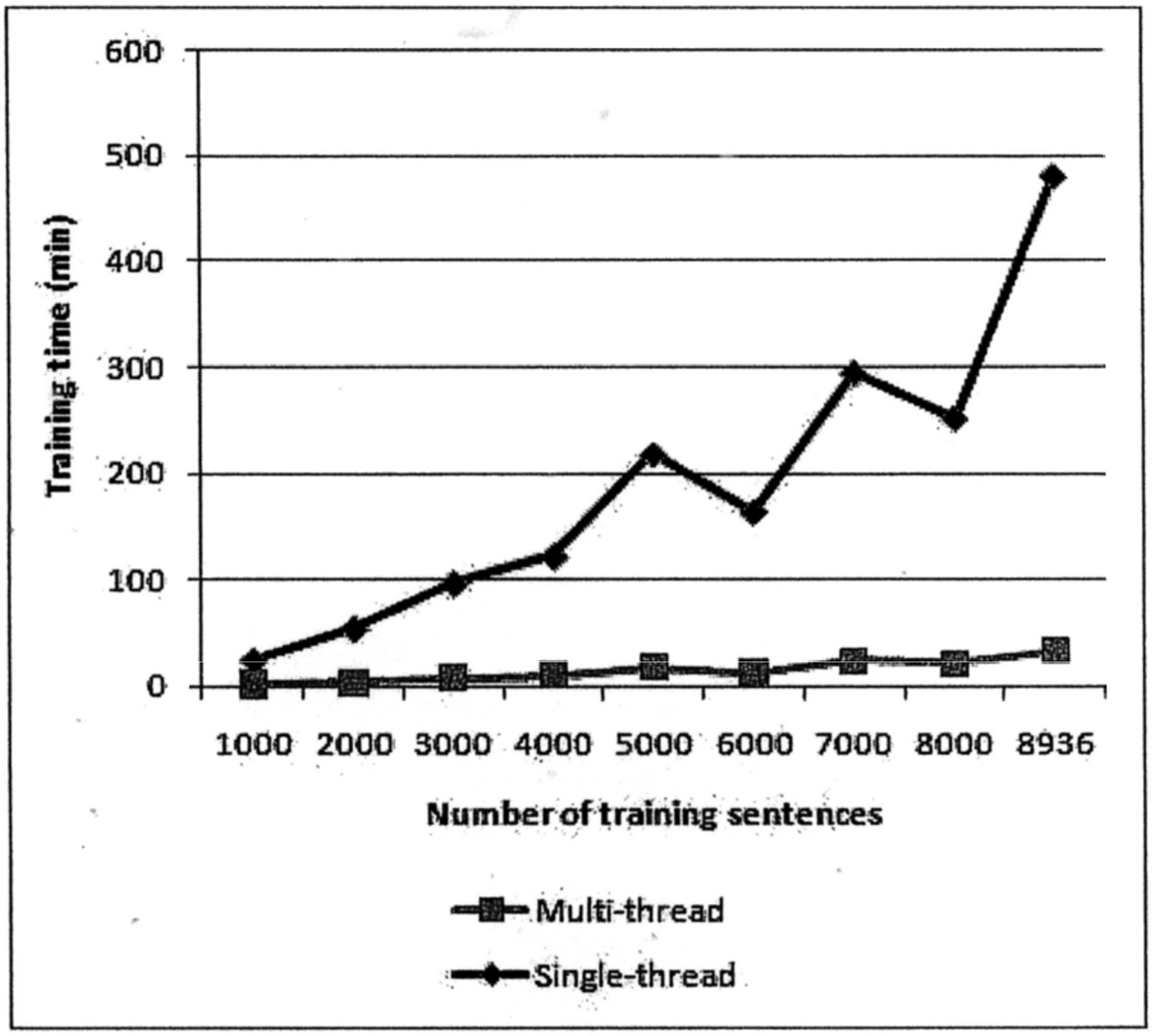

Figure 5.3: Speed up of isomeric CRF using a parallel implementation. Using the same feature templates in the CoNLL-2000 experiment, we repeated the experiment with different number of training instances, from 1000 to 8936. The multi-threaded approach used a total of 16 threads. On average, the speedup is 12.67x.

Chapter 6

Conclusions

We have presented a cascaded framework for extracting named entities with automatically generated features. Three named entity recognition (NER) models based on different principles are investigated. The maximum-entropy margin-based (MEMB) model is designed to make use of the concept of margin in a maximum entropy model. It measures the error between the original tags and the predicted tags of a named entity and incorporates it in training. The isomeric conditional random field (ICRF) model is designed to allow efficient joint training with a soft constraint. The segmentation and classification models communicate with each other through different representations of a joint probability. The online cascaded reranking (OCR) model uses online training to conduct parameter estimation so that its margin is maximized. The segmentation ranks its top-N candidates with scores and passes this N-best list to the classification which gives a score to each candidate and chooses the candidate with the highest total score to be the final output. A theoretical analysis of the constraints on the probability distributions in the MEMB and ICRF models is given. By using Stirling's approximation, we prove that almost all possible data sets are typical for an empirical frequency distribution.

We have evaluated the three models on two publicly-available datasets, namely the GENIA Corpus in the BioNLP/NLPBA (2004) shared task, and the Reuters Corpus in the CoNLL-2003 shared task. Experimental results show that our proposed models are effective. Our models outperform the baseline model and achieve state-of-the-art performance. A parallel implementation of the ICRF model is also evaluated and achieves a 10x speedup.

A future direction is to investigate how a "margin" can be used efficiently in a joint training fashion. Margin-based training usually gives robust parameter values and joint training allows different stages of a system interact with each other. Another direction is to explore parallel algorithms that can help training and testing. With the advent of multi-core CPUs and GPUs, researchers can be less worried about the running speed and focus more on the characteristic of a problem and its proper modeling. Furthermore, we expect that we will be able to perform extensive experiments on other text mining tasks such as semantic labeling that usually has more than 10 labels and need to employ more language resources which requires more feature functions to be defined either manually or automatically by a template.

Bibliography

[1] Avrim L. Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artif. Intell.*, 97(1-2):245–271, 1997.

[2] Shing-Kit Chan and Wai Lam. Efficient methods for biomedical named entity recognition. In *Proceedings of the 7th IEEE International Conference on Bioinformatics and Bioengineering (BIBE)*, pages 729–735, 2007.

[3] Shing-Kit Chan and Wai Lam. Pseudo conditional random fields: Joint training approach to segmenting and labeling sequence data. *Data Mining, IEEE International Conference on*, 0:767–772, 2010.

[4] Shing-Kit Chan, Wai Lam, and Xiaofeng Yu. A cascaded approach to biomedical named entity recognition using a unified model. *Data Mining, IEEE International Conference on*, 0:93–102, 2007.

[5] Shing-Kit Chan, Wai Lam, and Xiaofeng Yu. An online cascaded approach to biomedical named entity recognition. In *Proceedings of the Third International Joint Conference on Natural Language Processing (IJCNLP)*, 2008.

[6] Yen-Lu Chow and Richard Schwartz. The n-best algorithm: an efficient procedure for finding top n sentence hypotheses. In *HLT '89: Proceed-*

ings of the workshop on Speech and Natural Language, pages 199–202, 1989.

[7] Michael Collins. Ranking algorithms for named-entity extraction: boosting and the voted perceptron. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 489–496, 2001.

[8] Michael Collins. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, pages 1–8, 2002.

[9] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.

[10] Koby Crammer and Yoram Singer. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991, 2003.

[11] Nello Cristianini. Support vector and kernel machines. ICML tutorial, 2001. Available at http://www.support-vector.net/icml-tutorial.pdf.

[12] Aron Culotta, Michael Wick, Robert Hall, and Andrew Mccallum. First-order probabilistic models for coreference resolution. In *In Proceedings of HLT-NAACL 2007*, 2007.

[13] Gal Elidan. Residual belief propagation: Informed scheduling for asynchronous message passing. In *in Proceedings of the Twenty-second Conference on Uncertainty in AI (UAI*, 2006.

[14] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

[15] Jenny Rose Finkel, Christopher D. Manning, and Andrew Y. Ng. Solving the problem of cascading errors: approximate bayesian inference for linguistic annotation pipelines. In *EMNLP '06: Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 618–626, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

[16] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, April 1970.

[17] Kristy Hollingshead and Brian Roark. Pipeline iteration. In *ACL*, 2007.

[18] Chun-Nan Hsu, Yu-Ming Chang, Hanshen Huang, and Yuh-Jye Lee. Periodic step size adaptation for single pass on-line learning. pages 763–771, 2009.

[19] Chun-Nan Hsu, Han-Shen Huang, Yu-Ming Chang, and Yuh-Jye Lee. Periodic step-size adaptation in second-order gradient descent for single-pass on-line structured learning. *Maching Learning*, 77:195–224, December 2009.

[20] Hal Daum Iii and Daniel Marcu. Np bracketing by maximum entropy tagging and svm reranking. In *In EMNLP*, pages 254–261, 2004.

[21] Seonho Kim, Juntae Yoon, Kyung-Mi Park, and Hae-Chang Rim. Two-phase biomedical named entity recognition using a hybrid method. In

Proceedings of The Second International Joint Conference on Natural Language Processing (IJCNLP-05), pages 646–657, 2005.

[22] Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artif. Intell.*, 97(1-2):273–324, 1997.

[23] Daphne Koller and Mehran Sahami. Toward optimal feature selection. In *ICML*, pages 284–292, 1996.

[24] Taku Kudo. CRF++: Yet another CRF toolkit, 2006. Available at http://chasen.org/ taku/software/CRF++.

[25] Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. Applying conditional random fields to japanese morphological analysis. In *In Proc. of EMNLP*, pages 230–237, 2004.

[26] Nojun Kwak and Chong-Ho Choi. Input feature selection by mutual information based on parzen window. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(12):1667–1671, 2002.

[27] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289, 2001.

[28] D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Math. Program.*, 45(3):503–528, 1989.

[29] Robert Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of CoNLL-2002*, pages 49–55, 2002.

[30] Andrew McCallum. Efficiently inducing features of conditional random fields. In *Proceedings of the 19th Annual Conference on Uncertainty in*

Artificial Intelligence (UAI-03), pages 403–41, San Francisco, CA, 2003. Morgan Kaufmann.

[31] Andrew McCallum and Wei Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 188–191, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[32] Andrew Kachites McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

[33] Ryan McDonald, Koby Crammer, and Fernando Pereira. Online large-margin training of dependency parsers. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 91–98, 2005.

[34] Kevin P. Murphy, Yair Weiss, and Michael I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *In Proceedings of Uncertainty in AI*, pages 467–475, 1999.

[35] Daisuke Okanohara, Yusuke Miyao, Yoshimasa Tsuruoka, and Jun'ichi Tsujii. Improving the scalability of semi-markov conditional random fields for named entity recognition. In *ACL '06: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 465–472, 2006.

[36] Xuan-Hieu Phan, Le-Minh Nguyen, and Cam-Tu Nguyen. Flexcrfs: Flexible conditional random field toolkit, 2005. http://flexcrfs.sourceforge.net.

[37] Hoifung Poon and Pedro Domingos. Joint inference in information extraction. In *In Proceedings of the 22nd National Conference on Artificial Intelligence (2007*, pages 913–918.

[38] Sunita Sarawagi and William W. Cohen. Semi-markov conditional random fields for information extraction. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1185–1192, Cambridge, MA, 2005. MIT Press.

[39] B. Settles. Biomedical Named Entity Recognition Using Conditional Random Fields and Rich Feature Sets. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA)*, pages 104–107, 2004.

[40] B. Settles. ABNER: an open source tool for automatically tagging genes, proteins, and other entity names in text. *Bioinformatics*, 21(14):3191–3192, 2005.

[41] Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 134–141, 2003.

[42] Yanxin Shi and Mengqiu Wang. A dual-layer crfs based joint decoding method for cascaded segmentation and labeling tasks. In *IJCAI 2007: Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 1707–1712, 2007.

[43] Andreas Stolcke, Yochai Konig, Andreas Stolcke Yochai, and Mitchel Weintraub. Explicit word error minimization in n-best list rescoring, 1997.

[44] Charles Sutton and Andrew McCallum. Composition of conditional random fields for transfer learning. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 748–754, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

[45] Charles Sutton and Andrew McCallum. An introduction to conditional random fields for relational learning. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning.* MIT Press, 2006.

[46] Charles Sutton, Andrew McCallum, and Khashayar Rohanimanesh. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *J. Mach. Learn. Res.*, 8:693–723, 2007.

[47] B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *Advances in Neural Information Processing Systems (NIPS 2003)*, Vancouver, Canada, 2004. Winner of the Best Student Paper Award.

[48] B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning. Max-margin parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2004. Winner of the Best Paper Award.

[49] Ben Tasker, Abbeel Pieter, and Daphne Koller. Discriminative probabilistic models for relational data. In *Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence (UAI-02)*, pages 485–492, San Francisco, CA, 2002. Morgan Kaufmann.

[50] Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent

and structured output spaces. In *ICML '04: Proceedings of the Twenty-first International Conference on Machine Learning*, New York, NY, USA, 2004. ACM Press.

[51] Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[52] S. V. N. Vishwanathan, Nicol N. Schraudolph, Mark W. Schmidt, and Kevin P. Murphy. Accelerated training of conditional random fields with stochastic gradient methods. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 969–976, 2006.

[53] Martin J. Wainwright, Tommi Jaakkola, and Alan S. Willsky. Tree-based reparameterization for approximate estimation on loopy graphs. In *Advances in Neural Information Processing Systems (NIPS*, 2001.

[54] Lei Xu. A trend on regularization and model selection in statistical learning: A bayesian ying yang learning perspective. *Challenges for Computational Intelligence*, pages 365–406, 2007.