NAIST-IS-DD461029

Doctoral Dissertation

Biomedical Text Mining Based on Machine Learning: from Information Extraction to Coordination Identification

Kazuo Hara

March 17, 2008

Department of Information Processing
Graduate School of Information Science
Nara Institute of Science and Technology

Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Kazuo Hara

Thesis Committee: Yuji Matsumoto, Professor
Kiyohiro Shikano, Professor
Kentaro Inui, Associate Professor
Masashi Shimbo, Assistant Professor

Biomedical Text Mining Based on Machine Learning: from Information Extraction to Coordination Identification *

Kazuo Hara

Abstract

We focus on the information extraction from clinical trial MEDLINE abstracts. This is strongly connected with Evidence-based medicine (EBM), which has arisen from daily demands from patients for valid information about diagnosis, prognosis, therapy, and prevention, as well as from clinical doctors for up-to-date, correct, and effective sources of that information. At present, there are several practices that support EBM; one example is to manage systems that provide the summary of clinical studies for specified disorders. However, they mostly depend on human labor. Our future goal is to create a system that automatically summarizes original articles (such as clinical trial MEDLINE abstracts). As the necessary pre-processing, in this thesis, we concentrate on to develop the systems for information extraction and coordination disambiguation.

In information extraction we shall extract compared treatments and patient population from phase III clinical trial MEDLINE abstracts. There we shall see that to mark terms of treatments and patients is not so difficult, however, not all the marked terms are valid for our purposes: extraction targets should be the treatments and patients that directly connect with the focusing clinical trial. So we shall attempt to select terms by employing a sentence classification filter that can make use of grammatical structures of sentences. There we encounter with the difficulties when we analyze sentences containing coordinated phrases using the state-of-the-art parser. From a viewpoint that coordinated phrases are likely to include important information for EBM, this problem is also significant.

Then we propose a method for detecting and disambiguating coordinate phrases. Nearly all the previous methods focus on the construction of heuristic rules. Unlike

*Doctoral Dissertation, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD461029, March 17, 2008.

these approaches that lack both evidence in justification and inter-domain flexibility, we represent a coordinated phrase in the upper triangular shaped state space models based on the edit graphs. A unique feature of our method is that it employs a perceptron-like learning algorithm to adapt the substitution matrix to the training data drawn from the target language and domain. We obtained a promising empirical result in detecting and disambiguating coordinated noun phrases in the GENIA corpus, despite using a relatively small number of training examples with minimal features.

Keywords:

information extraction, biomedical text mining, evidence-based medicine, coordination identification, sequence alignment, machine learning

機械学習を用いたテキストマイニング －医療情報抽出から並列句解析まで－*

原 一夫

内容梗概

本研究は、臨床試験論文からの情報抽出に焦点を当てる。これは近年一般的になりつつあるコンセプトである「エヴィデンスに基づく医療（EBM）」と密接な関連をもつ。EBM の普及により、医療現場では診断、予後予測、治療、予防に関する最新で正確かつ効果的な方法についての知識が求められるが、それを支援するシステム作成は人手作業で行われているのが現状である。我々の最終的な目的は、医療文献を自動的に要約して EBM に必要な情報を患者や医師に提示するシステムの作成であるが、本論文では、その前処理として必要となる情報抽出タスクと並列句同定タスクについて論じる。

情報抽出タスクでは、既存の自然言語処理の技術を用いてどの程度の精度で重要情報抽出ができるかについて論じる（本博士論文の前半部分）。抽出対象はその臨床試験で比較する治療方法と対象患者である。そこで我々が最初に得る知見は、治療方法と患者を表す基本名詞句の切り出し自体は比較的容易に行うことができることである。しかし同時に、当該臨床試験で比較する治療方法ならびに対象とする患者だけを抽出するのは容易でないことも明らかになる。そこで我々は文分類によるフィルタリングを試みるが、文の構文構造を素性として用いようとするなら構文解析が成功することが前提となる。しかし、比較結果を記述する臨床試験論文においては、並列句が高頻度に出現する。そして、並列句の存在が構文解析を困難にすることは、自然言語処理学分野ではよく知られている。なおかつ、並列句は情報抽出の観点からも重要な情報を含みやすい。

そこで我々は、並列句同定法を新しく提案する（本博士論文の後半部分）。従来の手法はルールを発見的に作成するというものがほとんどである。これに対して、我々の提案手法は並列句同定問題を上三角形状の編集グラフにおける系列アラインメントの問題とみなし、編集コスト（素性の重み）を事前に与えることな

*奈良先端科学技術大学院大学 情報科学研究科 情報処理学専攻 博士論文, NAIST-IS-DD461029, 2008 年 3 月 17 日.

く、訓練データから学習することができる。GENIA コーパスを用いた実験で、従来手法と比較して良い並列句同定結果を得ることに成功した。なお、提案手法は医薬生物学分野以外のテキストにも適用可能な、自然言語処理の要素技術として用いることができる。

キーワード

情報抽出、医療テキストマイニング、エビデンスに基づく医療、並列句同定、系列アラインメント、機械学習

Acknowledgements

I would like to thank my supervisor, Professor Yuji Matsumoto for all his variable advice and suggestions.

I also greatly thank the other member of my thesis committee, Professor Kiyohiro Shikano and Associate Professor Kentaro Inui and Assistant Professor Masashi Shimbo for their advice on writing this thesis.

I am grateful to the other staff of Computational Linguistics Laboratory: Assistant Professor Masayuki Asahara for the valuable comments and advice. I would also like to thank all people in the Computational Linguistics Laboratory.

I also thank Hirotoshi Taira in NTT Communication Science Laboratories, Hideto Kazawa at Google Inc., and Hiroya Takamura at the Tokyo Institute of Technology.

I am also grateful to my former supervisor, Professor Tatsuya Kubokawa and Fumiyasu Komaki at Tokyo University, and my superior Yasushi Orihashi at Daiichi-Sankyo Co. Ltd.

Contents

Acknowledgements **v**

1 Introduction **1**
1.1 Background 1
1.2 Research objectives 2
1.3 Dissertation Outline 6

2 Preliminaries **9**
2.1 Overview of the tasks and strategies 9
2.2 Modeling joint probability 11
2.2.1 Classification (naive Bayes classifier) 11
2.2.2 Sequence labeling (HMMs) 14
2.2.3 Sequence alignment (pair HMMs) 17
2.3 Modeling conditional probability 21
2.3.1 Classification (e.g. logistic regression) 21
2.3.2 Sequence labeling (linear chain CRFs) 22
2.3.3 CRFs for sequence alignment 24
2.3.4 The match or mismatch determination for pair sequences . . . 25
2.4 Maximum margin methods 28
2.4.1 Two class SVMs for classification 29
2.4.2 SVMs for sequence labeling and sequence alignment 34
2.5 The perceptron algorithms 34
2.5.1 Two class perceptron algorithms 35
2.5.2 The perceptron algorithms for sequence labeling and sequence alignment 37
2.6 Variants of the perceptron algorithms 37
2.6.1 Regularization 38
2.6.2 Lambda trick 38

2.6.3 MIRA; online perceptron taking the margin into consideration 39
2.6.4 Averaged perceptron 41
2.6.5 Bayes point machines 41

3 Extracting Clinical Trial Design Information from MEDLINE Abstracts 43
3.1 Introduction 43
3.2 Background and Objectives 44
3.2.1 Phase III clinical trials: directly linking to EBM 44
3.2.2 MEDLINE abstracts: available resource for EBM 44
3.2.3 Information extraction targets and data in this thesis 46
3.3 Information extraction applied to phase III abstracts 46
3.3.1 Base noun phrase chunking and its categorization 47
3.3.2 Methods and Results of automatic base noun phrase chunking and categorization 47
3.3.3 Regular expression pattern matching 49
3.3.4 Results of information extraction from MEDLINE abstracts . . 50
3.4 Filtering based on classification techniques 50
3.4.1 Document filtering 50
3.4.2 Methods and results of automatic document classification . . . 51
3.4.3 Sentence filtering 51
3.4.4 Methods and results of automatic sentence classification 52
3.4.5 Results of information extraction from MEDLINE abstracts with or without filtering 53
3.5 Discussion 56
3.6 Summary 58

4 A Discriminative Learning Model for Coordinations 59
4.1 Introduction 59
4.2 History of coordination disambiguation 60
4.3 Coordination disambiguation as sequence alignment 61
4.3.1 Sequence alignment 61
4.3.2 Edit graph for coordinations 62
4.4 Automatic parameter tuning 62
4.4.1 State space for learning coordinations 63
4.4.2 Learning task 65
4.4.3 More complex coordinations 66

4.5 Algorithms . . . 66
4.5.1 Reducing the cost of training data construction . . . 66
4.5.2 Path-based method . . . 67
4.5.3 Box-based method . . . 68
4.6 Related work . . . 70
4.6.1 Discriminative learning of edit distance . . . 70
4.6.2 Inverse sequence alignment in computational biology . . . 71
4.7 Empirical evaluation . . . 71
4.7.1 Dataset and Task . . . 71
4.7.2 Baselines . . . 72
4.7.3 Features . . . 73
4.7.4 Evaluation criteria . . . 74
4.7.5 Results and discussion . . . 75
4.8 Summary . . . 77

5 Conclusions **79**

Bibliography **81**

List of Publications **87**

List of Figures

1.1 A MEDLINE abstract 4
1.2 Coordination disambiguation by parsing 7
1.3 Coordination disambiguation by edit graph 8

2.1 An example of sequence alignment on the edit graph 10
2.2 A formal statement of classification, sequence labeling, and sequence alignment tasks on the supervised setting 12
2.3 A schematic view of naive Bayes 13
2.4 The first order Markov Model for sequence labeling (size of label set = 5) 15
2.5 The trellis for a given sequence (length = S) 16
2.6 An assumption of HMMs; observed tokens are emitted from unobserved (hidden) labels 16
2.7 The first order Markov Model for sequence alignment 18
2.8 The trellis for a given sequence pair (length = U and V) 19
2.9 An assumption of pair HMMs; observed tokens are emitted from unobserved (hidden) edit operations 20
2.10 a finite state machine for discriminating matched/mismatched sequence pairs 26
2.11 The trellis for a given sequence pair 27
2.12 A formal statement of coordination identification 28
2.13 Separating hyperplanes on contour maps 30
2.14 An illustration of batch perceptron learning 36
2.15 Comparison of the ways to update the weights between the ordinary online perceptron and the MIRA algorithm 40

3.1 The IE process in this thesis 47

4.1 An alignment between ‘writer’ and ‘vintner’, represented as a path in an edit graph 61
4.2 An edit graph for coordinate detection 64
4.3 Five node types created for a vertex in an edit graph: (a) *Inside Delete*, (b) *Inside Insert*, (c) *Inside Substitute*, (d) *Outside Delete*, and (e) *Outside Insert*. 64
4.4 Series of edit operations with an equivalent net effect. (a) $(Insert, Delete)$, and (b) $(Delete, Insert)$. (b) is prohibited in our model. 65
4.5 A coordination with four conjuncts represented as (a) chainable, and (b) non-chainable partial paths. We take (a) as the canonical representation. 66
4.6 Path-based algorithm 68
4.7 Box-based algorithm 69
4.8 Number of paths passing through the vertices of an 8×8 grid. 70

List of Tables

1.1 An output from the EBM support system (The Ovid) 3
1.2 Summary tables of a MEDLINE abstract 4
1.3 The resolution accuracy of "COOD" tags in GENIA corpus by using Charniak and Johnson's state-of-the-art parser 5

2.1 An example of sequence labeling task; base noun phrase identification 10
2.2 An example of sequence alignment task 10
2.3 Averaged perceptron; weights for testing $= \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{W}_t$ 40
2.4 Bayes point machines; weights for testing $= \frac{1}{M}\sum_{m=1}^{M} \boldsymbol{W}_{mT}$ 41

3.1 The number of abstracts registered in MEDLINE 45
3.2 Examples of IE targets 45
3.3 Categories for base NPs 46
3.4 Results of base NP chunking 48
3.5 Results of base NP categorization 48
3.6 Regular expression patterns for IE 49
3.7 Results of IE; # of abstracts correct information extracted 50
3.8 Results of document classification 51
3.9 Results of sentence classification using BACT 53
3.10 Results of IE with or without filtering in terms of "Patient Population"; # of abstracts correct information extracted 54
3.11 Results of IE with or without filtering in terms of "Compared Treatments"; # of abstracts correct information extracted 55
3.12 Comparison of F rate (%) for base NP chunking and categorization . . 57
3.13 The top categories in MeSH thesaurus 58

4.1 Features for the proposed methods 73
4.2 Performance on conjunct bracketing. P: precision (%), R: recall (%), F: F rate. 75

4.3 Performance on coordination bracketing. P: precision (%), R: recall (%), F: F rate. 76
4.4 Performance on coordination bracketing; path-based algorithms without using word and suffix features. P: precision (%), R: recall (%), F: F rate. 76

5.1 Evaluation of Charniak and Johnson's state-of-the-art parsing accuracy (recall) with respect to the "COOD" tags in GENIA corpus, stratified by non nested or nested coordination 80

Chapter 1

Introduction

1.1 Background

The goal of biomedicine is to uncover the mechanisms of life. Regarding life as a product whose design is given by genes, scientists in molucular biology are studying at the laboratory bench to examine gene dymamics, cell structures, cells complex (e.g. cancer or imune systems), using micro organisms or experiment animals. Meanwhile, clinical doctors are working beside beds to help patients by giving accurate diagnosis and selecting appropriate treatments. Despite the huge efforts from both biological scientists and clinical practitioners in decades, the mechanisms of life remain a mystery.

Such continued efforts result in a huge volume of biomedical documents; experimental results from bench at biological laboratories and clinical records from bedsides in medical facilities. Although each document is in itself valuable, dealing with them as an aggregate often leads to new knowledge. Two experimental results, one supporting the interaction between gene A and gene B and the other supporting the interaction between gene B and gene C, suggests possible connections between gene A and gene C. We also benefit from aggregating documents when, for example, it strengthens the confidence of the results in case that the number of collected texts presenting similar results are large. Processing text archives for the purposes such as knowledge acquisition, knowledge confidence evaluation, information extraction and summarization called *text mining*.

Having been recognized its importance in the public, text mining has become popular in both the biomedical and NLP (natural language processing) community. Mining goals includes, as an example in molucular biology domain, gene name identification, gene/protein name normalization and extraction of protein-protein interaction

from scientific research paper articles, which are studied in the shared task of BioCreAtIvE [26], and as two examples of shared tasks in clinical domain, de-identication (or anonymization) of personal health information from discharge summaries [58] and assignment of ICD-9-CM codes (i.e. disease codes) to radiology reports [44].

Here we point out that biomedical text mining is not a task that can not be realized. We are aware that mining from biomedical documents would be easier than from other text documents. One reason is that scientific article writers are supposed to compose sentences with less ambiguities. To do so they have a tendency to avoid ellipsis, for example. More specifically we have no trouble in determining sentence boundaries when we handles scientific articles, which is not the case for mining from web documents. In addition, compared to newspaper articles that cover a wide area of subjects, the number of terms occurring in biomedical texts are expected to be quite small, which is advantageous for text mining. Nonetheless biomedical text mining is still a challenge for us, because we face semantic and syntactic ambiguities that are difficult to disambiguate automatically, even in such biomedical documents.

1.2 Research objectives

In this thesis, we focus on the information extraction from clinical trial MEDLINE abstracts. This is strongly connected with *Evidence-based medicine* (EBM [48]), the concept that has been popularized rapidly. The EBM has arisen from daily demands from patients for valid information about diagnosis, prognosis, therapy, and prevention, as well as from clinical doctors for up-to-date, correct, and effective sources of that information [53]. At present, there are several systems that support EBM; one example is the Ovid (http://clinicalevidence.bmj.com/). Although not free, it provides the summary of clinical studies for specified disorders, as illustrated in Table 1.1, which is apparently useful to acquire current best evidence. Our future goal is, in short, to arrange such tables automatically from original articles such as clinical trial MEDLINE abstracts. As the necessary pre-processing, in this thesis, we concentrate on developing the systems for information extraction and coordination disambiguation.

In order to explain why the information extraction and coordination disambiguation are important to create summary tables, we present an example using an actual MEDLINE abstract [51]. In Figure 1.1, bold texts are clinically important terms, that is, name of diseases, drugs, outcomes, interventions, treatments and the phrases describing experimental conditions or schedules. It is important to note that just to mark key

Table 1.1: An output from the EBM support system (The Ovid)

Study	Interventions	Duration (years)	Outcome	Events/sample size (%)	
				Intervention	Control
UKPDS	"Tight" target BP (<150/<85) with captopril or atenolol vs. "less-tight" target (<180/<105)	8.4	AMI (fatal or non-fatal) stroke Peripheral vascular events	107/758 (14%)	83/390 (21%)
HOT	Felodipine and ACE inhibitor, or beta-blocker, with three distinct target BPs	3.8	AMI (fatal or non-fatal), stroke (fatal or non-fatal), or other cardiovascular death	22/499 (4.4%)	45/501 (9.0%)
FACET	Fosinopril vs. amiodipine	2.9	AMI, stroke, or admission to hospital for angina	14/189 (7.4%)	27/191 (14%)
ABCD	Enalapril vs. nisoldipine	5	AMI (fatal or non-fatal) MI, CHF, or sudden cardiac death	5/235 (2.1%)	25/235 (11%)
Syst-Eur	Nitrendipine; enalapril + hydrochlorothiazide vs. placebo	2	MI, CHF, or sudden cardiac death	13/252 (5%)	31/240 (13%)
AFCAPS/TexCAPS	Lovastatin	5	MI, unstable angina, or sudden cardiac death	4/84 (4.8%)	6/71 (8.5%)
SENDCAP	Bezafibrate	3	MI, or new ischemic changes on ECG	5/64 (7.8%)	16/64 (25%)
Helsinki	Gemfibrozil	5	MI or cardiac death	2/59 (3.4%)	8/76 (10.5%)
HPS	Simvastatin vs. placebo	5	CHD, stroke, revascularization	133/1455 (9.1%)	197/1457 (13.5%)

ABSTRACT:

PURPOSE: The combination of **paclitaxel** with **carboplatin** is effective in **advanced-stage non-small cell lung cancer** (**NSCLC**). This phase III study was designed to compare the **efficacy** *and* **tolerability** of **a weekly** *versus* **an every-3-week** schedule in **the first-line treatment** of **advanced-stage NSCLC**. PATIENTS AND METHODS: Chemotherapy-naive patients were randomized to receive **paclitaxel 100 mg/m2** *and* **carboplatin at an area under the curve of 2 once weekly for 6-8 weeks (arm A)** *or* **paclitaxel 200 mg/m2** *and* **carboplatin at an area under the curve of 6 on day 1 every 21 days (arm B)**. RESULTS: A total of 883 patients received >or= 1 chemotherapy cycle and were included in the results. **The objective response rates** observed (complete response plus partial response) were **38%** for **arm A** *and* **33%** for **arm B**. **Median times to progression** *and* **median survival times** were **6.1 months** *and* **8.9 months** in **arm A** *and* **7.2 months** *and* **9.5 months** in **arm B**, respectively. There were no significant differences between treatment arms. The chemotherapy was well tolerated in both schedules. However, **grade 3/4 sensory neuropathy** occurred more frequently with **the every-3-week schedule** (**9.1%** *vs.* **4.4%**), whereas **grade 3/4 diarrhea** occurred more frequently with **the weekly schedule** (**4.2%** *vs.* **1.1%**). CONCLUSION: In terms of **response** *and* **survival**, **paclitaxel/carboplatin** administered **once weekly** is comparable with **the every-3-week schedule**. **Toxicity** differences should be considered when choosing the appropriate schedule for the individual.

Figure 1.1: A MEDLINE abstract; bold texts are clinical terms. Coordinations (underlined) are important to create summary tables (below).

Table 1.2: Summary tables constructed from a MEDLINE abstract

Experimental setting:

	paclitaxel	carboplatin	(schedule)
arm A	100 mg/m2	AUC of 2	once weekly for 6-8 weeks
arm B	200 mg/m2	AUC of 6	on day 1 every 21 days

Comparison of efficacy:

	response rates	median times to progression	median survival times
arm A	38 %	6.1 months	8.9 months
arm B	33 %	7.2 months	9.5 months

Comparison of safety:

	grade 3/4 sensory neuropathy	grade 3/4 diarrhea
arm A	4.4 %	4.2 %
arm B	9.1 %	1.1 %

Table 1.3: The resolution accuracy of "COOD" tags in GENIA corpus by using Charniak and Johnson's state-of-the-art parser

COOD type	# of tags	precision (%)	recall (%)	F rate
NP	2591	45.7	41.8	43.7
VP	485	63.3	61.6	62.5
ADJP	409	41.2	46.9	43.9
S	283	57.7	41.0	47.9
PP	209	49.8	48.3	49.0

terms reduces reader's labor, and which is the first process towards information extraction. We also show in Figure 1.1 that such key terms very often occur in coordinations (underlined in the figure). This is because the aims of clinical trials are the comparison between new and old treatments. In many cases, the conjuncts consist of the experimental settings or the results regarding efficacy and safety, those are the very contents in the summary tables, as shown in Table 1.2.

Coordination Identification is considered as an important subject not only in the practical biomedical informatics. It is also the task involving disambiguation difficulties in the general NLP, conventionally solved by parsers. The recent state-of-the-art parsers are statistical ones that output most probable grammatical structures from the input sentences. Although their overall performance is excellent, that achieves around 90% accuracy in f-scores [9, 13, 10], they have weakness in coordination identification as shown in Table 1.3, where we summarize the result of identifying (the regions of) coordinated phrases occurred in the GENIA corpus, using the most accurate parser originated from Charniak and Johnson [10]. We consider that one reason for such low accuracy comes from the fact that conventional parsers have payed less attention to incorporate conjunct parallelism into the models. Therefore we shall focus on the point in Chapter 4.

Another cause of disambiguation difficulties is ellipsis in coordinations, as has been discussed by many researchers [45, 23, 8, 40, 7]. In case conjuncts share tokens (e.g., "ripe" is shared in "ripe apples and ripe bananas") the expression tends to be shortened by eliminating shared components ("ripe apples and bananas"). However this raises the problem of ambiguity when the context provides few clues for readers to disambiguate the interpretations either with or without ellipsis ("ripe apples and ripe bananas" or "ripe apples and bananas", respectively). In the bracketing guidelines for

Penn Treebank [4, Chapter 8], the authors described in fairly detailed manner a way to annotate shared or unshared complements and modifiers in coordination structures, this implies that elliptical coordinations accommodate such intrinsic disambiguation difficulties.

One way to resolve elliptical coordinations is to resort to external knowledge sources (e.g., such as dictionaries, ontologies, and Web documents). For example, see the coordinated phrases below, where brackets indicate the correct regions.

- (IL-2 and IL-2-R alpha) promoters
- either (a polyclonal or a clonal) immune response

we reach the correct answers when we find "IL-2 promoters" and "a polyclonal immune response" as proper expressions in some knowledge sources. However, we will not focus on the problem in this thesis, and leave the issue for future work.

1.3 Dissertation Outline

In this thesis, we focus on developing a system for information extraction and coordination disambiguation. The systems of text mining are modularly developed in general, such that include sentence segmentation, part-of-speech tagging, base noun phrase chunking, named entity extraction, sentence classification, document classification, and parsing. In each component, it is common that the techniques based on statistical machine learning are employed if an annotated corpus (i.e. document archives where target information is marked by human labor) is available, and rule based approaches are used otherwise. Regarding some of those components, in Chapter 3, we make use of developed techniques (and their software packages). In Chapter 4 we concentrate on developing a novel coordination identification technique, that can make up for the weakness of the state-of-the-art parsers.

This thesis is organized as follows: In Chapter 2, we overview the tasks (i.e. classification, sequence labeling, and sequence alignment) for which we shall employ or develop machine learning systems in this thesis. Then, we introduce some conventional approaches to the tasks and also explain the relation between these and our proposing method for coordination disambiguation.

In Chapter 3 we shall extract compared treatments and patient population from clinical trial MEDLINE abstracts. There we see that to mark terms regarding treatments

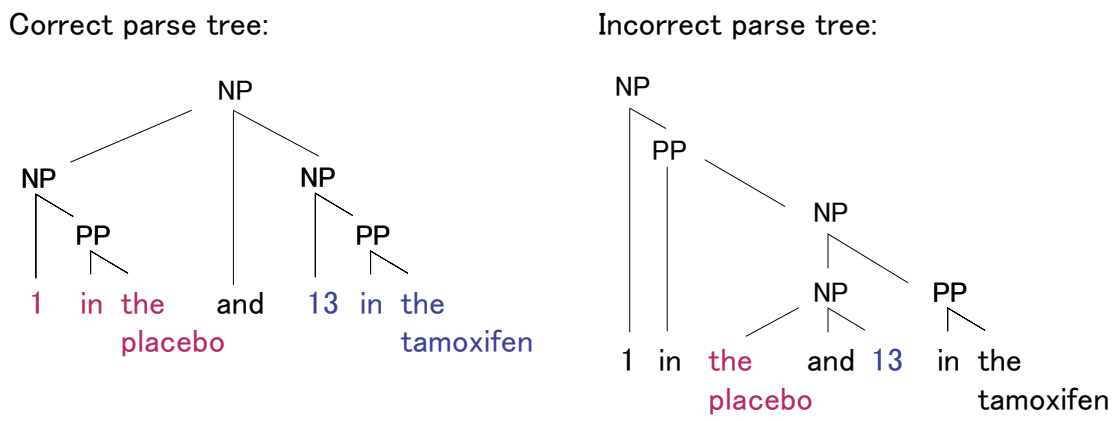


Figure 1.2: Coordination disambiguation by parsing

and patients is not so difficult, however, not all the marked terms are valid for our purposes; extraction targets should be the treatments and patients directly connected with the focusing clinical trial. So we shall attempt to select terms by employing a sentence classification filter that can make use of grammatical structures of sentences. There we encounter with a problem caused from parsing difficulties when we deal with sentences containing coordinated phrases. The situation is illustrated in Figure 1.2, where two possible parse trees for a phrase

"1 in the placebo and 13 in the tamoxifen"

are displayed. The correct parse tree is shown with an incorrect one. From a viewpoint that coordinated phrases are likely to include important information for EBM, this problem is also crucial.

In Chapter 4 we propose a method for detecting and disambiguating coordinate phrases. Nearly all the previous methods except parsing focus on the construction of heuristic rules. Unlike these approaches that lack both evidence in justification and inter-domain flexibility, we shall represent a coordination in the upper triangular shaped state space models based on the edit graphs. Figure 1.3 shows our representation of a coordination, using the same example as Figure 1.2. After learning model parameters, we shall see our proposing method achieves higher accuracy in the experiments using the GENIA corpus.

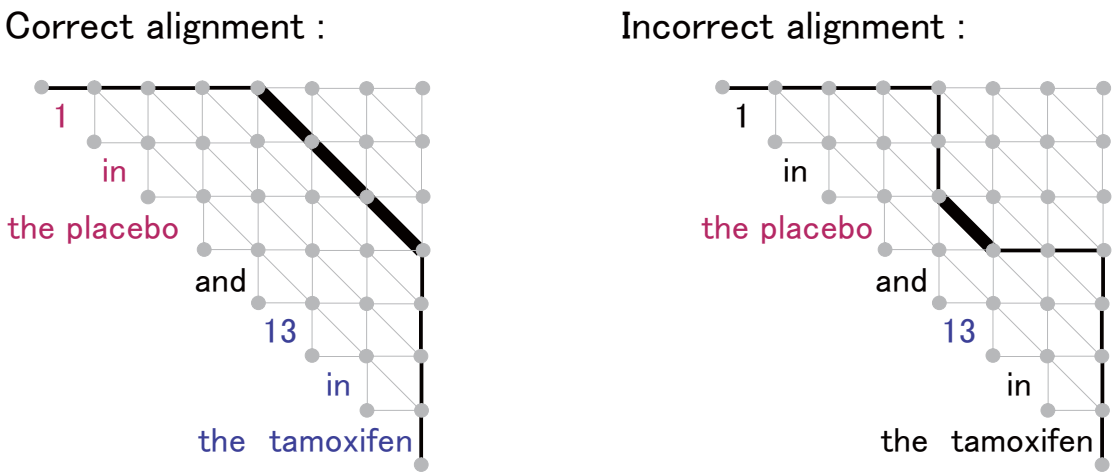


Figure 1.3: Coordination disambiguation by edit graph

Chapter 2

Preliminaries

2.1 Overview of the tasks and strategies

Here we overview classification, sequence labeling, and sequence alignment tasks, for which we shall employ or develop machine learning systems in this thesis. In Chapter 3, we make use of SVMs (Support Vector Machines) for text classification, employ a classifier based on a boosting algorithm for the tree structured data, and apply CRFs (Conditional Random Fields) to the tasks of sequence labeling such as base noun phrase identification and categorization. We shall resort to distributed software packages for these above. On the other hand, in Chapter 4 we propose a system based on the averaged perceptron for coordination disambiguation. There we see the problem as a task of sequence alignment, which shares properties with sequence labeling.

Although it is straightforward to say that the classification task is to assign one of given labels to an input object, both sequence labeling and sequence alignment tasks are rather complicated. In Table 2.1 we present an example of base noun phrase identification task realized within the framework of sequence labeling. There the goal is to classify each word into one among "B", "I", "O" labels, that means, beginning, inside, and outside of a base noun phrase, respectively. It differs from a simple classification task in that words occur (in a sentence) correlated with each other. For example, the word next to a determiner (e.g. "an") is expected to be the "I" tag with very high probability, because a determiner usually comprises a noun phrase with subsequent words. To incorporate such knowledges into a prediction model, one option, but most prevalently used, is taking the Markov assumption between labels.

Sequence alignment is a task to line up two sequences in accordance with their similarities. The output of the task forms a sequence of edit operations. A simple set of edit

Table 2.1: An example of sequence labeling task; base noun phrase identification

x_{seq}	Time	flies	like	an	arrow	.
y_{seq}	B	O	O	B	I	O

Table 2.2: An example of sequence alignment task

x_{seq}	99%	for	the			standard	arm
y_{seq}	182%	for	the	dose	dense		arm
e_{seq}	S	S	S	I	I	D	S

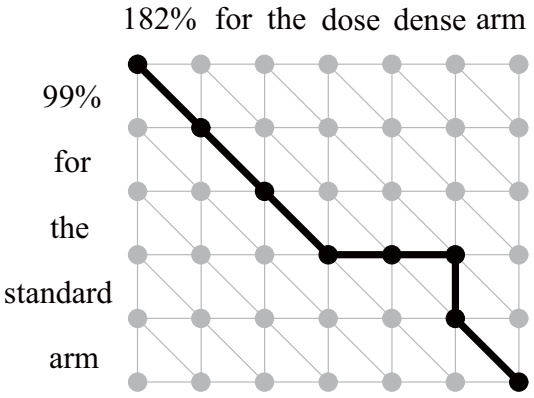


Figure 2.1: An example of sequence alignment on the edit graph

operations consists of three types; substitution, deletion, and insertion (we abbreviate those by "S", "D", and "I", respectively). In case considering word sequence alignment, "S" operation indicates that a component word from the first sequence can be substituted by one from the second (with paying low costs) because they are similar. "D" and "I" operations mean that there is no corresponding word in the opposite sequence, with respect to a component from the first and second sequences, respectively. This is illustrated in Table 2.2. Furthermore, a result of sequence alignment can be drawn on the edit graph, where "S", "D", and "I" operations correspond to diagonal, vertical, and horizontal lines respectively, as shown in Figure 2.1. The edit graph is very convenient to represent a sequence alignment on the Markov Model.

We give a formal statement of classification, sequence labeling, and sequence alignment tasks in Figure 2.2, where prediction functions $\hat{f}_{pred}$ are optimized in terms of the following strategies:

- Maximum likelihood (or maximun a posteriori) estimate on the joint (input, output) probability models,
- Maximum likelihood (or maximun a posteriori) estimate on the conditional (output given input) probability models,
- Maximum margin from a separating hyperplane in the feature space (i.e. SVMs), and,
- Search for a separating hyperplane by correcting errors (i.e. Perceptron),

In this chapter we outline the above strategies in turn, in applying some conventional ways to the tasks of classification and sequence labeling, and explaining recent exploitation towards sequence alignment including our proposing methods for coordination disambiguation.

2.2 Modeling joint probability

2.2.1 Classification (naive Bayes classifier)

A traditional way to optimize $\hat{f}_{pred}$ is based on the estimation of joint probability that generates data (i.e. input output pair). It is parameterized by Θ, that is,

$$Prob(input, output|\Theta).$$

Fit a prediction function $\hat{f}_{pred}$ in terms of an optimization principle.

Classification

Given N independent instances $\{\boldsymbol{x}^{(n)}, y^{(n)}\}_{n=1}^{N}$ as training data.

input: $\boldsymbol{x} = (x_1, x_2, \ldots, x_D)^T \in \boldsymbol{R}^D$

output: $y \in L$, $L = \{\text{a set of labels; } l_1, l_2, \ldots, l_K\}$

prediction function $\hat{f}_{pred} : \boldsymbol{X} \subseteq \boldsymbol{R}^D \rightarrow L$

Sequence Labeling

Given N independent instances $\{x_{seq}^{(n)}, y_{seq}^{(n)}\}_{n=1}^{N}$ as training data.

input: $x_{seq} = x_1 x_2 \ldots x_{S_n} \in W_{seq}$, $x_s \in W = \{\text{a set of tokens}\}$

output: $y_{seq} = y_1 y_2 \ldots y_{S_n} \in L_{seq}$, $y_s \in L = \{\text{a set of labels}\}$

prediction function $\hat{f}_{pred} : X_{seq} \subseteq W_{seq} \rightarrow L_{seq}$

Sequence Alignment

Given N independent instances $\{x_{seq}^{(n)}, y_{seq}^{(n)}, e_{seq}^{(n)}\}_{n=1}^{N}$ as training data.

input: $x_{seq} = x_1 x_2 \ldots x_{U_n} \in W_{seq}$, $x_u \in W = \{\text{a set of tokens}\}$

$y_{seq} = y_1 y_2 \ldots y_{V_n} \in W_{seq}$, $y_v \in W = \{\text{a set of tokens}\}$

output: $e_{seq} = e_1 e_2 \ldots e_{Z_n} \in E_{seq}$, $e_z \in L = \{\text{a set of edit operations}\}$

prediction function $\hat{f}_{pred} : X_{seq} \times Y_{seq} \subseteq W_{seq} \times W_{seq} \rightarrow E_{seq}$

Figure 2.2: A formal statement of classification, sequence labeling, and sequence alignment tasks on the supervised setting. We can replace W with other set according to the application domains.

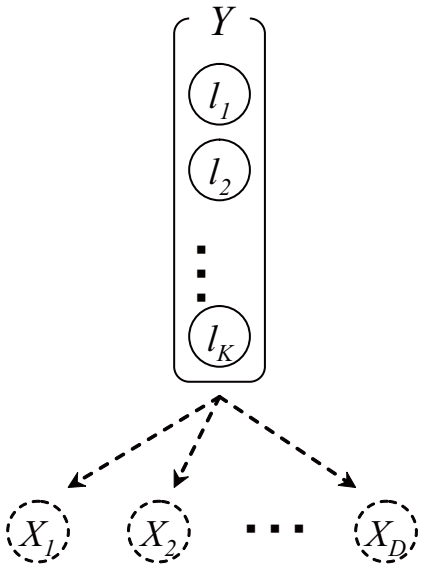


Figure 2.3: A schematic view of naive Bayes

The probability of generating N independent training data $\{x^{(n)}, y^{(n)}\}_{n=1}^{N}$ is

$$\prod_{n=1}^{N} Prob(\boldsymbol{x}^{(n)}, y^{(n)}|\boldsymbol{\Theta}),$$

from which we get maximum likelihood estimate of $\boldsymbol{\Theta}$ as

$$\boldsymbol{\Theta}_{opt} = \underset{\boldsymbol{\Theta}}{\operatorname{argmax}} \prod_{n=1}^{N} Prob(\boldsymbol{x}^{(n)}, y^{(n)}|\boldsymbol{\Theta}), \tag{2.1}$$

and therefore, we obtain a prediction function for a new input $\boldsymbol{x}^{(test)}$ to be

$$\hat{f}_{pred}(\boldsymbol{x}^{(test)}) = \underset{y \in L}{\operatorname{argmax}}\, Prob(\boldsymbol{x}^{(test)}, y|\boldsymbol{\Theta}_{opt}).$$

When D (the dimension of input data space) is large, precise estimation of probability distribution on $\boldsymbol{R}^D \times L$ requires a great number of training data. In such cases, a simple but very powerful approximation, so-called naive Bayes assumption is useful, such that

$$Prob(\boldsymbol{x}|y) = \prod_{d=1}^{D} Prob(x_d|y).$$

Maximizing (log) likelihood, we get an optimal parameter estimate under the assumption, such that

$$\begin{aligned}
\boldsymbol{\Theta}_{opt} &= \underset{\boldsymbol{\Theta}}{\text{argmax}}\, log \prod_{n=1}^{N} Prob(\boldsymbol{x}^{(n)}, y^{(n)}|\boldsymbol{\Theta}) \\
&= \underset{\boldsymbol{\Theta}}{\text{argmax}} \sum_{n=1}^{N} logProb(\boldsymbol{x}^{(n)}, y^{(n)}|\boldsymbol{\Theta}) \\
&= \underset{\boldsymbol{\Theta}}{\text{argmax}} \sum_{n=1}^{N} logProb(y^{(n)}|\boldsymbol{\Theta}) Prob(\boldsymbol{x}^{(n)}|y^{(n)}, \boldsymbol{\Theta}) \\
&= \underset{\boldsymbol{\Theta}}{\text{argmax}} \sum_{n=1}^{N} logProb(y^{(n)}|\boldsymbol{\Theta}) \prod_{d=1}^{D} Prob(x_d^{(n)}|y^{(n)}, \boldsymbol{\Theta}) \\
&= \underset{\boldsymbol{\Theta}}{\text{argmax}} \sum_{n=1}^{N} logProb(y^{(n)}|\boldsymbol{\Theta}) + \sum_{n=1}^{N} \sum_{d=1}^{D} logProb(x_d^{(n)}|y^{(n)}, \boldsymbol{\Theta}). \qquad (2.2)
\end{aligned}$$

A prediction function for a new input $\boldsymbol{x}^{(test)}$ is

$$\begin{aligned}
\hat{f}_{pred}(\boldsymbol{x}^{(test)}) &= \underset{y \in L}{\text{argmax}}\, Prob(\boldsymbol{x}^{(test)}, y|\boldsymbol{\Theta}_{opt}) \\
&= \underset{y \in L}{\text{argmax}}\, logProb(y|\boldsymbol{\Theta}_{opt}) + \sum_{d=1}^{D} logProb(x_d^{(n)}|y, \boldsymbol{\Theta}_{opt}). \qquad (2.3)
\end{aligned}$$

In Figure 2.3, a schematic view of naive Bayes model is shown, which corresponds to (2.2); selecting a label out of K types (K circles in figure) with $Prob(y|\boldsymbol{\Theta})$ (first term in equation), and generating D real numbers independently from the selected label (dotted arrows in figure) with $Prob(\boldsymbol{x}_d|y, \boldsymbol{\Theta})$ (second term in equation).

So far we do not give a specific form with respect to $Prob(x|y, \boldsymbol{\Theta})$. Popular ones are Gaussian, Multinomial, and Bernoulli distributions, which construct so-called Gaussian, Multinomial, Bernoulli mixture models, respectively, with K mixing coefficients given as $Prob(y|\boldsymbol{\Theta})$. We can use the EM algorithm to optimize parameters, and also incorporate priors to avoid overfitting.

2.2.2 Sequence labeling (HMMs)

Here we introduce the Hidden Markov Models (HMMs) that had been playing an important role from the early study of natural language processing, as well as speech

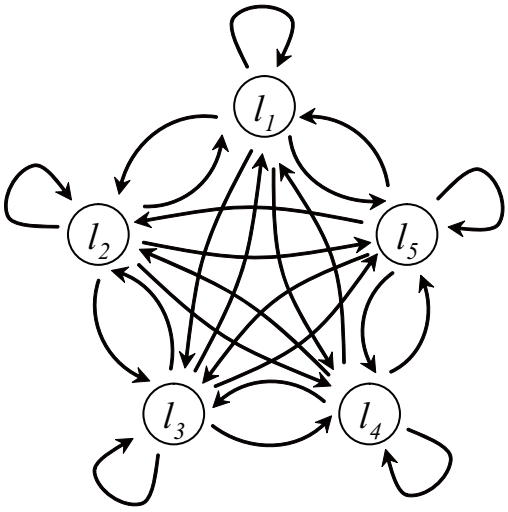


Figure 2.4: The first order Markov Model for sequence labeling (size of label set = 5)

recognition and genome informatics. In this thesis, we assume first order Markov Models where each label depends on just a previous one. A finite state machine diagram for the Markov Model is given briefly in Figure 2.4, and the trellis for a given sequence is shown in Figure 2.5.

In the setting of HMMs, each token x_s is emitted from the corresponding label y_s (dotted arrows in Figure 2.6) with $Prob(x_s|y_s,\Theta)$. Together with label (or state) transition probability given as $Prob(y_s|y_{s-1},\Theta)$, the joint probability of token and label sequences is

$$Prob(x_{seq},y_{seq}|\Theta) = \prod_{s=1}^{S} Prob(y_s|y_{s-1},\Theta)Prob(x_s|y_s,\Theta), \tag{2.4}$$

where

$$Prob(y_1|y_0,\Theta) \equiv Prob(y_1|\Theta).$$

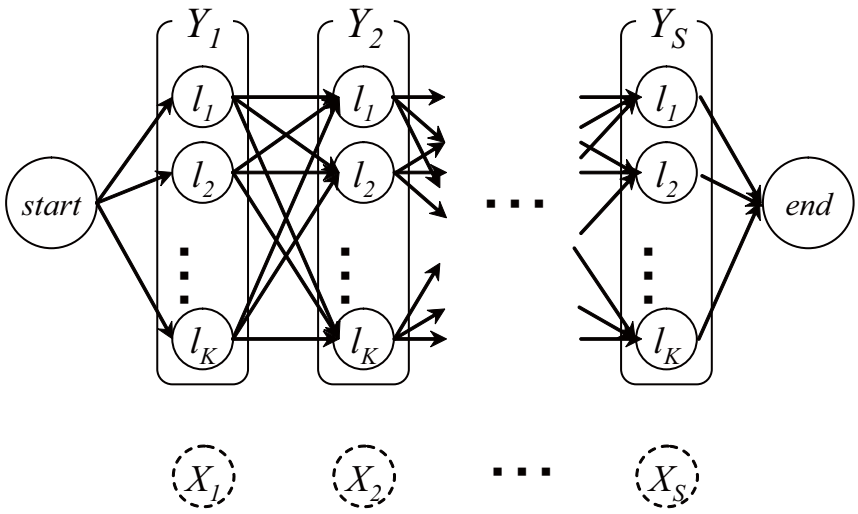


Figure 2.5: The trellis for a given sequence (length = S)

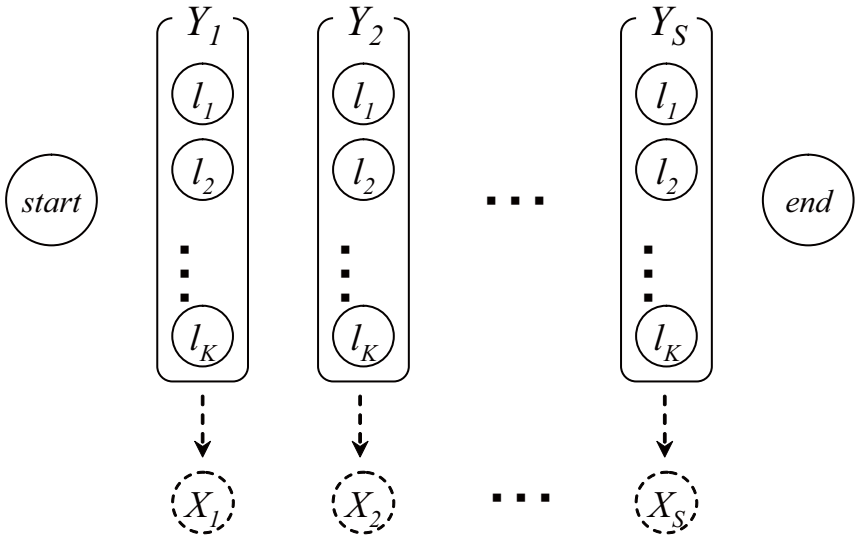


Figure 2.6: An assumption of HMMs; observed tokens are emitted from unobserved (hidden) labels

Then, a maximum (log) likelihood solution from N training instances is

$$\begin{aligned}
\Theta_{opt} &= \underset{\Theta}{\operatorname{argmax}} \, log \prod_{n=1}^{N} Prob(x_{seq}^{(n)}, y_{seq}^{(n)}|\Theta) \\
&= \underset{\Theta}{\operatorname{argmax}} \sum_{n=1}^{N} logProb(x_{seq}^{(n)}, y_{seq}^{(n)}|\Theta) \\
&= \underset{\Theta}{\operatorname{argmax}} \sum_{n=1}^{N} log \prod_{s=1}^{S_n} Prob(y_s^{(n)}|y_{s-1}^{(n)}, \Theta) Prob(x_s^{(n)}|y_s^{(n)}, \Theta) \\
&= \underset{\Theta}{\operatorname{argmax}} \sum_{n=1}^{N} \sum_{s=1}^{S_n} logProb(y_s^{(n)}|y_{s-1}^{(n)}, \Theta) + \sum_{n=1}^{N} \sum_{s=1}^{S_n} logProb(x_s^{(n)}|y_s^{(n)}, \Theta).
\end{aligned}$$

Therefore, we obtain a prediction function for a new input sequence $x_{seq}^{(test)}$ to be

$$\begin{aligned}
\hat{f}_{pred}(x_{seq}^{(test)}) &= \underset{y_{seq} \in L_{seq}}{\operatorname{argmax}} \, logProb(x_{seq}^{(test)}, y_{seq}|\Theta_{opt}) \\
&= \underset{y_{seq} \in L_{seq}}{\operatorname{argmax}} \sum_{s=1}^{S_n} logProb(y_s|y_{s-1}, \Theta_{opt}) + \sum_{s=1}^{S_n} logProb(x_s^{(n)}|y_s, \Theta_{opt}).
\end{aligned}$$

2.2.3 Sequence alignment (pair HMMs)

We introduce a variant of HMMs for the task of sequence alignment, that is widely known as pair HMMs. There, it is assumed that unobserved edit operations (in an output sequence) are generated from the Markov Model. As noted in the beginning of this chapter, a simplest set of edit operations consists of three types:"S", "D", and "I". Its finite state machine diagram (Figure 2.7) is the same as the one for sequence labeling task, however, the trellis for a given sequence pair shown in Figure 2.8 looks a little complicated. This is because it is also assumed that each edit operation emits observed (input) tokens and the way of emission differs according to the type of operation: "S" emits a pair of tokens x_u and y_v, but others emit a single token, that is, x_u for "D" and y_v for "I" (see Figure 2.9). It is important to note that the trellis is equivalent to the edit graph illustrated in Figure 2.1.

In analogy with Equation (2.4), the joint probability of input and output under the pair HMMs is

$$Prob(x_{seq}, y_{seq}|e_{seq}, \Theta) = \prod_{z=1}^{Z} Prob(e_z|e_{z-1}, \Theta) Prob(x_{u(z)}, y_{v(z)}|e_z, \Theta),$$

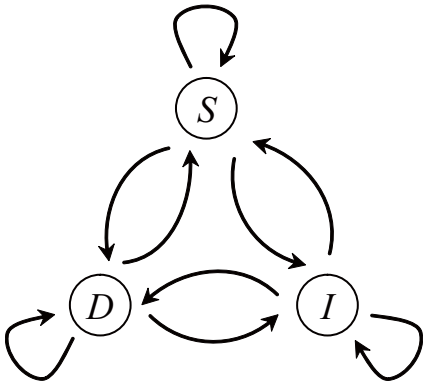


Figure 2.7: The first order Markov Model for sequence alignment

where, using indicator function $I(\cdot)$,

$$u(z) = \sum_{i=1}^{z} I(e_z = \text{"}S\text{"} \text{ or } e_z = \text{"}D\text{"}),$$
$$v(z) = \sum_{i=1}^{z} I(e_z = \text{"}S\text{"} \text{ or } e_z = \text{"}I\text{"}),$$

and also,

$$Prob(e_1|e_0, \Theta) \equiv Prob(e_1|\Theta),$$
$$Prob(x_{u(z)}, y_{v(z)}|e_z = \text{"}D\text{"}, \Theta) \equiv Prob(x_{u(z)}|e_z = \text{"}D\text{"}, \Theta),$$
$$Prob(x_{u(z)}, y_{v(z)}|e_z = \text{"}I\text{"}, \Theta) \equiv Prob(y_{v(z)}|e_z = \text{"}I\text{"}, \Theta).$$

Then, a maximum (log) likelihood solution from N training instances is

$$\begin{aligned}
\Theta_{opt} &= \underset{\Theta}{\text{argmax}} \, log \prod_{n=1}^{N} Prob(x_{seq}^{(n)}, y_{seq}^{(n)}, e_{seq}^{(n)}|\Theta) \\
&= \underset{\Theta}{\text{argmax}} \sum_{n=1}^{N} logProb(x_{seq}^{(n)}, y_{seq}^{(n)}, e_{seq}^{(n)}|\Theta) \\
&= \underset{\Theta}{\text{argmax}} \sum_{n=1}^{N} log \prod_{z=1}^{Z_n} Prob(e_z^{(n)}|e_{z-1}^{(n)}, \Theta) Prob(x_{u(z)}^{(n)}, y_{v(z)}^{(n)}|e_z^{(n)}, \Theta) \\
&= \underset{\Theta}{\text{argmax}} \sum_{n=1}^{N} \sum_{z=1}^{Z_n} logProb(e_z^{(n)}|e_{z-1}^{(n)}, \Theta) + \sum_{n=1}^{N} \sum_{z=1}^{Z_n} logProb(x_{u(z)}^{(n)}, y_{v(z)}^{(n)}|e_z^{(n)}, \Theta).
\end{aligned}$$

Therefore, we obtain a prediction function for a pair of new input sequences $(x_{seq}^{(test)}, y_{seq}^{(test)})$

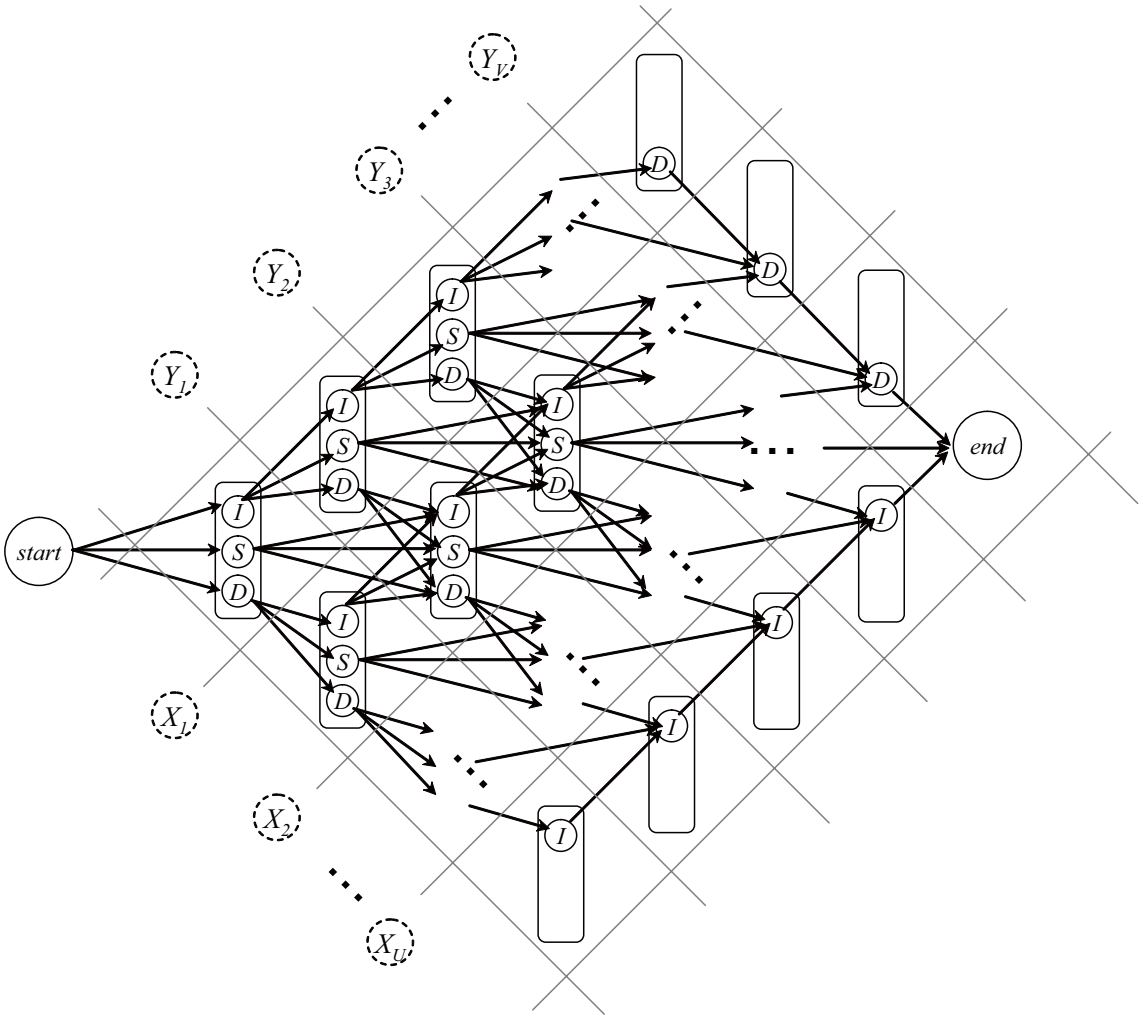


Figure 2.8: The trellis for a given sequence pair (length = U and V)

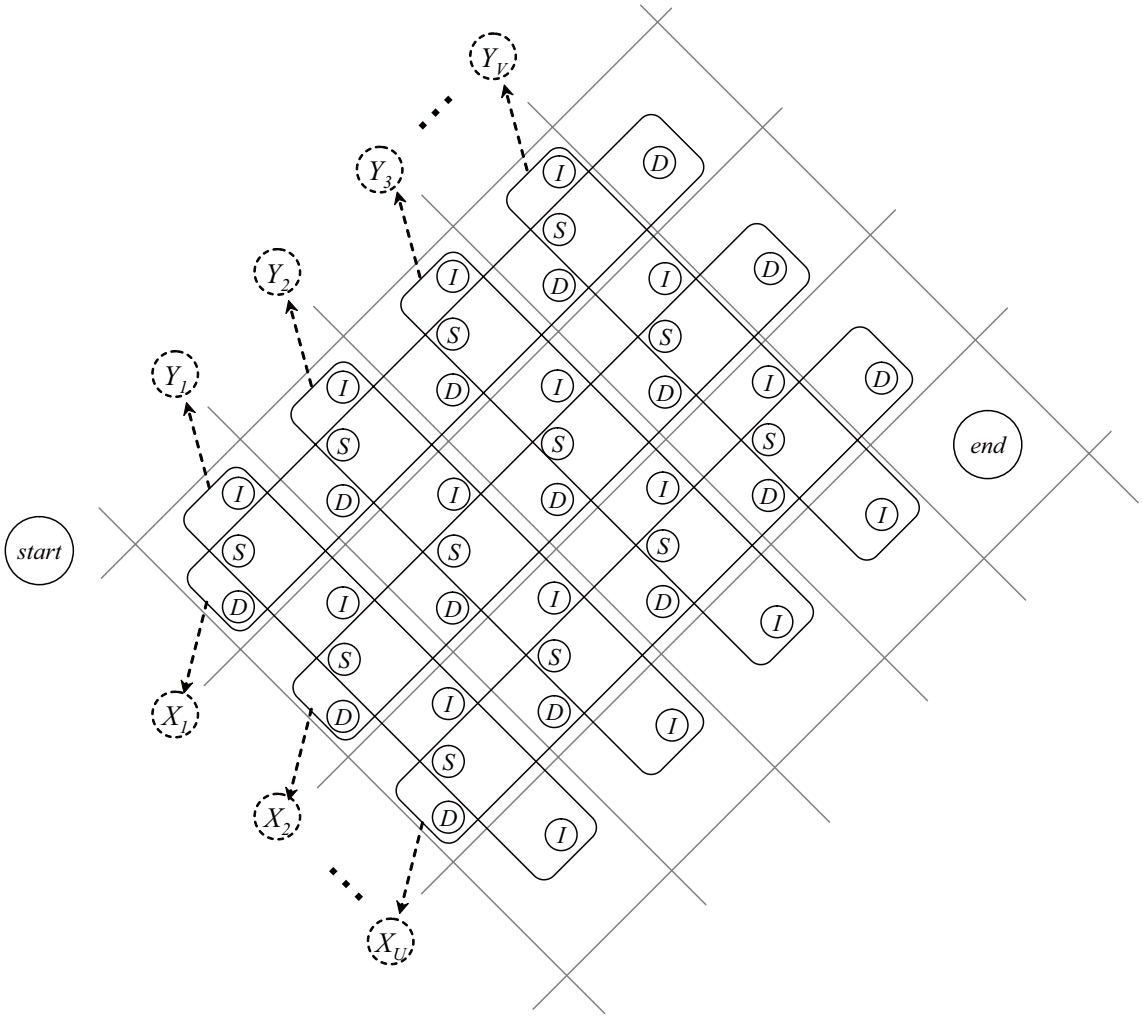


Figure 2.9: An assumption of pair HMMs; observed tokens are emitted from unobserved (hidden) edit operations

to be

$$\hat{f}_{pred}(x_{seq}^{(test)}, y_{seq}^{(test)}) = \underset{e_{seq} \in E_{seq}}{\operatorname{argmax}}\, logProb(x_{seq}^{(test)}, y_{seq}^{(test)}, e_{seq} | \Theta_{opt})$$
$$= \underset{e_{seq} \in E_{seq}}{\operatorname{argmax}} \sum_{z=1}^{Z_n} logProb(e_z | e_{z-1}, \Theta_{opt}) + \sum_{z=1}^{Z_n} logProb(x_{u(z)}^{(n)}, y_{v(z)}^{(n)} | e_z, \Theta_{opt}).$$

2.3 Modeling conditional probability

2.3.1 Classification (e.g. logistic regression)

An advantage of modeling joint probability, $Prob(input, output|\Theta)$, is that it is possible to obtain a distribution of inputs by marginalizing with respect to output variables. However, from a view of classification, it is often sufficient to estimate conditional probability, $Prob(output|input, \Theta)$, because we can conduct predictions by

$$\hat{f}_{pred}(new\ input) = \underset{output \in candidate\ set}{\operatorname{argmax}} Prob(output|new\ input, \Theta),$$

and this also saves the amount of training data compared to joint probability estimation.

Once we decide to leave inputs out of targets for probability estimation, we can use those as clues to estimate (conditional) probability of outputs. Combining with each label l_k, we can freely set feature functions $\boldsymbol{f}_k$ ($k = 1, \ldots, K$) such that

$$\boldsymbol{f}_k : \boldsymbol{R}^D \times L\ (= l_k)\ \rightarrow \boldsymbol{R}^{M_k},$$

and their corresponding M_k dimensional weight vectors $\boldsymbol{W}_k$ comprise the model parameter Θ, that is,

$$\Theta = (\boldsymbol{W}_1^T, \boldsymbol{W}_2^T, \ldots, \boldsymbol{W}_K^T)^T.$$

Then, the conditional model can be defined as

$$Prob(y = l_k | \boldsymbol{x}, \Theta) = \frac{1}{Z(\boldsymbol{x}, \Theta)} exp(\boldsymbol{W}_k^T \boldsymbol{f}_k(\boldsymbol{x}, y = l_k)),$$

where

$$Z(\boldsymbol{x}, \Theta) = \sum_{k=1}^{K} exp(\boldsymbol{W}_k^T \boldsymbol{f}_k(\boldsymbol{x}, y = l_k)),$$

and *exp* means the exponential function. For the moment, a well-known model called logistic regression is a special case when $\boldsymbol{f}_k$ is given as

$$\boldsymbol{f}_k(\boldsymbol{x}, y = l_k) = (1, x_1, x_2, \ldots, x_D)^T.$$

The log likelihood for N training instances is

$$\begin{aligned} L(\boldsymbol{\Theta}) &= log \prod_{n=1}^{N} Prob(y^{(n)} = l_{k^{(n)}} | \boldsymbol{x}^{(n)}, \boldsymbol{\Theta}) \\ &= \sum_{n=1}^{N} logProb(y^{(n)} = l_{k^{(n)}} | \boldsymbol{x}^{(n)}, \boldsymbol{\Theta}) \\ &= \sum_{n=1}^{N} \boldsymbol{W}_{k^{(n)}}^T \boldsymbol{f}_{k^{(n)}}(\boldsymbol{x}^{(n)}, y^{(n)} = l_{k^{(n)}}) - logZ(\boldsymbol{x}^{(n)}, \boldsymbol{\Theta}). \end{aligned}$$

At the point where $L(\boldsymbol{\Theta})$ is maximum, its derivative equals zero, so we obtain

$$\frac{\partial L}{\partial \boldsymbol{W}_k} = \sum_{n=1}^{N} \boldsymbol{f}_k(\boldsymbol{x}^{(n)}, y^{(n)} = l_k) - \sum_{n=1}^{N} \boldsymbol{f}_k(\boldsymbol{x}^{(n)}, y = l_k) Prob(y = l_k | \boldsymbol{x}^{(n)}, \boldsymbol{\Theta}) = 0.$$

We can get $\boldsymbol{\Theta}_{opt}$ using a technique based on the Newton-Raphson algorithm using the Hessian matrix of $L(\boldsymbol{\Theta})$. In case incorporating priors into parameters, we substitute $L(\boldsymbol{\Theta})$ for the one based on the posterior distribution. Then a prediction function for a new input $\boldsymbol{x}^{(test)}$ is given as

$$\begin{aligned} \hat{f}_{pred}(\boldsymbol{x}^{(test)}) &= \operatorname*{argmax}_{y \in L} Prob(y | \boldsymbol{x}^{(test)}, \boldsymbol{\Theta}_{opt}) \\ &= \operatorname*{argmax}_{y \in L} \boldsymbol{W}_{k,opt}^T \boldsymbol{f}_k(\boldsymbol{x}^{(test)}, y = l_k). \end{aligned}$$

2.3.2 Sequence labeling (linear chain CRFs)

In contrast to the classification task where there are only K types of the target label, large output-type variations occur in the sequence labeling task; K^S types with a sequence of length S. Except for the cases where K^S is small or training data is abundant, it is necessary to elaborate the design of feature functions if we intend to apply the same method, so-called log linear models, described in the previous section.

In the setting of linear chain CRFs, there are two types of feature functions, the one involved in a label with a component of output sequences and the other involved in

labels of adjacent components, such that

$$\boldsymbol{f}_k : X_{seq} \times L\,(= l_k) \;\rightarrow \boldsymbol{R}^{M_k}, \tag{2.5}$$

$$\boldsymbol{f}_{k'k} : X_{seq} \times L\,(= l_{k'}) \times L\,(= l_k) \;\rightarrow \boldsymbol{R}^{M_{k'k}}, \tag{2.6}$$

and their corresponding weight vectors $\boldsymbol{W}_k$ and $\boldsymbol{W}_{k'k}$ comprise the model parameter $\boldsymbol{\Theta}$, that is,

$$\boldsymbol{\Theta} = (\boldsymbol{W}_1^T, \boldsymbol{W}_2^T, \ldots, \boldsymbol{W}_K^T, \boldsymbol{W}_{11}^T, \boldsymbol{W}_{12}^T, \ldots, \boldsymbol{W}_{KK}^T)^T. \tag{2.7}$$

Then, the conditional model is formulated as

$$Prob(y_{seq}|x_{seq}, \boldsymbol{\Theta}) = \frac{1}{Z(x_{seq}, \boldsymbol{\Theta})} exp\ PathScore(x_{seq}, y_{seq}, \boldsymbol{\Theta}),$$

where

$$\begin{aligned} PathScore(x_{seq}, y_{seq}, \boldsymbol{\Theta}) &= \sum_{s=1}^{S} \boldsymbol{W}_k^T \boldsymbol{f}_k(x_{seq}, y_s = l_k) \\ &\quad + \sum_{s=1}^{S} \boldsymbol{W}_{k'k}^T \boldsymbol{f}_{k'k}(x_{seq}, y_{s-1} = l_{k'}, y_s = l_k), \qquad (2.8) \\ Z(x_{seq}, \boldsymbol{\Theta}) &= \sum_{y_{seq} \in L_{seq}} exp\ PathScore(x_{seq}, y_{seq}, \boldsymbol{\Theta}), \\ \boldsymbol{f}_{k'k}(x_{seq}, y_0 = l_{k'}, y_1 = l_k) &\equiv \boldsymbol{f}_{k'k}(x_{seq}, y_1 = l_k). \end{aligned}$$

The value of *PathScore* amounts to a weighted sum of feature function values corresponding to the "path" of label sequence on the trellis (shown in Figure 2.5). The log likelihood for N training instances is

$$\begin{aligned} L(\boldsymbol{\Theta}) &= log \prod_{n=1}^{N} Prob(y_{seq}^{(n)}|x_{seq}^{(n)}, \boldsymbol{\Theta}) \\ &= \sum_{n=1}^{N} logProb(y_{seq}^{(n)}|x_{seq}^{(n)}, \boldsymbol{\Theta}) \\ &= \sum_{n=1}^{N} PathScore(x_{seq}^{(n)}, y_{seq}^{(n)}, \boldsymbol{\Theta}) - logZ(x_{seq}^{(n)}, \boldsymbol{\Theta}). \end{aligned}$$

At the point where $L(\boldsymbol{\Theta})$ is maximum, its derivative equals zero, so we obtain

$$\begin{aligned}\frac{\partial L}{\partial \boldsymbol{W}_k} &= \sum_{n=1}^{N}\sum_{s=1}^{S_n} \boldsymbol{f}_k(x_{seq}^{(n)}, y_s^{(n)} = l_k) \\ &\quad - \sum_{n=1}^{N}\sum_{s=1}^{S_n} \boldsymbol{f}_k(x_{seq}^{(n)}, y_s = l_k) Prob(y_{seq}|x_{seq}^{(n)}, \boldsymbol{\Theta}) = 0, \qquad (2.9)\\ \frac{\partial L}{\partial \boldsymbol{W}_{k'k}} &= \sum_{n=1}^{N}\sum_{s=1}^{S_n} \boldsymbol{f}_{k'k}(x_{seq}^{(n)}, y_{s-1}^{(n)} = l_{k'}, y_s^{(n)} = l_k) \\ &\quad - \sum_{n=1}^{N}\sum_{s=1}^{S_n} \boldsymbol{f}_{k'k}(x_{seq}^{(n)}, y_{s-1} = l_{k'}, y_s = l_k) Prob(y_{seq}|x_{seq}^{(n)}, \boldsymbol{\Theta}) = 0. \qquad (2.10)\end{aligned}$$

To calculate second terms in each equation, dynamic programming algorithms can be efficiently used on the trellis. When the size of parameter vector $\boldsymbol{\Theta}$ is large (it is often the case), computing the Hessian matrix of $L(\boldsymbol{\Theta})$ becomes expensive, so it is common to approximate it using quasi-Newton methods (see [55] for details). After calculating $\boldsymbol{\Theta}_{opt}$ using some iterative methods, we get a prediction function for a new input $x_{seq}^{(test)}$ such that

$$\begin{aligned}\hat{f}_{pred}(x_{seq}^{(test)}) &= \underset{y_{seq}\in L_{seq}}{\operatorname{argmax}} Prob(y_{seq}|x_{seq}^{(test)}, \boldsymbol{\Theta}_{opt}) \\ &= \underset{y_{seq}\in L_{seq}}{\operatorname{argmax}} PathScore(x_{seq}^{(n)}, y_{seq}, \boldsymbol{\Theta}_{opt}).\end{aligned}$$

2.3.3 CRFs for sequence alignment

It is straightforward to extend linear chain CRFs to the task of sequence alignment, because output sequences can be represented on trellises in the same way as the sequence labeling task. In fact there is a work [19] that applies CRFs to protein sequence alignment. Since there are two input sequences, feature functions are revised to

$$\begin{aligned}\boldsymbol{f}_k &: X_{seq} \times Y_{seq} \times L\ (= l_k)\ \rightarrow \boldsymbol{R}^{M_k}, \\ \boldsymbol{f}_{k'k} &: X_{seq} \times Y_{seq} \times L\ (= l_{k'}) \times L\ (= l_k)\ \rightarrow \boldsymbol{R}^{M_{k'k}},\end{aligned}$$

where L is the set of edit operations. Then, the conditional model turns to be

$$Prob(e_{seq}|x_{seq}, y_{seq}, \boldsymbol{\Theta}) = \frac{1}{Z(x_{seq}, y_{seq}, \boldsymbol{\Theta})} exp\ PathScore(x_{seq}, y_{seq}, e_{seq}, \boldsymbol{\Theta}),$$

where

$$PathScore(x_{seq}, y_{seq}, e_{seq}, \Theta) = \sum_{z=1}^{Z} \boldsymbol{W}_k^T \boldsymbol{f}_k(x_{seq}, y_{seq}, e_z = l_k) + \sum_{z=1}^{Z} \boldsymbol{W}_{k'k}^T \boldsymbol{f}_{k'k}(x_{seq}, y_{seq}, e_{z-1} = l_{k'}, e_z = l_k),$$

$$Z(_{seq}, y_{seq}, \Theta) = \sum_{e_{seq} \in E_{seq}} exp\ PathScore(x_{seq}, y_{seq}, e_{seq}, \Theta).$$

2.3.4 The match or mismatch determination for pair sequences

Here we focus on a study that has close relationship to the method we shall propose in Chapter 4. So far, the goal of the sequence alignment task is to line up two distinct sequences. In this section we consider a variant of this, such that to determine whether a given sequence pair is similar to each other. A formal statement is given in the following:

The match or mismatch determination for pair sequences

Given N independent instances $\{x_{seq}^{(n)}, y_{seq}^{(n)}, m^{(n)}\}_{n=1}^{N}$ as training data.

input: $x_{seq} = x_1 x_2 \ldots x_{U_n} \in W_{seq}$, $x_u \in W = \{\text{a set of tokens}\}$

$y_{seq} = y_1 y_2 \ldots y_{V_n} \in W_{seq}$, $y_v \in W = \{\text{a set of tokens}\}$

output: $m \in M = \{+1; \text{ match}, -1; \text{ mismatch}\}$

Fit a prediction function $\hat{f}_{pred} : X_{seq} \times Y_{seq} \subseteq W_{seq} \times W_{seq} \rightarrow M$.

The solution described in [39] is based on CRFs, where McCallum et al. assumed a finite state machine comprising two parts such that the one for match states and the other for mismatch states (Figure 2.10). The two parts are connected only with the start state, therefore the trellis for a given sequence pair takes a parallel form (Figure 2.11). In other words, they augmented the label set to $L_+ \cup L_-$ where $L_+ = \{S_+, D_+, I_+\}$ and $L_- = \{S_-, D_-, I_-\}$, and restricted (unobserved) label sequences to either $E_{seq+} = \{e_{seq} = e_1 e_2 \ldots e_Z; e_z \in L_+\}$ for matched pairs, or $E_{seq-} = \{e_{seq} = e_1 e_2 \ldots e_Z; e_z \in L_-\}$ for mismatched pairs.

Then, their conditional model was given as

$$Prob(m = +1 | x_{seq}, y_{seq}, \Theta) = \frac{1}{Z(x_{seq}, y_{seq}, \Theta)} \sum_{e_{seq} \in E_{seq+}} exp\ PathScore(x_{seq}, y_{seq}, e_{seq}, \Theta), \tag{2.11}$$

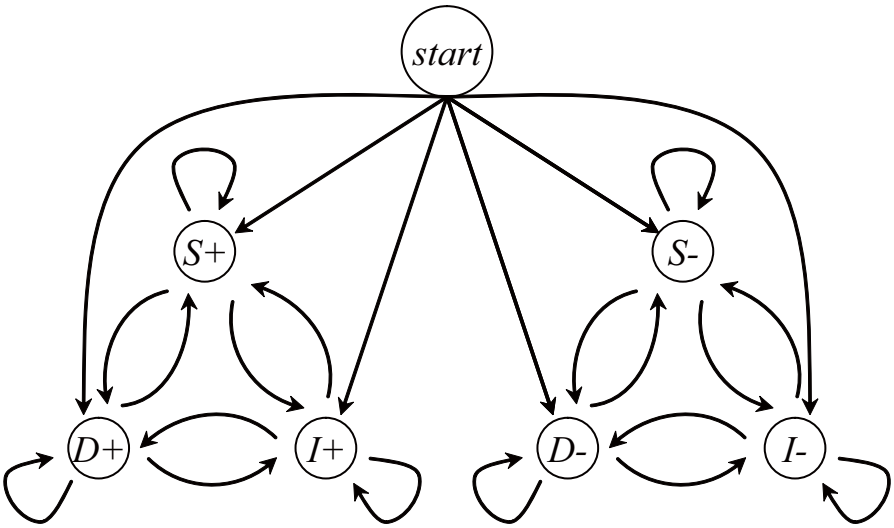


Figure 2.10: a finite state machine for discriminating matched/mismatched sequence pairs; "S+", "D+", "I+" imply matches, and "S-", "D-", "I-" imply mismatches.

$$Prob(m=-1|x_{seq},y_{seq},\Theta)=\frac{1}{Z(x_{seq},y_{seq},\Theta)}\sum_{e_{seq}\in E_{seq-}} exp\ PathScore(x_{seq},y_{seq},e_{seq},\Theta), \tag{2.12}$$

where

$$\begin{aligned} PathScore(x_{seq},y_{seq},e_{seq},\Theta) &= \sum_{z=1}^{Z} \boldsymbol{W}_k^T \boldsymbol{f}_k(x_{seq},y_{seq},e_z=l_k) \\ &\quad + \sum_{z=1}^{Z} \boldsymbol{W}_{k'k}^T \boldsymbol{f}_{k'k}(x_{seq},y_{seq},e_{z-1}=l_{k'},e_z=l_k), \end{aligned}$$

$$Z(x_{seq},y_{seq},\Theta)=\sum_{e_{seq}\in E_{seq+}\cup E_{seq-}} exp\ PathScore(x_{seq},y_{seq},e_{seq},\Theta).$$

At this point, in comparison with McCallum's work, we would like to give a profile of our method for coordination identification that we shall describe details in Chapter 4. In Figure 2.12, we give the task description. The important requirements are:

- to identify conjuncts **within a same word sequence** (= sentence),
- to discriminate **matched** conjunct pairs from **mismatched** word sequence pairs,

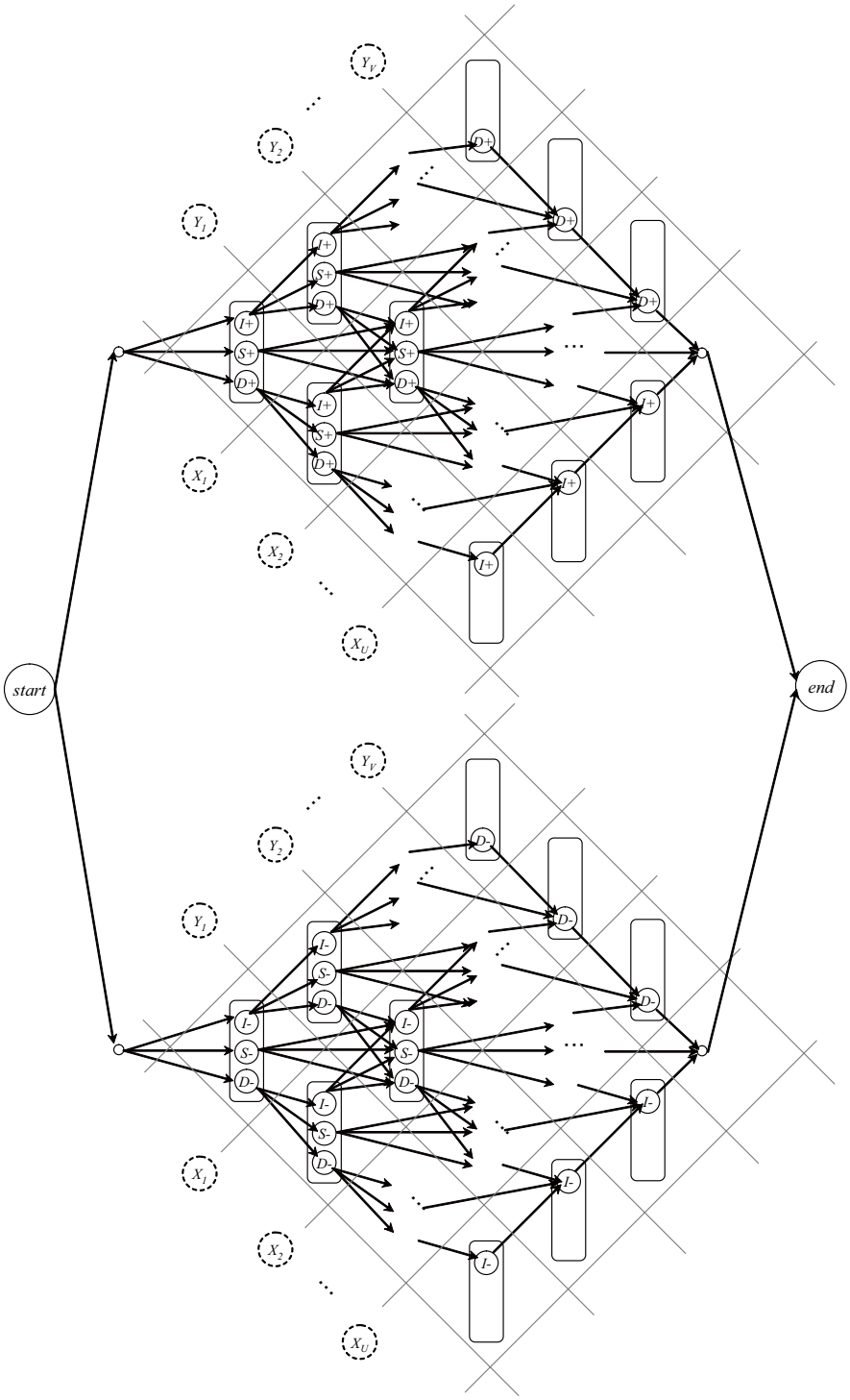


Figure 2.11: The trellis for a given sequence pair; the upper part is for matched, and the lower part is for mismatched.

The coordination identification

Given N independent instances $\{x_{seq}^{(n)}, conjunct\_regions^{(n)}\}_{n=1}^{N}$ as training data.

input: $x_{seq} = x_1 x_2 \ldots x_{S_n} \in W_{seq}$, $x_u \in W = \{\text{a set of tokens}\}$

output: *conjunct_regions*; regions are disjointed each other.

Fit a prediction function $\hat{f}_{pred} : X_{seq} \subseteq W_{seq} \rightarrow conjunct\_regions$ candidate set.

Figure 2.12: A formal statement of coordination identification

and, in some cases

- to detect conjunct pairs **without canonical word-by-word alignments**.

Because of the first one, our trellis have a shape of upper triangles (see Figure 4.2). The second leads to the necessity of separate label sets for matched and mismatched parts, like L_+ and L_-, respectively. However, unlike the task of match/mismatch determination, we allow transitions between states in the two sets, for the sake of identifying matched conjunct regions.

The third requirement means that there is a case where we can not map one input sequence pair into one label sequence. For example, "dose dense" and "standard" in Figure 2.1 align essentially as phrases, not by words. One solution is to determine a canonical label sequence by some predefined rules. Other one, in fact we shall use, is to consider all label sequences as canonical that are consistent with the phrase-by-phrase alignment. This situation has a common feature with the task of match/mismatch determination in that "a given sequence pair is similar (or dissimilar)" means that there are various canonical sequences in E_{seq+} (or E_{seq-}).

2.4 Maximum margin methods

In this section, we introduce two class SVMs that originated in Vapnik [59]. Then we give a brief description of the method based on SVMs that can deal with both sequence labeling and sequence alignment tasks [57, 2, 29, 60]. The method has something to do with our attempts to disambiguate coordinated phrases based on the perceptron algorithms, in that the goal is to find a hyperplane that separates **the difference of feature vectors** , that is, the difference between the feature vector involved in a correct label (or label sequence) and the one involved in an incorrect label (or label sequence) for each instance.

2.4.1 Two class SVMs for classification

Here we restrict the task to binary classification, so we fix the label set in Figure 2.2 as $L = \{+1, -1\}$. In the space of inputs $\boldsymbol{R}^D$, we consider a separating hyperplane with a D dimensional vector $\boldsymbol{W}$ and a scalar b as parameters,

$$\boldsymbol{W}^T \boldsymbol{x} + b = 0.$$

We define the margin of an instance $(\boldsymbol{x}^{(n)}, y^{(n)})$ from the hyperplane as

$$margin(\boldsymbol{x}^{(n)}, y^{(n)} | \boldsymbol{W}, b) = \frac{1}{\|\boldsymbol{W}\|} y^{(n)} (\boldsymbol{W}^T \boldsymbol{x}^{(n)} + b),$$

where $\|\boldsymbol{W}\| = \sqrt{\boldsymbol{W}^T \boldsymbol{W}}$. When the sign of $\boldsymbol{W}^T \boldsymbol{x}^{(n)} + b$ agrees with $y^{(n)}$, we say the hyperplane correctly classifies the instance, and the margin equals to the geometric distance of the instance from the hyperplane. We also define the margin with respect to the training data set $\{\boldsymbol{x}^{(n)}, y^{(n)}\}_{n=1}^N$, such that

$$MARGIN(\{\boldsymbol{x}^{(n)}, y^{(n)}\}_{n=1}^N | \boldsymbol{W}, b) = \min_{1 \leq n \leq N} margin(\boldsymbol{x}^{(n)}, y^{(n)} | \boldsymbol{W}, b). \tag{2.13}$$

We begin an explanation of two class SVMs using a simplest case, that is, there are only two instances, $(\boldsymbol{x}^{(1)}, y^{(1)} = +1)$ and $(\boldsymbol{x}^{(2)}, y^{(2)} = -1)$. The goal is to find the hyperplane, or to optimize $\boldsymbol{W}$ and b, that maximizes the *MARGIN*. Three hyperplanes that correctly classify the two instances are illustrated in Figure 2.13, in case $D = 2$. We see from the top and middle figure that the *MARGIN* is equivalent, in terms of the linear transformation of $\boldsymbol{W}$ and b (i.e. $\boldsymbol{W}_2 = 3\boldsymbol{W}_1$, and $b_2 = 3b_1$). Therefore, to get a unique solution, we impose the constraints

$$\begin{aligned} \boldsymbol{W}^T \boldsymbol{x}^{(1)} + b &= 1, \\ \boldsymbol{W}^T \boldsymbol{x}^{(2)} + b &= -1. \end{aligned}$$

Then Equation (2.13) reduces to $\frac{1}{\|\boldsymbol{W}\|}$, so maximizing the *MARGIN* becomes equivalent to minimizing $\boldsymbol{W}^T \boldsymbol{W}$ or $\|\boldsymbol{W}\|$. In fact, the bottom in Figure 2.13, that has larger $\|\boldsymbol{W}\|$ than the top, results in the worse *MARGIN*.

In the general case of N instances, if it is guaranteed that there is a hyperplane that correctly classifies them (this assumption is sometimes unrealistic), maximizing the *MARGIN* is also equivalent to minimizing $\boldsymbol{W}^T \boldsymbol{W}$ under the constraints that

$$y^{(n)} (\boldsymbol{W}^T \boldsymbol{x}^{(n)} + b) \geq 1, \; n = 1, \ldots, N, \tag{2.14}$$

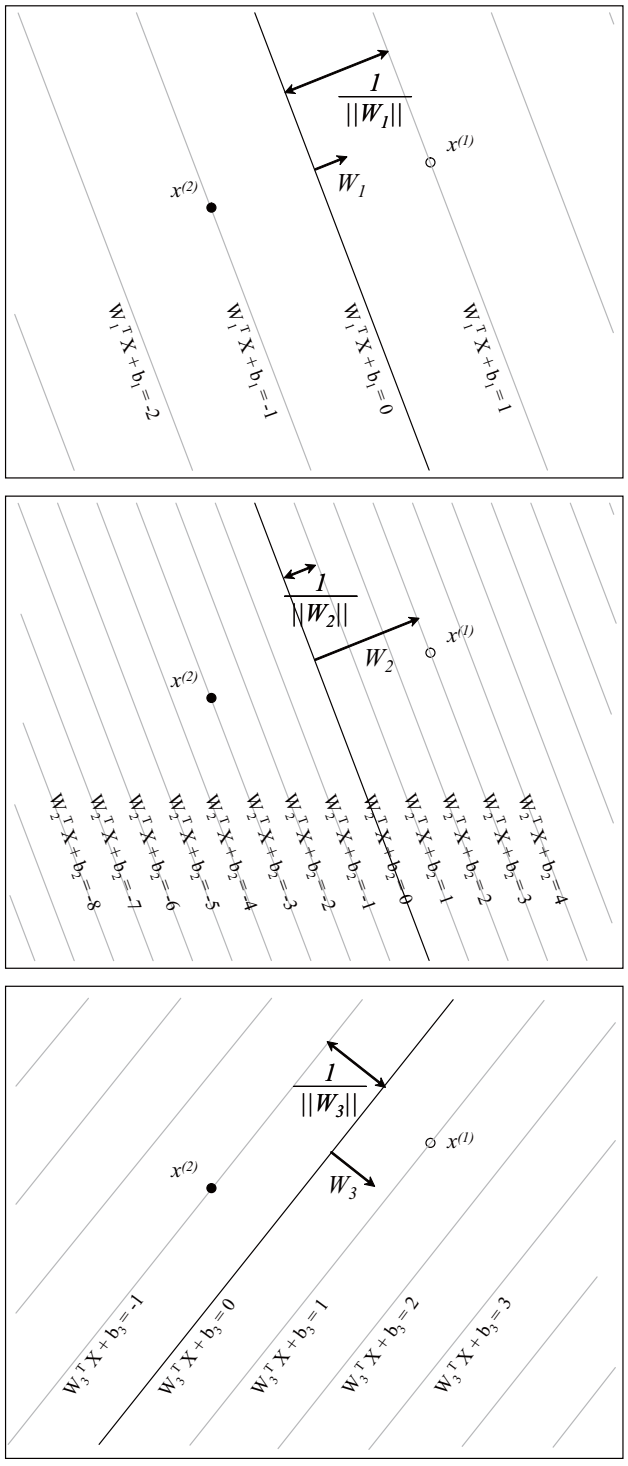


Figure 2.13: Separating hyperplanes on contour maps

the equalities satisfy when the instances are on (the border of) the margin. Using the Lagrangian theory, this constrained optimization problem can be expressed by the dual form, that is, to maximize

$$\sum_{n=1}^{N} a^{(n)} - \frac{1}{2} \sum_{n=1}^{N} \sum_{n'=1}^{N} a^{(n)} a^{(n')} y^{(n)} y^{(n')} \boldsymbol{x}^{(n)T} \boldsymbol{x}^{(n')},$$

with respect to $a^{(n)}$, $n = 1, \ldots, N$, subject to the constraints

$$\sum_{n=1}^{N} y^{(n)} a^{(n)} = 0, \tag{2.15}$$

$$a^{(n)} \geq 0, \; n = 1, \ldots, N, \tag{2.16}$$

and the following relation between parameters of the primal and dual forms ($\boldsymbol{W}$ and $a^{(n)}$, respectively) holds:

$$\boldsymbol{W} = \sum_{n=1}^{N} a^{(n)} y^{(n)} \boldsymbol{x}^{(n)}. \tag{2.17}$$

The optimization of the dual problem satisfies the Karush-Kuhn-Tucker conditions (see [17, 6] for details) that lead to an interesting property, such that, for each instance, $a_{opt}^{(n)} = 0$ or $y^{(n)}(\boldsymbol{W}_{opt}^T \boldsymbol{x}^{(n)} + b_{opt}) = 1$. In other words, only the instances that lie on the border of the margin contribute to decide the parameter $\boldsymbol{W}$ of hyperplanes through Equation (2.17). As a result, we get a prediction function

$$\begin{aligned} \hat{f}_{pred}(\boldsymbol{x}^{(test)}) &= sign(\boldsymbol{W}_{opt}^T \boldsymbol{x}^{(test)} + b_{opt}) \\ &= sign(\sum_{n=1}^{N} a_{opt}^{(n)} y^{(n)} \boldsymbol{x}^{(n)T} \boldsymbol{x}^{(test)} + b_{opt}). \end{aligned}$$

Next we consider the more realistic situation where there is no hyperplane that classifies N instances. The solution in [59] is to introduce a slack variable $\xi^{(n)}$ for each instance, and to relax the constraints (2.14) as

$$y^{(n)}(\boldsymbol{W}^T \boldsymbol{x}^{(n)} + b) \geq 1 - \xi^{(n)}, \; n = 1, \ldots, N.$$

Using a trade-off parameter C that controls the slack variable penalty and margin, the objective function to minimize is, for the 1-norm soft margin SVMs,

$$\boldsymbol{W}^T \boldsymbol{W} + C \sum_{n=1}^{N} \xi^{(n)},$$

and for the 2-norm soft margin SVMs,

$$\boldsymbol{W}^T\boldsymbol{W} + C\sum_{n=1}^{N}\xi^{(n)2}.$$

Although the original in [59] is the former, here we focus on the 2-norm soft margin SVMs because they relate Freund and Schapire's augmented space methods [21] that we shall incorporate to our proposing method. Using the Lagrangian theory, the dual objective function becomes

$$\sum_{n=1}^{N} a^{(n)} - \frac{1}{2}\sum_{n=1}^{N}\sum_{n'=1}^{N} a^{(n)}a^{(n')}y^{(n)}y^{(n')}(\boldsymbol{x}^{(n)T}\boldsymbol{x}^{(n')} - \frac{1}{C}\delta_{nn'}),$$

where $\delta_{nn'}$ is the Kronecker δ defined to be 1 if $n = n'$ and 0 otherwise. The problem turns to maximizing this objective function with respect to $a^{(n)}$, $n = 1,\ldots,N$, subject to the same constraints as (2.15) and (2.16). In addition to Equation (2.17), it holds that

$$a^{(n)} = C\xi^{(n)},\ n = 1,\ldots,N. \tag{2.18}$$

For each instance, due to the Karush-Kuhn-Tucker conditions, $a^{(n)}_{opt} = 0$ or $y^{(n)}(\boldsymbol{W}^T_{opt}\boldsymbol{x}^{(n)} + b_{opt}) = 1 - \xi^{(n)}$ holds. Together with Equation (2.18), we see that only the instances that lie inside the margin (not on the border) contribute to decide the parameter $\boldsymbol{W}$ of hyperplanes through Equation (2.17).

It is important to note that the 2-norm soft margin SVMs are equivalent to the original SVMs (i.e. without slack variables) when we convert D dimensional vectors $\boldsymbol{x}^{(n)}$ to $D+N$ dimensional vectors $\tilde{\boldsymbol{x}}^{(n)}$, such that for $n = 1,\ldots,N$,

$$\tilde{\boldsymbol{x}}^{(n)T} = (\boldsymbol{x}^{(n)T}, 0,\ldots,0,\tfrac{1}{\sqrt{C}},0,\ldots,0)^T, \tag{2.19}$$

in which the first D dimensions are identical to $x^{(n)}$, and for the remaining N dimensions, $\sqrt{C}$ is assigned to the $(D+n)$th element, and 0 for the rest. Freund and Schapire used this conversion in [21] to give the mistake bound of the online perceptron algorithm in the inseparable case. It is also called "lambda trick" according to [38]. We

can confirm the equivalence by the calculation as follows:

$$\begin{aligned}
\tilde{\boldsymbol{W}}^T\tilde{\boldsymbol{W}} &= (\sum_{n=1}^{N} a^{(n)}y^{(n)}\tilde{\boldsymbol{x}}^{(n)})^T(\sum_{n'=1}^{N} a^{(n')}y^{(n')}\tilde{\boldsymbol{x}}^{(n')}) \\
&= \sum_{n=1}^{N}\sum_{n'=1}^{N} a^{(n)}a^{(n')}y^{(n)}y^{(n')}\tilde{\boldsymbol{x}}^{(n)T}\tilde{\boldsymbol{x}}^{(n')} \\
&= \sum_{n=1}^{N}\sum_{n'=1}^{N} a^{(n)}a^{(n')}y^{(n)}y^{(n')}(\boldsymbol{x}^{(n)T}\boldsymbol{x}^{(n')} + \frac{1}{C}\delta_{nn'}) \\
&= \sum_{n=1}^{N}\sum_{n'=1}^{N} a^{(n)}a^{(n')}y^{(n)}y^{(n')}\boldsymbol{x}^{(n)T}\boldsymbol{x}^{(n')} + \frac{1}{C}\sum_{n=1}^{N} a^{(n)2}y^{(n)2} \\
&= \sum_{n=1}^{N}\sum_{n'=1}^{N} a^{(n)}a^{(n')}y^{(n)}y^{(n')}\boldsymbol{x}^{(n)T}\boldsymbol{x}^{(n')} + \frac{1}{C}\sum_{n=1}^{N} a^{(n)2} \\
&= \boldsymbol{W}^T\boldsymbol{W} + C\sum_{n=1}^{N}\xi^{(n)2},
\end{aligned}$$

$$\begin{aligned}
y^{(n)}(\tilde{\boldsymbol{W}}^T\tilde{\boldsymbol{x}}^{(n)} + b) &= y^{(n)}\tilde{\boldsymbol{W}}^T\tilde{\boldsymbol{x}}^{(n)} + y^{(n)}b \\
&= y^{(n)}\sum_{n'=1}^{N} a^{(n')}y^{(n')}\tilde{\boldsymbol{x}}^{(n')T}\tilde{\boldsymbol{x}}^{(n)} + y^{(n)}b \\
&= y^{(n)}\sum_{n'=1}^{N} a^{(n')}y^{(n')}(\boldsymbol{x}^{(n')T}\boldsymbol{x}^{(n)} + \frac{1}{C}\delta_{nn'}) + y^{(n)}b \\
&= y^{(n)}\sum_{n'=1}^{N} a^{(n')}y^{(n')}\boldsymbol{x}^{(n')T}\boldsymbol{x}^{(n)} + y^{(n)}\sum_{n'=1}^{N} a^{(n')}y^{(n')}\frac{1}{C}\delta_{nn'} + y^{(n)}b \\
&= y^{(n)}\boldsymbol{W}^T\boldsymbol{x}^{(n)} + y^{(n)}a^{(n)}y^{(n)}\frac{1}{C} + y^{(n)}b \\
&= y^{(n)}\boldsymbol{W}^T\boldsymbol{x}^{(n)} + a^{(n)}\frac{1}{C} + y^{(n)}b \\
&= y^{(n)}\boldsymbol{W}^T\boldsymbol{x}^{(n)} + \xi^{(n)} + y^{(n)}b \\
&= y^{(n)}(\boldsymbol{W}^T\boldsymbol{x}^{(n)} + b) + \xi^{(n)}.
\end{aligned}$$

As a final note, we put a comment on the multi class SVMs. Roughly speaking, there are two strategies. One is to make use of two class SVMs, that is, one versus the rest, pairwise classification, and error-correcting output coding. The other is to solve problems directly, in considering multi class objective functions. According to [50], there is no approach that generally outperforms the others.

2.4.2 SVMs for sequence labeling and sequence alignment

Here we give a short introduction about the work in [57] that can deal with discriminative problems whose outputs have complex structures. We shall describe the method in the case of sequence labeling below, however, it can be adopted to sequence alignment in the same way.

Under the first order Markov assumption on the label sequences, we employ the same feature functions (2.5) and (2.6) described in the section of linear chain CRFs. Then for each instance $(x_{seq}^{(n)}, y_{seq}^{(n)})$, using Equation (2.8), calculate the difference between the score of correct path and the maximum score of incorrect path on the trellis, given as

$$\gamma^{(n)} = PathScore(x_{seq}^{(n)}, y_{seq}^{(n)}) - \underset{y_{seq} \in L_{seq} \setminus y_{seq}^{(n)}}{\operatorname{argmax}} PathScore(x_{seq}^{(n)}, y_{seq}). \tag{2.20}$$

The maximum margin principle gives the constrained optimization problem, whose optimization algorithm is a highlight of [57] (but, we omit it here), as follows:

$$\text{minimize } \Theta^T \Theta, \text{ subject to } \gamma^{(n)} \geq 1,\ n = 1, \ldots, N,$$

where Θ is the parameter vector defined in (2.7). In this place, we define the global feature vector as

$$\boldsymbol{F}(x_{seq}, y_{seq}) = (\boldsymbol{f}_1^T, \ldots, \boldsymbol{f}_K^T, \boldsymbol{f}_{11}^T, \ldots, \boldsymbol{f}_{KK}^T)^T. \tag{2.21}$$

Then we can rewrite (2.20) to be

$$\gamma^{(n)} = \underset{y_{seq} \in L_{seq} \setminus y_{seq}^{(n)}}{\operatorname{argmax}} \Theta^T (\boldsymbol{F}(x_{seq}^{(n)}, y_{seq}^{(n)}) - \boldsymbol{F}(x_{seq}^{(n)}, y_{seq})).$$

From this representation, we see that feature vectors associated with individual label sequence (correct or incorrect) are not important; what is important is the difference of those.

2.5 The perceptron algorithms

In this section, we review the original perceptron algorithms [47] for two class problems. Then we explain how to extend the original to the ones for sequence labeling and sequence alignment tasks, showing a comparison with the methods based on CRFs.

2.5.1 Two class perceptron algorithms

We consider the same situation as in Section 2.4.1 for two class SVMs, that is, the goal is to find a hyperplane

$$\boldsymbol{W}^T\boldsymbol{x} + b = 0, \tag{2.22}$$

on the D dimensional space which correctly classify N training instances. For the moment, we do not think about *MARGIN* (defined in (2.13)), in this point, the situation differs from SVMs. For the ease of description, we incorporate the bias term b into $\boldsymbol{W}$, and add the corresponding component, that is, 1, into each input vector $\boldsymbol{x}$. Then, the above hyperplane (2.22) becomes

$$\boldsymbol{W}^T\boldsymbol{x} = 0$$

on the $D+1$ dimensional space.

Once a hyperplane is given, we can enumerate instances that are not correctly classified by the hyperplane, and we define the *Loss* given by

$$Loss = \sum_{n=1}^{N} -y^{(n)}\boldsymbol{W}^T\boldsymbol{x}^{(n)}\ I(y^{(n)}\boldsymbol{W}^T\boldsymbol{x}^{(n)} < 0), \tag{2.23}$$

where $I(\cdot)$ is the indicator function, and the condition $y^{(n)}\boldsymbol{W}^T\boldsymbol{x}^{(n)} < 0$ satisfies when the instance is misclassified by the hyperplane determined by $\boldsymbol{W}$. Then the derivative becomes

$$\frac{\partial Loss}{\partial \boldsymbol{W}} = \sum_{n=1}^{N} -y^{(n)}\boldsymbol{x}^{(n)}\ I(y^{(n)}\boldsymbol{W}^T\boldsymbol{x}^{(n)} < 0),$$

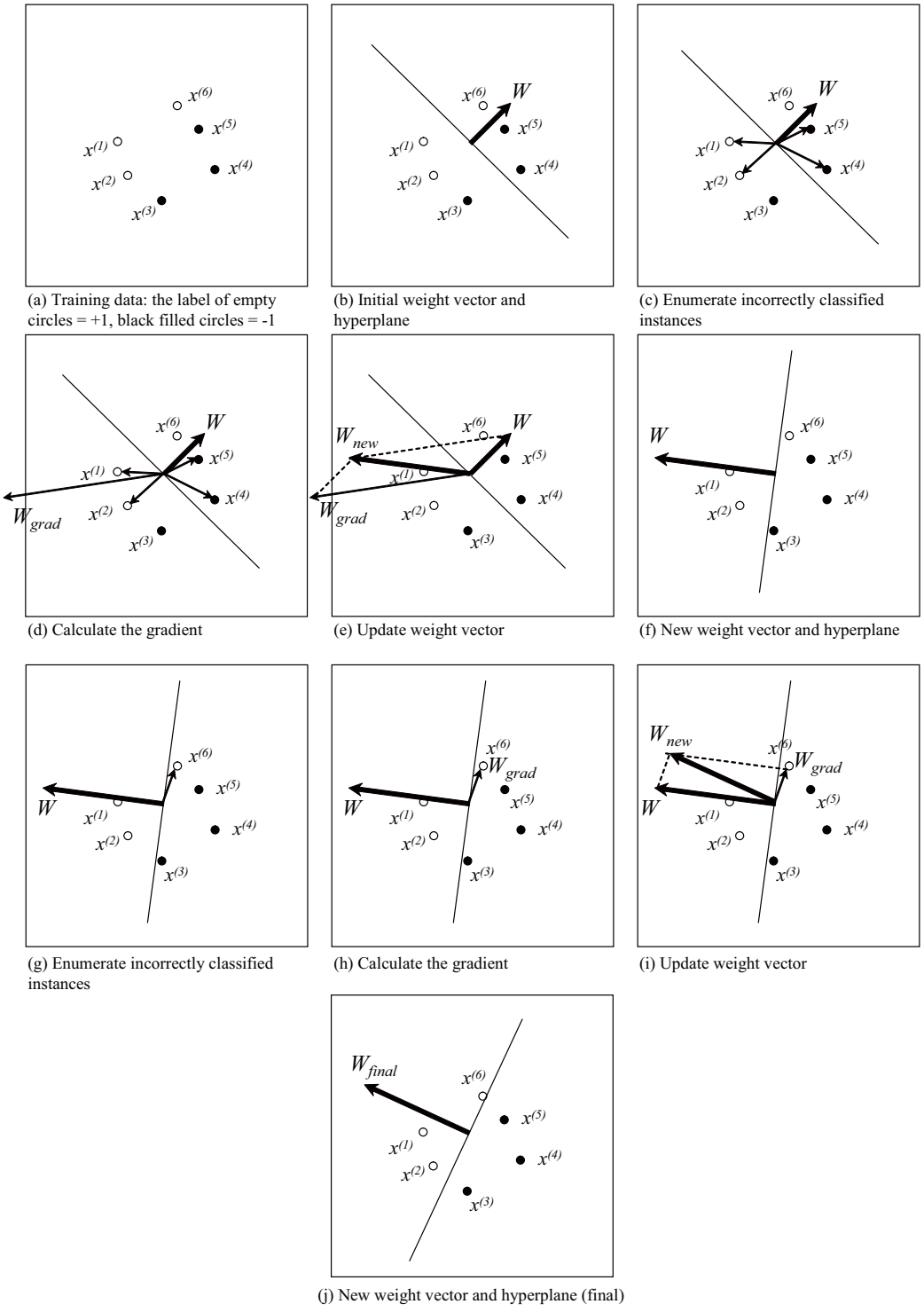
therefore, the batch perceptron algorithms update $\boldsymbol{W}$ in accordance with the gradient, such that

$$\boldsymbol{W}_{new} \leftarrow \boldsymbol{W} + \sum_{n=1}^{N} y^{(n)}\boldsymbol{x}^{(n)}\ I(y^{(n)}\boldsymbol{W}^T\boldsymbol{x}^{(n)} < 0),$$

whereas the online perceptron algorithms use the stochastic gradient descent that update weights for each misclassified instance, that is

$$\boldsymbol{W}_{new} \leftarrow \boldsymbol{W} + y^{(n)}\boldsymbol{x}^{(n)}, \text{ if } y^{(n)}\boldsymbol{W}^T\boldsymbol{x}^{(n)} < 0 \text{ satisfies.} \tag{2.24}$$

The algorithms sweep training data iteratively until convergence, but the convergence is guaranteed only when the linearly separable case. So in general we stop the learning process by consulting the development data, or just after sufficient number of iterations. It is also important to note that the result depends on the initial parameter values, and the order of training instances (in case of online learning). An illustration of batch perceptron learning is shown in Figure 2.14.



(a) Training data: the label of empty circles = +1, black filled circles = -1

(b) Initial weight vector and hyperplane

(c) Enumerate incorrectly classified instances

(d) Calculate the gradient

(e) Update weight vector

(f) New weight vector and hyperplane

(g) Enumerate incorrectly classified instances

(h) Calculate the gradient

(i) Update weight vector

(j) New weight vector and hyperplane (final)

Figure 2.14: An illustration of batch perceptron learning

2.5.2 The perceptron algorithms for sequence labeling and sequence alignment

As we described above, the perceptron algorithms updates parameters when the classification fails for training instances. For the task of sequence labeling (the same reasoning as the following is applicable to sequence alignment task), the condition for occurring classification error is represented as, by using the global feature vector in (2.21),

$$\underset{y_{seq} \in L_{seq} \setminus y_{seq}^{(n)}}{\operatorname{argmax}} \Theta^T (\boldsymbol{F}(x_{seq}^{(n)}, y_{seq}^{(n)}) - \boldsymbol{F}(x_{seq}^{(n)}, y_{seq})) < 0.$$

Under the condition for each instance, parameter update (in case of batch learning) is executed as follows:

$$\Theta_{new} = \Theta + \sum_{n=1}^{N} (\boldsymbol{F}(x_{seq}^{(n)}, y_{seq}^{(n)}) - \boldsymbol{F}(x_{seq}^{(n)}, \hat{y}_{seq})), \tag{2.25}$$

where

$$\hat{y}_{seq} = \underset{y_{seq} \in L_{seq} \setminus y_{seq}^{(n)}}{\operatorname{argmax}} \Theta^T \boldsymbol{F}(x_{seq}^{(n)}, y_{seq}).$$

The search for $\hat{y}_{seq}$ can be carried out efficiently using dynamic programming algorithms on the trellis.

We importantly note that the way of perceptron's parameter update is closely related to CRFs. In CRFs, the gradient is represented using the global feature vector, such that from (2.9) and (2.10),

$$\begin{aligned} \boldsymbol{W}_{grad} &= \sum_{n=1}^{N} (\boldsymbol{F}(x_{seq}^{(n)}, y_{seq}^{(n)}) - \sum_{y_{seq} \in L_{seq}} \boldsymbol{F}(x_{seq}^{(n)}, y_{seq}) Prob(y_{seq} | x_{seq}^{(n)}, \Theta)) \\ &= \sum_{n=1}^{N} \left(\boldsymbol{F}(x_{seq}^{(n)}, y_{seq}^{(n)}) - \sum_{y_{seq} \in L_{seq}} \boldsymbol{F}(x_{seq}^{(n)}, y_{seq}) \frac{\Theta^T \boldsymbol{F}(x_{seq}^{(n)}, y_{seq})}{\sum_{y'_{seq} \in L_{seq}} \Theta^T \boldsymbol{F}(x_{seq}^{(n)}, y'_{seq})} \right). \end{aligned}$$

In comparison with (2.25), we see that CRFs use all candidate paths on the trellis in averaging, on the other hand, the perceptron algorithms update parameters by using only a most probable candidate path on the trellis.

2.6 Variants of the perceptron algorithms

Here, we describe some strategies to generalize perceptron algorithms to reduce the risk of over-fitting to training data, which include ensemble methods such as the

averaged perceptron and the Bayes point machines. In Chapter 4 we shall actually employ those for coordination disambiguation, and see we get superior results by using averaged perceptron.

2.6.1 Regularization

Learning model parameters (or weights) from training data is likely to be suffered from the over-fitting problem, particularly when the dimensionality of the weight vector is considerably larger than the size of the data. To avoid the problem and achieve higher performance in test data, one way, so-called regularization, is to add a penalty term respecting the norm of weight vector to the loss function.

Given the constant C to control the magnitude of the penalty, the *Loss* (2.23) turns to be, in case of the L1-norm

$$Loss = \sum_{n=1}^{N} -y^{(n)} \boldsymbol{W}^T \boldsymbol{x}^{(n)} \; I(y^{(n)} \boldsymbol{W}^T \boldsymbol{x}^{(n)} < 0) \; + \; C \sum_{d=1}^{D} |W_d|,$$

and for the L2-norm,

$$Loss = \sum_{n=1}^{N} -y^{(n)} \boldsymbol{W}^T \boldsymbol{x}^{(n)} \; I(y^{(n)} \boldsymbol{W}^T \boldsymbol{x}^{(n)} < 0) \; + \; \frac{C}{2} \sum_{d=1}^{D} W_d^2.$$

Then, the gradient of the dth element of the weight vector becomes, for the L1-norm,

$$\frac{\partial Loss}{\partial W_d} = \sum_{n=1}^{N} -y^{(n)} x_d^{(n)} \; I(y^{(n)} \boldsymbol{W}^T \boldsymbol{x}^{(n)} < 0) \; + \; C \; sign(W_d), \tag{2.26}$$

for the L2-norm,

$$\frac{\partial Loss}{\partial W_d} = \sum_{n=1}^{N} -y^{(n)} x_d^{(n)} \; I(y^{(n)} \boldsymbol{W}^T \boldsymbol{x}^{(n)} < 0) \; + \; C \; W_d, \tag{2.27}$$

and we update the weight vector accordingly.

2.6.2 Lambda trick

We have seen in Section 2.4.1 that the transformation of input $\boldsymbol{x}^{(n)}$ to $\tilde{\boldsymbol{x}}^{(n)}$ by (2.19), which appears when considering 2-norm soft margin SVMs, reduces an inseparable

problem to a separable one. Applying the transformation ("lambda trick") to the perceptron algorithm, the condition for updating the weights changes to be

$$y^{(n)}\tilde{\boldsymbol{W}}^T\tilde{\boldsymbol{x}}^{(n)} = y^{(n)}\boldsymbol{W}^T x^{(n)} < -\frac{ErrorTimes^{(n)}}{C}, \tag{2.28}$$

where $\tilde{\boldsymbol{W}}$ is also $(D+N)$ dimensional vector, $ErrorTimes^{(n)}$ is the number of times the nth instance has been misclassified before, and $\tilde{\boldsymbol{W}}$ memorizes $ErrorTimes^{(n)}$ (multiplied by $\frac{1}{\sqrt{C}}$) in the $(D+n)$th element. This means that the algorithm permits misclassification to the extent that the significance of the error (i.e. $y^{(n)}\boldsymbol{W}^T x^{(n)}$) is tolerable in terms of the frequency of mistakes for the instance.

2.6.3 MIRA; online perceptron taking the margin into consideration

In the iterative procedure of the (ordinary) online perceptron algorithm, the weight vector is updated according to (2.24) for the nth instance. There the coefficient of the term $y^{(n)}\boldsymbol{x}^{(n)}$, called as learning rate, is fixed to unity. The MIRA algorithm [15] changes the coefficient to τ, given by

$$\tau = G\Big(-\frac{y^{(n)}\boldsymbol{W}^T\boldsymbol{x}^{(n)} - 1}{\boldsymbol{x}^{(n)T}\boldsymbol{x}^{(n)}}\Big), \tag{2.29}$$

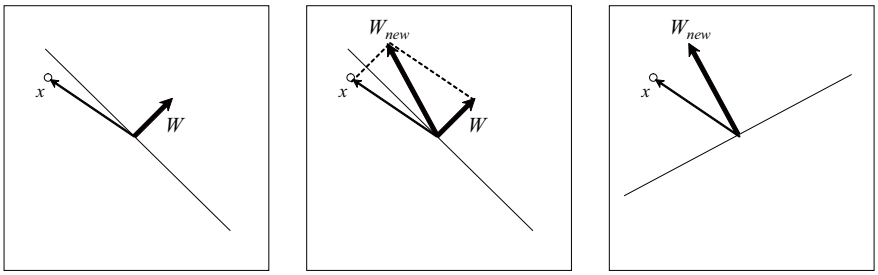
where

$$G(z) = \begin{cases} 0 & \text{if } z < 0 \\ z & \text{if } 0 \le z \le 1 \\ 1 & \text{if } 1 < z. \end{cases}$$

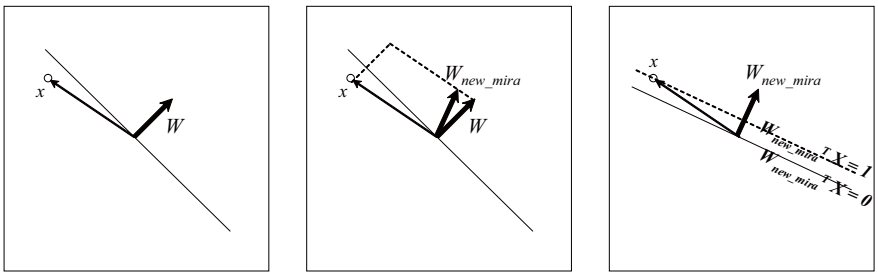
In short,

$$\boldsymbol{W}_{new\_mira} \leftarrow \boldsymbol{W} + \tau\, y^{(n)}\boldsymbol{x}^{(n)}, \text{ if } y^{(n)}\boldsymbol{W}^T\boldsymbol{x}^{(n)} < 0 \text{ satisfies.} \tag{2.30}$$

We visualize in Figure 2.15 the effect of using τ in comparison with the ordinary online perceptron. The left column in the figure indicates that the focusing instance has the positive label ($y = +1$) but resides in the negative region on account of the hyperplane determined by $\boldsymbol{W}$. The mid column shows the weight update by (2.24) for the ordinary perceptron and by (2.30) for the MIRA algorithm. The new hyperplane obtained by the ordinary update brings the instance deep inside the positive region, on the other hand, the instance is moved to at most the margin of the new hyperplane by the MIRA algorithm, as shown in the right column.



(a) Online perceptron ; $\boldsymbol{W_{new}} = \boldsymbol{W} + \boldsymbol{x}$



(b) MIRA ; $\boldsymbol{W_{new\_mira}} = \boldsymbol{W} + \tau\,\boldsymbol{x}$

Figure 2.15: Comparison of the ways to update the weights between the ordinary online perceptron and the MIRA algorithm

Table 2.3: Averaged perceptron; weights for testing $= \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{W}_t$

iteration index t	weights after t iteration
1	$\boldsymbol{W}_1$
2	$\boldsymbol{W}_2$
$\vdots$	$\vdots$
T	$\boldsymbol{W}_T$

Table 2.4: Bayes point machines; weights for testing $= \frac{1}{M}\sum_{m=1}^{M} \boldsymbol{W}_{mT}$

	weights after t iteration			
iteration index t	system 1	system 2	...	system M
1	$\boldsymbol{W}_{11}$	$\boldsymbol{W}_{21}$	...	$\boldsymbol{W}_{M1}$
2	$\boldsymbol{W}_{12}$	$\boldsymbol{W}_{22}$	...	$\boldsymbol{W}_{M2}$
$\vdots$	$\vdots$	$\vdots$		$\vdots$
T	$\boldsymbol{W}_{1T}$	$\boldsymbol{W}_{2T}$	...	$\boldsymbol{W}_{MT}$

2.6.4 Averaged perceptron

Our proposing method for coordination disambiguation from which we obtain the best result we shall present in Chapter 4, is based on the averaged perceptron originated in [22]. The idea is quite simple. In the training phase, to keep in the stack all updated weight vectors, and to use their average in the test phase (see Table 2.3). The implementation is uncomplicated and the discriminative power, in applying to the sequence labeling task, is known to be competitive to (or, sometimes greater than) the methods representing the state-of-the-art, according to [12, 3].

2.6.5 Bayes point machines

In addition to the averaged perceptron, there is another ensemble technique known as the Bayes point machines [25]. In the learning phase, the method trains multiple perceptron classifiers in parallel, which deal with the same training data set but are different in the ways to use individual instances in choice (for both the batch and online algorithms) or order (for the online algorithm). In Table 2.4, we illustrate the method having M perceptron classifiers. In contrast to the averaged perceptron that averages weights over iteration indexes, Bayes point machines use averaged weights over classifiers for testing.

Chapter 3

Extracting Clinical Trial Design Information from MEDLINE Abstracts

3.1 Introduction

Recently, people who practice medical treatments have been paying more attention to Evidence-Based Medicine (EBM), the concept of which is defined as: "the conscientious, explicit, and judicious use of current best evidence in making decisions about the care of individual patients" [48]. In practicing EBM, they are encouraged to be well informed on up-to-date sources of medical knowledge such as MEDLINE, the US National Library of Medicine's bibliographic database covering the fields of medical, pharmaceutical and biological sciences.

Among MEDLINE abstracts, those about clinical trials play one of the most important roles in EBM, because the results of clinical trials can provide firm evidence to support applying a certain therapy in actual medical treatments. However, since the rate at which new articles are being introduced into the MEDLINE database is fairly high, it takes patients or doctors who seek beneficial knowledge quite some time to read all of the articles that may contain clues in finding a suitable therapy. So, in order to assist members of the medical community, our goal is to extract important information from MEDLINE clinical trial abstracts, in an effort to reduce the amount of time required to find relevant medical information.

In the research field of natural language processing (NLP), the task of information extraction (IE) has been pursued with a great deal of interest for decades. For example,

in the series of Message Understanding Conferences (MUCs), participants developed methods for extracting information of the scenario-templates presented by the conference organizers. The focus of the study there was the construction of domain-specific lexicons and extraction patterns based on human labor. Following the MUCs, the attention of researchers has shifted to automatic knowledge acquisition including lexicons and patterns.

In this thesis, we report experimental results of extracting information about clinical trial design from the abstracts of phase III clinical trials, to investigate how far the existing NLP techniques could support EBM on the use of MEDLINE database.

3.2 Background and Objectives

3.2.1 Phase III clinical trials: directly linking to EBM

The broad aim of clinical trials is to demonstrate the efficacy and safety of the new or focusing treatments. In the development of a new drug, for example, clinical trials are conducted in a stepwise process such as the one consisting of phase I, to evaluate the safety and tolerability in healthy volunteers, phase II, to explore the dose response relationship in patient populations, and phase III, to confirm the efficacy and safety compared with an active control or placebo in patient populations. Among those steps, phase III clinical trials are most closely linked to EBM, in other words, hypotheses of superiority to an active control or placebo is always stated before starting the enrollment of patients, and statistical hypothesis testing is conducted after the completion of the trials [42]. Such a way protects the statistical test results against bias affection, so the conclusion obtained from phase III confirmatory analyses can be considered as a firm evidence of a new drug administration, in case its superiority is proven. In contrast, exploratory analyses such that involve stratified or subgroup analyses don't declare any hypotheses in advance.

3.2.2 MEDLINE abstracts: available resource for EBM

The bibliographic information stored in MEDLINE database is surely the resource for EBM. However the articles relevant to clinical trials are just a small part of the database. For selecting those, MEDLINE users can utilize PubMed (http://pubmed.gov),

Table 3.1: The number of abstracts registered in MEDLINE

PubMed search query	# of abstracts
"hasabstract"	8,120,830
"hasabstract AND Clinical Trial[PT]"	352,576
"hasabstract AND Clinical Trial, Phase I[PT]"	6,548
"hasabstract AND Clinical Trial, Phase II[PT]"	10,013
"hasabstract AND Clinical Trial, Phase III[PT]"	2,789
"hasabstract AND Clinical Trial, Phase IV[PT]"	243
"hasabstract AND Neoplasms[MT] AND Clinical Trial[PT]"	52,094
"hasabstract AND Neoplasms[MT] AND Clinical Trial, Phase III[PT]"	1,528

Table 3.2: Examples of IE targets

PMID	Patient Population	Compared Treatments
16234567	hepatocellular carcinoma	doxorubicin versus cisplatin, interferon alpha-2b, doxorubicin, and fluorouracil
16192591	metastatic breast cancer	doxorubicin and docetaxel versus fluorouracil, doxorubicin, and cyclophosphamide
16192580	malignant pleural mesothelioma	cisplatin versus raltitrexed and cisplatin

the searchable interface of MEDLINE on the Web. As search queries, users can specify not only plain text keywords but also categories annotated on each abstract. By December 2005, a total of 8,120,830 abstracts have been registered in MEDLINE with approximately 4.3% are categorized as the class of "Clinical Trial [Publication Type]", and 2,789 abstracts are annotated as "Clinical Trial, Phase III [Publication Type]" (see Table 3.1 [1]). One of the reasons that only a small number of abstracts are categorized as phase III might be that the sub-categorization of "Clinical Trial [Publication Type]" to phase I, II, III and IV has started very recently.

[1] "hasabstract" in the search query specifies articles that have abstracts. "PT" and "MH" are abbreviations of "Publication Type" and "MeSH Terms", respectively.

Table 3.3: Categories for base NPs

Class label	Covered concept	Example
Disease	disease, symptom, pain, complication	"metastatic breast cancer"
Treatment	drug, placebo, therapy, surgery	"doxorubicin"
Patient	participants in clinical trials	"patients"
Study	clinical trial	"a randomized controlled trial"
Others	other than the above	"the efficacy and safety"

3.2.3 Information extraction targets and data in this thesis

In this thesis, we report results of experiment in extracting "Patient Population" and "Compared Treatments" as the two most important clinical trial design information from phase III abstracts.[2] Examples of IE targets in abstracts are shown in Table 3.2.[3] Other information such as the endpoints or the number of patients enrolled in the clinical trials is also crucial for EBM but the former two information best characterizes the clinical trials, as seen in the patient recruiting site (http://clinicaltrials.gov) managed by the same organization as MEDLINE database.

The data we use in this thesis are the most recent 200 out of 1,528 abstracts labeled as both "Neoplasms [MeSH Terms]" and "Clinical Trial, Phase III [Publication Type]", on December 2005. The reasons for having added neoplasms to the search query are that more than half of 2,789 available phase III abstracts are annotated as "Neoplasms [MeSH Terms]", that the reduction of vocabulary variety caused by single disease type filtering will make the interpretation of our experimental results simple and clear, and that neoplasms are one of the ever-threatening diseases against humanity.

3.3 Information extraction applied to phase III abstracts

In this chapter, we describe the methods and results of the IE experiment using conventional NLP techniques, which consist of two parts: base NP (noun phrase) chunking and its categorization, and regular expression pattern matching. The process is illustrated in Figure 3.1.

[2]Strictly speaking, the term "clinical trial design" stands for the methodology to avoid bias, such as blinding or randomization. Here, however, we use the term in a broader sense.

[3]"PMID" is the PubMed index of a MEDLINE abstract.

1. Original text (an example)
    - Phase II to III study comparing docetaxel with fluorouracil in patients with metastatic breast cancer.
2. Base NP chunking
    - [Phase II to III study] comparing [docetaxel] with [fluorouracil] in [patients] with [metastatic breast cancer].
3. Base NP categorization
    - **Study** comparing **Treatment** with **Treatment** in **Patient** with **Disease**.
4. IE by pattern matching
    - "Patient Population": metastatic breast cancer
      (pattern matched with: **/Patient** with **Disease/**)
    - "Compared Treatments": docetaxel versus fluorouracil
      (pattern matched with: /compar.* **Treatment** .* **Treatment/**)

Figure 3.1: The IE process in this thesis

3.3.1 Base noun phrase chunking and its categorization

In the chunking step, our goal is the determination of base NPs in original texts. A base NP is defined as the shortest unit of noun phrase. For example in Figure 3.1, "patients with metastatic breast cancer" is an NP but not a base NP, because it could be divided into two base NPs of "patients" and "metastatic breast cancer". Next in the categorization step, we attach a class label to each base NP. The class labels we set up are shown in Table 3.3. After the process of chunking and categorization, original texts are transformed into a set of simpler sentences, which are amiable to the following process of pattern matching.

3.3.2 Methods and Results of automatic base noun phrase chunking and categorization

We manually annotated the 200 abstracts with correct answers of chunking and categorization, divided them into training and test sets, and then used machine learning techniques based on support vector machines (SVM) or conditional random fields

Table 3.4: Results of base NP chunking

chunker	answer	system	share	recall	precision
SVM (YamCha)	17,990	18,282	16,572	92.1%	90.6%
CRF (CRF++)	17,990	18,156	16,466	91.5%	90.7%

Table 3.5: Results of base NP categorization

classifier	class	length	answer	system	share	recall	precision
SVM (YamCha)	Disease	2.7	1,332	1,051	909	68.2%	86.5%
	Treatment	2.0	4,198	3,900	3,423	81.5%	87.8%
	Patient	1.9	1,309	1,175	1,150	87.9%	97.9%
	Study	4.2	491	443	412	83.9%	93.0%
	Others	2.1	10,660	11,421	10,131	95.0%	88.7%
	total	2.2	17,990	17,990	16,025	89.1%	89.1%
CRF (CRF++)	Disease	2.7	1,332	981	921	69.1%	93.9%
	Treatment	2.0	4,198	3,914	3,508	83.6%	89.6%
	Patient	1.9	1,309	1,216	1,201	91.7%	98.8%
	Study	4.2	491	462	450	91.6%	97.4%
	Others	2.1	10,660	11,409	10,292	96.5%	90.2%
	total	2.2	17,990	17,982	16,372	91.0%	91.0%

(CRF). Among available NLP software tools, we employed YamCha [4] and CRF++ [5] for SVM and CRF, respectively.

Features we used are words and part-of-speeches (window size = 5; 2 previous, 2 next), some of their combinations, and chunking results in case of category determination, both on the use of SVM and CRF. As of training data in chunk determination, we also used the sections 15 to 18 of the wall street journal part of the Penn Treebank corpus [6], which are widely used for the NP chunking tasks.

The results of 10-fold cross validation of base NP chunking and categorization are shown in Table 3.4 and Table 3.5, respectively.[7] In the results of chunk determina-

[4] http://www.chasen.org/taku/software/yamcha

[5] http://www.chasen.org/taku/software/CRF++

[6] http://www.cis.upenn.edu/treebank

[7] In the result tables, "answer": # of hand-labeled answers, "system": # in system output, "share": # shared in answer and system, "recall": percentage of share in answer, "precision": percentage of share in system, and "length": average word length of base NPs.

Table 3.6: Regular expression patterns for IE

IE target	Regular expression pattern
Patient Population	/Patient with Disease/ /Treatment (for|of|in) Disease/
Compared Treatments	/compar.* Treatment .* Treatment/ /Treatment.* (versus|vs|or|compared with) .*Treatment/

tion, SVM and CRF show almost the same performance. Compared with the best NP chunking results [31] using the standard data set (CoNLL 2000; sections 15 to 18 of the wall street journal as training data and section 20 as test data), the performance is lower. One of the reasons might be the highly frequent appearance of a long base NP such as "a two-arm, randomized, placebo-controlled, double-blind, parallel-group, clinical trial". Other reasons include many uses of parentheses and hyphens in clinical trial abstracts.

In the results of category determination, CRF shows slightly better performance than SVM. It is consistent with the theoretical fact that CRF has an advantage in determining labels with the consideration of global features. In the following, we use CRF for automatic base NP chunking and categorization.

3.3.3 Regular expression pattern matching

In order to extract two IE targets such as "Patient Population" and "Compared Treatments", we set up regular expression patterns shown in Table 3.6. We extract base NPs having the class label of "Disease" from the pattern matched expressions for "Patient Population", and extract base NPs having the label of "Treatment" for "Compared Treatments", as the final IE results. We expect readers agree that the patterns are not too elaborate to make out. It is possibly true that those simple patterns do not match all the expressions containing IE targets, or do match non IE targets. However, in this thesis, we focus not on the hand-made patterns but on automatic pattern construction based on machine learning techniques, which will be described in Chapter 3.4.

Table 3.7: Results of IE; # of abstracts correct information extracted

IE target	base NP chunking and categorization; automatic labeling (AL) or hand-labeling (HL)	# of abstracts
Patient Population	automatic labeling (AL)	125
	hand-labeling (HL)	131
Compared Treatments	automatic labeling (AL)	118
	hand-labeling (HL)	148

3.3.4 Results of information extraction from MEDLINE abstracts

The evaluation measure is the number of abstracts, whose IE targets only are extracted by regular expression pattern matching. As for the abstracts that don't have IE targets, correct answers are the cases nothing is extracted. If an IE target entity is mentioned multiple times in an abstract, count as a correct answer when at least one of those is extracted.

The results of IE from a total of 200 abstracts are shown in Table 3.7. On the situation of automatic base NP chunking and categorization, "Patient Population" and "Compared Treatments" are correctly extracted from 125 and 118 abstracts, respectively. In the comparison of the results in "Compared Treatments" between automatic labeling (118 abstracts) and hand-labeling (148 abstracts), we notice that the failure of automatic labeling largely affects the IE results. In addition, seeing that even the results using hand-labeled data (131 and 148 abstracts for "Patient Population" and "Compared Treatments", respectively) don't reach near full marks (namely 200 abstracts), we can say that there is room (left for the next chapter) to improve the results.

3.4 Filtering based on classification techniques

Here, in order to improve the results obtained in the previous chapter, we conduct IE with filtering based on the document and sentence classification techniques.

3.4.1 Document filtering

The purpose here is to filter out the exploratory abstracts that really don't contain our IE targets. In other words, selecting the MEDLINE documents using the query

Table 3.8: Results of document classification

classifier	answer	system	share	recall	precision
SVM (TinySVM)	140	159	132	94.3%	83.0%

of "Clinical Trial, Phase III [Publication Type]" does not fully achieve the purpose of selecting confirmatory and removing exploratory abstracts. For example, abstracts that just report the results of exploratory analyses using data or participants in past phase III clinical trials are not excluded. In fact, the reports of exploratory analyses are less important for EBM than those of confirmatory analyses, even with having to do with the phase III clinical trials.

3.4.2 Methods and results of automatic document classification

We manually annotated the 200 abstracts with a correct answer of the binary class, that is, positive class for 140 confirmatory abstracts and negative class for 60 exploratory abstracts. Then we divided them into 1 test and 199 training abstracts (leave-one-out setting), and used a SVM classifier [8] with word frequencies as features. The results are shown in Table 3.8.

3.4.3 Sentence filtering

Here, we are going to select the sentences that contain direct statements of IE targets. For example, from a sentence: "We compared doxorubicin to placebo in a phase III trial", we know without inference that the treatments compared in the trial are doxorubicin and placebo. However, from a sentence which describes a result: "The hazard ratio in doxorubicin compared with placebo was 0.97", we can only understand that doxorubicin and placebo were compared somewhere in the analyses of the trial, that is, it is possible that doxorubicin and placebo are not the IE targets ("Compared Treatments") in the trial. Another example is a sentence matched with the pattern "/Patient with Disease/". From a sentence: "We compared docetaxel to placebo in patients with metastatic breast cancer in a phase III trial", we know the patient population of the trial

[8] TinySVM: http://www.chasen.org/taku/software/TinySVM

is metastatic breast cancer. However, from a false positive sentence: "Subgroup analysis showed that patients with squamous-cell tumours had better survival", we can only know that a subgroup analysis was conducted using the patients with squamous-cell tumours, and so we cannot get the information about the IE targets ("Patient Population") in the trial.

3.4.4 Methods and results of automatic sentence classification

We manually annotated the 2,390 sentences in 200 abstracts with a correct answer of the binary class, that is, 369 positive and 2,021 negative sentences in "Patient Population", and 343 positive and 2,047 negative sentences in "Compared Treatments". After that, we divided them into training and test sets, and then employed BACT [9], a state-of-the-art sentence classification algorithm [32], that uses machine learning to acquire optimal classification patterns and classify sentences according to them.

BACT takes a sentence as input in the form of an ordered tree. The form of the trees varies according to how the connectivity of words in a sentence is defined. If the sentence is seen as a set of independent words (no connectivity between words; so-called bag of words; BOW), a sub-tree represents just a single word node. If the sentence is seen as a sequence of words, a sub-tree represents an N-gram. If the sentence is parsed into a word dependency tree, a sub-tree represents a set of words that are directly connected by syntactic relations. After constructing an ordered tree for each training sentence, BACT searches for sub-trees in each ordered tree and ranks the sub-trees that are found according to their applicability in sentence classification.

Both the BOW and N-gram assumptions are simple: BOW assumes interesting patterns exist within any single words in a sentence, and N-gram assumes interesting patterns exist within any sequences of neighboring words in a sentence, so they can be expressed using regular expressions. On the other hand, a dependency grammar restriction is available only if lexical knowledge can determine a dependency relationship between the words in a sentence correctly enough. In this experiment, we converted a base NP labeled sentence (automatic labeling or hand-labeling) into a phrase-structure tree using the phrase-structure analyzer proposed by Charniak [9], and then converted the phrase-structure tree into a dependency tree using the head rules described in Collin's doctoral dissertation [13]. Although BACT can also handle

[9] http://tahoo.org/taku/software/bact

Table 3.9: Results of sentence classification using BACT

IE target	assumption	answer	system	share	recall	precision
Patient Population	BOW	369	350	276	74.8%	78.9%
	N-gram	369	365	293	79.4%	80.3%
	dependency	369	366	290	78.6%	79.2%
Compared Treatments	BOW	343	333	268	78.1%	80.5%
	N-gram	343	333	272	79.3%	81.7%
	dependency	343	327	261	76.1%	79.8%

a phrase-structure tree as input, we select the BOW, N-gram, and dependency tree for its ease in interpreting patterns.

We compared performance between a BOW assumption, an N-gram assumption, and a dependency grammar restriction. The results of 10-fold cross validation of sentence classification are shown in Table 3.9. Assuming ideal circumstances in which the parser always works correctly, we might expect dependency trees to outperform N-grams and BOWs because dependency trees make use of more lexical knowledge. For example, N-grams can only deal with multiple neighboring word expressions and BOWs cannot even handle fixed expressions. However in this experiment, dependency trees are outperformed by N-grams, and even by BOWs regarding "Compared Treatments". We can guess the reason for this is that parse errors occurred in many of the dependency trees caused by the ambiguity of prepositional phrase (PP) attachments and coordination structures, which are quite often appeared in clinical trial abstracts. In the following, we use the N-gram assumption for automatic sentence filtering.

3.4.5 Results of information extraction from MEDLINE abstracts with or without filtering

The results of IE with or without filtering from a total of 200 abstracts are shown in Table 3.10 and Table 3.11 [10]. The evaluation measure is the same one as used in Section 3.4. On the fully automatic situation, "Patient Population" and "Compared Treatments" are correctly extracted in 153 and 136 abstracts, respectively. We can

[10]In Table 3.10 and Table 3.11, "AL": automatic labeling, "HL": hand-labeling, "AF": automatic filtering, and "HF": hand-filtering. "*", "**", and "***" mean that the results come from the process that includes one, two, and three (= all) non automatic steps, respectively.

Table 3.10: Results of IE with or without filtering in terms of "Patient Population"; # of abstracts correct information extracted

Patient Population			
base NP	document filtering	sentence filtering	# of abstracts
AL	not applied	not applied	125
AL	not applied	AF	142
AL	not applied	HF	*155
AL	AF	not applied	146
AL	AF	AF	153
AL	AF	HF	*161
AL	HF	not applied	*174
AL	HF	AF	*177
AL	HF	HF	**183
HL	not applied	not applied	*131
HL	not applied	AF	*142
HL	not applied	HF	**158
HL	AF	not applied	*152
HL	AF	AF	*154
HL	AF	HF	**163
HL	HF	not applied	**181
HL	HF	AF	**179
HL	HF	HF	***187

Table 3.11: Results of IE with or without filtering in terms of "Compared Treatments"; # of abstracts correct information extracted

Compared Treatments			
base NP	document filtering	sentence filtering	# of abstracts
AL	not applied	not applied	118
AL	not applied	AF	130
AL	not applied	HF	*137
AL	AF	not applied	130
AL	AF	AF	136
AL	AF	HF	*141
AL	HF	not applied	*157
AL	HF	AF	*156
AL	HF	HF	**160
HL	not applied	not applied	*148
HL	not applied	AF	*153
HL	not applied	HF	**162
HL	AF	not applied	*158
HL	AF	AF	*157
HL	AF	HF	**163
HL	HF	not applied	**184
HL	HF	AF	**178
HL	HF	HF	***185

confirm that document and sentence filtering is effective for "Patient Population": the shift of the number of correctly extracted abstracts according to the filtering condition is 125 (not applied), 153 (AF: automatic filtering), and 183 (HF: hand-filtering). As for "Compared Treatments", although it also improves the results from 118 (not applied) to 136 (AF: automatic filtering) and 160 (HF: hand-filtering), the base NP labeling problem still remains.

We note a comment about the number of abstracts on the fully manual labeling and filtering situation: 187 and 185 for "Patient Population" and "Compared Treatments", respectively. The reasons that IE targets are not extracted from the rest (13 and 15 abstracts) include having filtered out indirect statements that may contain IE targets, incorrect hand-labeling and hand-filtering, and typographical errors in original texts.

3.5 Discussion

In the above, we have reported the IE experiment using NLP techniques consisting of base NP chunking and categorization, regular expression pattern matching, and document and sentence filtering. Essentially, it is all right if we can assign a specific label to the IE targets via the first step only (i.e. base NP chunking and categorization) and extract them as final outputs. However, such the label is not so easy to learn by the framework of sequence labeling, hence we have written regular expression patterns, consulted global features in document classification, and tried to utilize information from syntactic structures in sentence classification. Then we have seen the results of IE are improved with the addition of document and sentence classification.

Still, of course, we need to make an effort to improve the performance of each step. In order to raise the accuracy of the base NP chunking and categorization, one way is to employ a multiclass classifier for the base NP categorization after executing base NP chunking by sequence labeling. A merit of this method is to be able to incorporate finer features such as the initial token, the last token, and the length of the chunked NPs. We can confirm the effectiveness by using the software SVM multiclass [11] [14, 56] as shown in Table 3.12, where the mid column corresponds to the F rate obtained by applying sequence labeling (CRF) to both chunking and categorization (Method A), and the right column corresponds to the method mentioned here (Method B). Note that the results (of CRF) in Table 3.5 that are derived from using correct (gold standard) chunks of the base NPs do not agree with the values of the mid column in Table 3.12.

[11] http://svmlight.joachims.org/svm_multiclass.html

Table 3.12: Comparison of F rate (%) for base NP chunking and categorization

class	Method A	Method B
Disease	74.0	81.6
Treatment	78.9	82.2
Patient	91.5	94.7
Study	90.1	92.9
Others	85.1	88.2

In the step of document classification, we used a classifier based on SVM. Here, we just put a comment that there are other classifiers based on probabilistic models such as probabilistic latent semantic indexing (PLSI) or naive Bayes classifiers.

In the sentence classification step, we might be able to utilize the information of the headings such as BACKGROUND, OBJECTIVE, METHOD, RESULTS, and CONCLUSION. In our experiments, 146 out of 200 abstracts have this kind of headings.

The idea that the IE benefits from text classification is not a new one [37, 46]. In bio-informatics, H. Yu and E. Agichtein have used a classifier based on SVM for extracting gene and protein synonyms from biological journal articles [61]. M. Craven and J. Kumlien have applied a naive Bayes classifier to the task of protein subcellular-localization, and also have approached using a shallow phrase-structure parser, an area that is similar to our approach [16]. The approaches differ in that, firstly, we used dependency trees, and secondly, we employed BACT that learns comprehensively from all sub-trees of the training sentences.

As of extracting "Compared Treatments", it has been shown that the improvement of accuracy in base NP chunking and categorization is critical. One solution is to conduct base NP categorization according to the MeSH, the US National Library of Medicine's controlled vocabulary thesaurus. One of the advantages in the solution is no requirement for hand-labeled data for training. On the other hand, there are at least two problems. Firstly, the MeSH categories (see Table 3.13) don't exactly suit our purpose. For example, although the "C" category in MeSH almost corresponds to our "Disease" label, it also includes more general terms such as "tumor" or "recurrence". With regard to our "Treatment" label, the "D" and "E" categories in MeSH are helpful, but "E" also includes terms in epidemiological methods or statistics such as "multivariate analysis" or "confidence interval". Secondly, the entry terms in MeSH don't cover all the representation actually used in MEDLINE abstracts. For example, "interferon

Table 3.13: The top categories in MeSH thesaurus

1	Anatomy [A]
2	Organisms [B]
3	Diseases [C]
4	Chemicals and Drugs [D]
5	Analytical, Diagnostic and Therapeutic Techniques and Equipment [E]
6	Psychiatry and Psychology [F]
7	Biological Sciences [G]
8	Physical Sciences [H]
9	Anthropology, Education, Sociology and Social Phenomena [I]
10	Technology and Food and Beverages [J]
11	Humanities [K]
12	Information Science [L]
13	Persons [M]
14	Health Care [N]
15	Publication Characteristics [V]
16	Geographic Locations [Z]

alfa-2a", an antiviral and antineoplastic agent, can be written as "interferon alpha-2a", "interferon alfa 2a", "INF alpha-2a", and so on. After all, we consider it is better we will make use of the MeSH information as a feature on the use of machine learning techniques.

3.6 Summary

In this chapter, we have reported results from experiment in extracting information about the design of phase III clinical trials such as "Patient Population" and "Compared Treatments" from their MEDLINE abstracts. We have seen that the results of IE are improved with the additional use of document and sentence classification techniques. To obtain better results in the next stage of research, the key lies in improving the accuracy of base NP chunking and categorization, and also in improving parsing accuracy in sentence classification, as coordination structure or PP attachment ambiguity reduces its overall accuracy.

Chapter 4

A Discriminative Learning Model for Coordinations

4.1 Introduction

Coordination, along with prepositional phrase attachment, is a major source of ambiguity in natural language. Although only a small number of previous studies in natural language processing have dealt with coordinations, this does not mean disambiguating coordinations is easy and negligible; it still remains one of the difficulties for state-of-the-art parsers. In Charniak and Johnson's recent work [10], for instance, two of the features incorporated in their parse reranker are aimed specifically at resolving coordination ambiguities.

Previous work on coordinations includes [1, 8, 35, 40, 41, 45]. Earlier studies [1, 41] attempt to find heuristic rules to disambiguate coordinations. More recent research are concerned with capturing structural similarity between conjuncts using thesauri and corpora [8], or web-based statistics [40].

We identify three problems associated with the previous work.

1. Most of these studies evaluate the proposed heuristics against restricted forms of conjunctions. In some cases, they only deal with coordinations with exactly two conjuncts. This leaves the generality of these heuristics unclear.

2. Most of these studies assume that the boundaries of coordinations are known in advance, which, in our opinion, is an impractical.

3. The proposed heuristics and statistics capture many different aspects of coordination. However, it is not clear how they interact and how they can be combined.

To address these problems, we propose a new framework for detecting and disambiguating coordinations. Being a discriminative learning model, it can incorporate a large number of overlapping features, encoding various heuristics for coordination disambiguation. It thus provides a test bed for examining combined use of the proposed heuristics as well as new ones. As the weight on each feature is automatically tuned on the training data, assessing these weights allows us to evaluate the relative merit of individual features.

Our learning model is also designed to admit examples in which only the beginning and end of coordinated phrases are marked, to reduce the cost of training data annotation.

The state space of our model resembles that of Kurohashi and Nagao's Japanese coordination detection method [35]. However, they considered only the decoding of coordinated phrases and did not address automatic parameter tuning.

4.2 History of coordination disambiguation

Coordinate phrases have been recognized as a source of ambiguity that can sometimes produce multiple interpretations, and so their disambiguation has been a matter of concern in the natural language processing community for long days. It is widely known that a key of disambiguation lies in the similarities between coordinated phrases [49]. Some researchers have even proceeded to build a syntax that accounts for coordinations [27, 28]. However, in terms of practical natural language parsers, and in particular, probabilistic context-free grammar parsers, incorporating such similarities into rewrite rules is not straightforward. In fact Charniak's famous parser has improved accuracy by adding a post-processing step, which reranks parse trees according to features including coordination relevant ones [9, 10].

There have been two topics in coordination disambiguation. One topic is the determination of the boundary of coordinated conjuncts. Resnik [45] attempted to disambiguate nominal compounds by heuristics based on counts in corpus, where correct bracketings like "(bank and warehouse) guard" or "(policeman) and (park guard)" are discriminated from incorrect ones of "(bank) and (warehouse guard)" or "(policeman and park) guard". Almost in the same way, [23] focused on attachment of ambiguous coordinate phrases, and [8] concentrated on coordinations where two syntactic

readings are possible. [40] also attempted to disambiguate noun compounds based on web-derived features.

Another topic is the coordination boundary detection from a given sentence. This is a harder topic because quite a few syntactic readings are possible, while it is much more practical and interesting for us in that this could be adopted as a pre-processing step for parsing. [1, 41] attempted to detect coordination by procedures based on heuristics. [34, 35] presented a more sophsticated way to detect coordination boundaries in Japanese using dynamic programming, however, the similarity values and penalty points for the calculation of the path scores are determined by heuristics.

Although all the above basically assumed that each coordinate conjunction conjoins only two conjuncts, more complex coordinations that comprise three or more conjuncts are not rare. [33] extended their method to disambiguate coordinate phrases including punctuation in English, such as ”A, B, C, and D”.

4.3 Coordination disambiguation as sequence alignment

It is widely acknowledged that coordinations often consist of two or more conjuncts having similar syntactic constructs [45]. Our coordination detection model also follows this observation. To detect such similar constructs, we use the sequence alignment technique [24].

4.3.1 Sequence alignment

Sequence alignment is defined in terms of transformation of one sequence (string) into another through an *alignment*, or a series of edit operations. Each of the edit

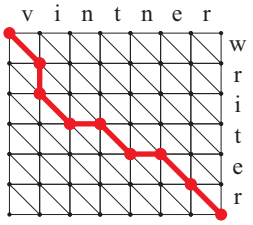


Figure 4.1: An alignment between ‘writer’ and ‘vintner’, represented as a path in an edit graph

operations has an associated cost, and the cost of an alignment is defined as the total cost of edit operations involved in the alignment. The minimum cost alignment can be computed by dynamic programming in a state space called an *edit graph*, such as illustrated in Figure 4.1. In this graph, a complete path starting from the upper-left initial vertex and arriving at the lower-right terminal vertex constitutes a global alignment. Likewise, a partial path corresponds to a local alignment.

Sequence alignment can also be formulated with the *scores* of edit operations instead of their *costs*. In this case, the sequence alignment problem is that of finding a series of edit operations with the maximum score.

4.3.2 Edit graph for coordinations

A fundamental difference between biological local sequence alignment and coordination detection is that the former deals with finding local homologies between two (or more) distinct sequences, whereas coordination detection is concerned with local similarities within a single sentence.

The maximal local alignment between two identical sequences is a trivial (global) alignment of identity transformation (the diagonal path in an edit graph). Coordination detection thus reduces to finding *off-diagonal* partial paths with the highest similarity score. Such paths never cross the diagonal, and we can limit our search space to the upper triangular part of the edit graph, as illustrated in Figure 4.2.

4.4 Automatic parameter tuning

Given a suitable substitution matrix, i.e., function from edit operations to scores, it is straightforward to find optimal alignments, or coordinations in our task, by running the Viterbi algorithm in an edit graph.

In computational biology, there exist established substitution matrices (e.g., PAM and BLOSUM) built on a generative model of mutations and their associated probabilities.

Such convenient substitution matrices do not exist for coordination detection. Moreover, optimal score functions are likely to vary from one domain (or language) to another. Instead of designing a specific function for a single domain, we propose a general discriminative learning model in which the score function is a linear function of

the *features* assigned to vertices and edges in the state space, and the weight of the features are automatically tuned for given gold standard data (training examples) drawn from the application domain. Designing heuristic rules for coordination detection, such as those proposed in previous studies, translates to designing suitable features in our model.

Our learning method is an extension of Collins's perceptron-based method for sequence labeling [12]. However, a few incompatibilities exists between Collins's sequence labeling method and edit graphs used for sequence alignment.

1. Collins's method, like the linear-chain conditional random fields (CRFs) [36, 52], seeks for a complete path from the initial vertex to the terminal using the Viterbi algorithm. In an edit graph, on the other hand, coordinations are represented by partial paths. And we somehow need to complement the partial path to make a complete path.

2. A substitution matrix, which defines the score of edit operations, can be represented as a function of features defined on edges. But to deal with complex coordinations, a more expressive score function is sometimes desirable, so that scores can be computed not only on the basis of a single edit operation, but also on consecutive edit operations. Edit graphs are not designed to accommodate features for such a higher-order interaction of edit operations.

To reconcile these incompatibilities, we derive a more finer-grained model from the original edit graph. In presenting the description of our model below, we reserve the terminology 'vertex' and 'edge' for the original edit graph, and use 'node' and 'arc' for our new model, to avoid confusion.

4.4.1 State space for learning coordinations

The new model is also based on the edit graph. In this model, we create a node for each triple (v, p, e) where v is a vertex in the original edit graph, $e \in \{Delete, Insert, Substitute\}$ is an admissible [1] edit operation at the vertex v, and $p \in \{Inside, Outside\}$ is a polarity denoting whether or not the edit operation e is involved in an alignment.

[1] For a vertex v at the border of an edit graph, some edit operations are not applicable (e.g., *Insert* and *Substitute* at vertices on the right border in Figure 4.2); we say such operations are *inadmissible* at v. Otherwise, an edit operation is *admissible*.

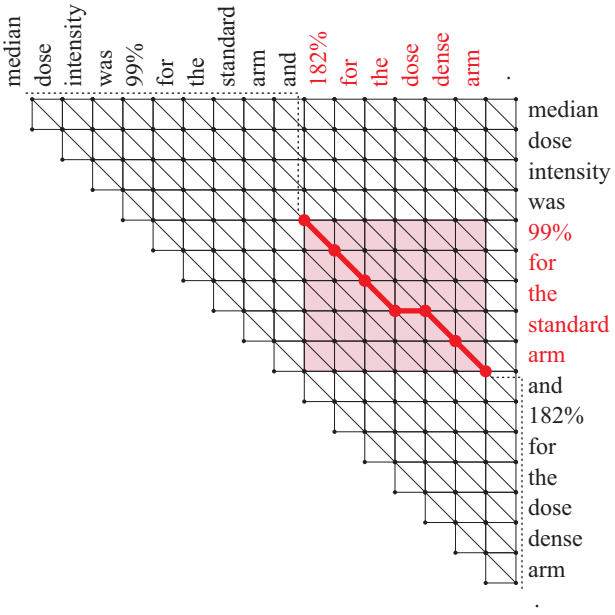


Figure 4.2: An edit graph for coordinate detection

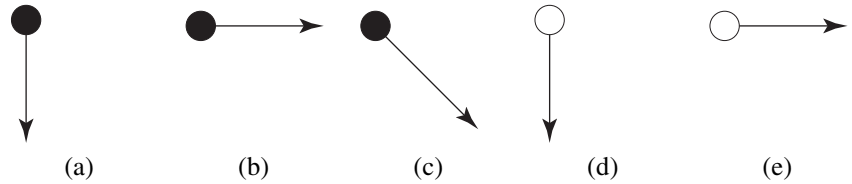


Figure 4.3: Five node types created for a vertex in an edit graph: (a) *Inside Delete*, (b) *Inside Insert*, (c) *Inside Substitute*, (d) *Outside Delete*, and (e) *Outside Insert*.

For a node (v, p, e), we call the pair (p, e) its *type*. All five possible node types for a single vertex of an edit graph are shown in Figure 4.3. We disallow the type $(Substitute, Outside)$, as it is difficult to attribute an intuitive meaning to substitution when two words are not aligned (i.e., *Outside*).

Arcs between nodes are built according to the transitions allowed in the original edit graph. To be precise, an arc between node (v_1, p_1, e_1) and node (v_2, p_2, e_2) is created if and only if the following three conditions are met.

(i) Edit operations e_1 and e_2 are admissible at v_1 and v_2, respectively,

(ii) the sink of the edge for e_1 at v_1 is v_2, and

(iii) it is not the case with $p_1 = p_2$ and $(e_1, e_2) = (Delete, Insert)$.

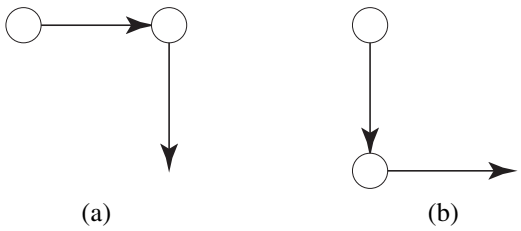


Figure 4.4: Series of edit operations with an equivalent net effect. (a) $(Insert, Delete)$, and (b) $(Delete, Insert)$. (b) is prohibited in our model.

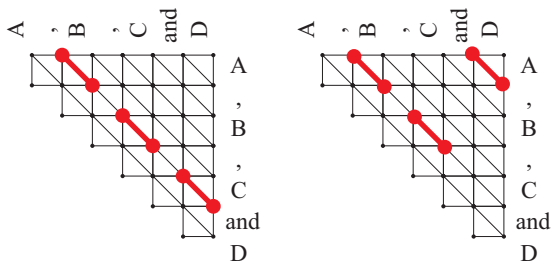
Condition (iii) is introduced so as to disallow transition depicted in Figure 4.4(b). In contrast, the sequence $(Insert, Delete)$ (Figure 4.4(a)) is allowed. The net effects of these operation sequences are identical, in that they both skip one word each from the two sequences to be aligned. As a result, there is no use in discriminating between these two, and one of them, namely $(Delete, Insert)$, is prohibited.

4.4.2 Learning task

By the restriction of condition (iii) introduced above and the omission of *Outside Substitute* from the node types, we can uniquely determine the complete path (from the initial node to the terminal node) that conjoins all the local alignments by *Outside* nodes (which correspond to edges in the original edit graph). In Figure 4.2, the augmented *Outside* edges in this unique path are plotted as dotted lines for illustration.

Thus we obtain a complete path which is compatible with Collins's perceptron-based sequence learning method. The objective of the learning algorithms, which we will describe in the next section, is to optimize the weight of features so that running the Viterbi algorithm will yield the same path as the gold standard.

Because a node in our state space corresponds to an edge in the original edit graph (see Figure 4.3), an arc in our state space is actually a pair of consecutive edges (or equivalently, edit operations) in the original graph. Hence our state model is more expressive than the original edit graph in that the score function can have a term (feature) defined on a pair of edit operations instead of one.



(a) valid (b) non-valid

Figure 4.5: A coordination with four conjuncts represented as (a) chainable, and (b) non-chainable partial paths. We take (a) as the canonical representation.

4.4.3 More complex coordinations

Even if a coordination comprises three or more conjuncts, our model can handle them as it can be represented as a set of pairwise local alignments that are *chainable* [24, Section 13.3]. If pairwise local alignments are chainable, a unique complete path that conjoins all these alignments can be determined, allowing the same treatment as the case with two conjuncts.

For instance, a coordination with four conjuncts (A, B, C and D) can be decomposed into a set of pairwise alignments $\{(A,B),(B,C),(C,D)\}$ as depicted in Figure 4.5 (a). This set of alignments are chainable and thus constitute the canonical encoding for this coordination; any other pairwise decomposition for these four conjuncts, like $\{(A,B),(B,C),(A,D)\}$ (Figure 4.5 (b)) is not chainable.

Our model can handle multiple non-nested coordinations in a single sentence as well, as they can also be decomposed into chainable pairwise alignments. However, it cannot encode nested coordinations like (A, B, and (C and D)), which should be a future challenge.

4.5 Algorithms

4.5.1 Reducing the cost of training data construction

Our learning method is supervised, meaning that it requires training data annotated with correct labels. Since a label in our problem is local alignments (or paths in an edit graph) representing coordinations, the training sentences have to be annotated with word-by-word alignments.

There are two reasons relaxing this requirement is desirable. First, it is expensive to construct such data. Second, there are coordinations in which the word-by-word correspondence is unclear even for humans. In Figure 4.2, for example, a word-by-word alignment of 'standard' with 'dense' is depicted, but it might be more natural to regard a word 'standard' as being aligned with two words 'dose dense' combined together.

Even when the word-by-word alignment is uncertain, the boundaries of conjuncts are often obvious, and it is also much easier to mark only the beginning and end of each conjunct. Thus we would like to allow for training examples in which only alignment boundaries are specified, instead of a full word-by-word alignment.

For these examples, conjunct boundaries corresponds to a rectangular region rather than a single path in an edit graph. The shaded box in Figure 4.2 illustrates the rectangular region determined by the boundaries of an alignment between the phrases "182 % for the dose dense arm" and "99 % for the standard arm". There are many possible alignment paths in this box, among which we do not know which one is correct (or even likely). To deal with this difficulty, we propose two simple heuristics we call (i) path-based and (ii) box-based methods. As mentioned earlier, both of these methods are based on Collins's averaged-perceptron algorithm for sequence labeling [12].

4.5.2 Path-based method

Our first method, which we call the "path-based" algorithm, is shown in Figure 4.6. We denote by $A(x)$ all possible alignments (paths) over x. The algorithm receives T, the maximum number of iterations, and a set of examples $S = \{(x_i, Y_i)\}$ as input, where x_i is a sentence (a sequence of words with their attributes, e.g., part-of-speech, prefixes and suffixes) and $Y_i \subset A(x_i)$ is the set of admissible alignments (paths) for x_i. When a sentence is fully annotated with a word-by-word alignment y, $Y_i = \{y\}$ is a singleton set. In general boundary-only examples we described in Section 4.5.1, Y_i holds all possible alignment compatible with the marked range, or equivalently, paths that pass through the upper-left and lower-right corners of a rectangular region. Note that it is not necessary to explicitly enumerate all the member paths of Y_i; the set notation here is only for the sake of presentation.

The external function $f(x, y)$ returns a vector (called the *global feature vector* in [52]) of the number of feature occurrences along the alignment path y. In the beginning (line 5 in the figure) of the inner loop, the target path (alignment) is recomputed with

input: Set of examples $S = \{(x_i, Y_i)\}$
Iteration cutoff T
output: Averaged weight vector $\bar{w}$

1: $\bar{w} \leftarrow 0$; $w \leftarrow 0$
2: **for** $t \leftarrow 1 \ldots T$ **do**
3: $\quad \Delta w \leftarrow 0$
4: $\quad$ **for each** $(x_i, Y_i) \in S$ **do**
5: $\quad\quad y \leftarrow \arg\max_{y \in Y_i} w \cdot f(x_i, y)$
6: $\quad\quad y' \leftarrow \arg\max_{y \in A(x_i)} w \cdot f(x_i, y)$
7: $\quad\quad \Delta f \leftarrow f(x_i, y) - f(x_i, y')$
8: $\quad\quad \Delta w \leftarrow \Delta w + \Delta f$
9: $\quad$ **end for**
10: $\quad$ **if** $\Delta w = 0$ **then**
11: $\quad\quad$ **return** $\bar{w}$
12: $\quad$ **end if**
13: $\quad w \leftarrow w + \Delta w$
14: $\quad \bar{w} \leftarrow [(t-1)\bar{w} + w]/t$
15: **end for**
16: **return** $\bar{w}$

Figure 4.6: Path-based algorithm

the current weight vector w. The arg max in lines 5 and 6 can be computed efficiently ($O(n^2)$, where n is the number of words in x) by running a pass of Viterbi algorithm in the edit graph for x. The weight vector w varies between iterations, and so does the most likely alignment with respect to w. Hence the recomputation in line 5 is needed.

4.5.3 Box-based method

Our next method, called "box-based", is designed on the following heuristics. Given a rectangle region representing a local alignment (hence all nodes in the region are of polarity *Inside*) in an edit graph, we distribute feature weights in proportion to the probability of a node (or an arc) being passed by a path from the initial (upper left) node to the terminal (lower right) node of the rectangle. We assume paths are uniformly distributed.

Figure 4.8 displays an 8×8 sub-grid of an edit graph. The figure under each vertex shows the number of paths passing through the vertex. Vertices near the upper-left and the lower-right corner have a large frequency, and the frequency drops exponentially towards the top right corner and the bottom left corner, hence placing a strong bias

input: Set of examples $S = \{(x_i, Y_i)\}$
Iteration cutoff T
output: Averaged weight vector $\bar{w}$

1: $\bar{w} \leftarrow 0$; $w \leftarrow 0$
2: **for each** $(x_i, Y_i) \in S$ **do**
3: $\quad g_i \leftarrow (1/|Y_i|) \sum_{y \in Y_i} f(x_i, y)$
4: **end for**
5: **for** $t \leftarrow 1 \ldots T$ **do**
6: $\quad \Delta w \leftarrow 0$
7: $\quad$ **for each** $(x_i, Y_i) \in S$ **do**
8: $\quad\quad y' \leftarrow \arg\max_{y \in A(x_i)} w \cdot f(x_i, y)$
9: $\quad\quad$ Convert y' into its box representation Y'
10: $\quad\quad g' \leftarrow (1/|Y'_i|) \sum_{y \in Y'_i} f(x_i, y)$
11: $\quad\quad \Delta f \leftarrow g_i - g'$
12: $\quad\quad \Delta w \leftarrow \Delta w + \Delta f$
13: $\quad$ **end for**
14: $\quad$ **if** $\Delta w = 0$ **then**
15: $\quad\quad$ **return** $\bar{w}$
16: $\quad$ **end if**
17: $\quad w \leftarrow w + \Delta w$
18: $\quad \bar{w} \leftarrow [(t-1)\bar{w} + w]/t$
19: **end for**
20: **return** $\bar{w}$

Figure 4.7: Box-based algorithm

on the paths near diagonals. This distribution naturally fits our preference towards alignments with a larger number of substitutions.

The pseudo-code for the box-based algorithm is shown in Figure 4.7. For each example x_i and its possible target labels (alignments) Y_i, this algorithm first (line 3) computes and stores in the vector g_i the average number of feature occurrences in all possible target paths in Y_i. This quantity can be computed simply by summing over all feature occurrences multiplied by the pre-computed frequency of each nodes and arcs at which these features occur, analogously to the forward-backward algorithm. In each iteration, the algorithm scans every example (lines 7–13), computing the Viterbi path y' (line 8) according to the current weight vector w. Line 9 then converts y' to its box representation Y', by sequentially collapsing consecutive *Inside* nodes in y' as a box.

For instance, let y' be the local alignment depicted in Figure 4.2. The box Y' computed in line 9 for this y' is the shaded area in the figure. In parallel to the initialization

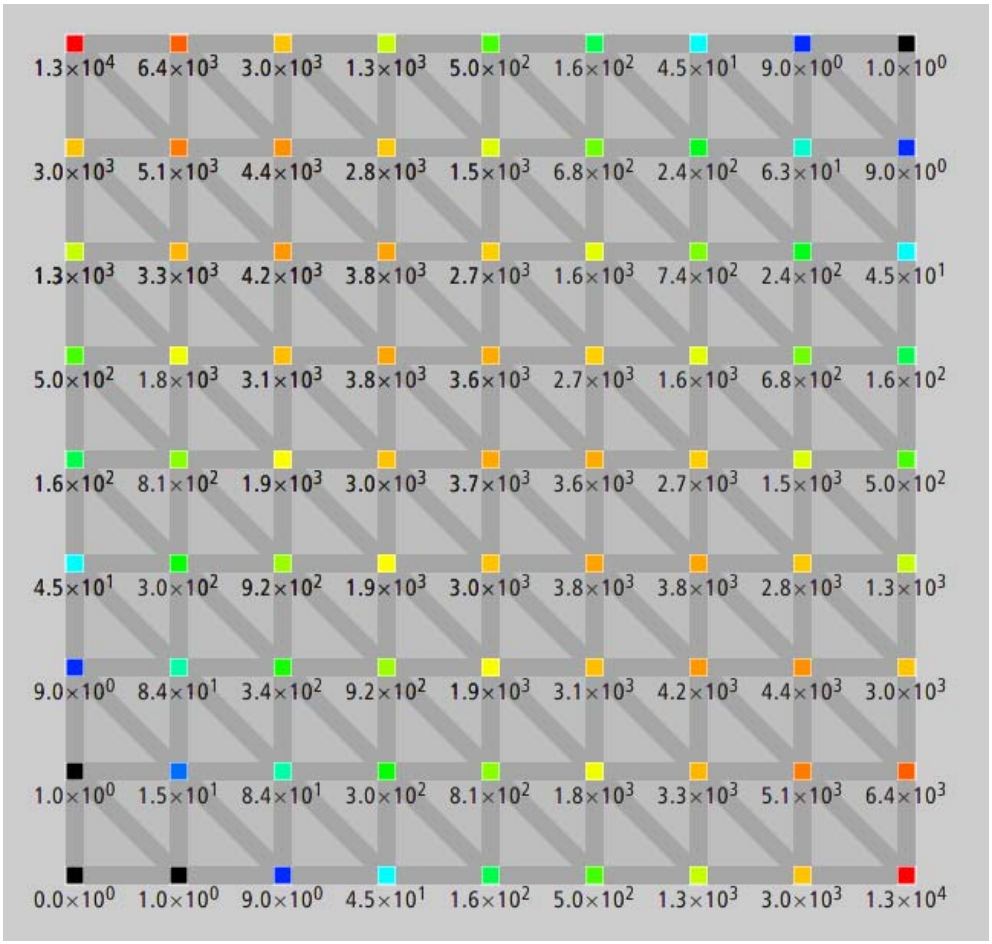


Figure 4.8: Number of paths passing through the vertices of an 8×8 grid.

step in line 3, we store in g' the average feature occurrences in Y' and update the current weight vector w by the difference between the target g_i and g'. These steps can be interpreted as a Viterbi approximation for computing optimal set Y' of alignments directly.

4.6 Related work

4.6.1 Discriminative learning of edit distance

In our model, the state space of sequence alignment, or edit graph, is two-dimensional (which is actually three-dimensional if the dimension for labels is taken into account).

This is contrastive to the one dimensional models used by Collins's perceptron-based sequence method [12] which our algorithms are based upon, and by the linear-chain CRFs.

McCallum et al. [39] proposed a CRF tailored to learning string edit distance for the identity uncertainty problems. The state space in their work is two dimensional just like our model, but it is composed of two decoupled subspaces, each corresponding to 'match' and 'mismatch,' thus sharing only the initial state. It is not possible to make a transition from a state in the 'match' state space to 'mismatch' space (and vice versa). As we can see from the decoupled state space, this method is based on global alignment rather than local alignment; it is not clear whether their method can identify local homologies in sequences. Our method uses a single state space in which both 'match (inside)' and 'mismatch (outside)' nodes co-exist and the transition between them is permitted.

4.6.2 Inverse sequence alignment in computational biology

In computational biology, the estimation of a substitution matrix from data is called the *inverse sequence alignment* problem. Until recently, there have been a relatively small number of papers in this field despite a large body of literature in sequence alignment. Theoretical studies in the inverse sequence alignment include [43, 54], both of which regards the inverse sequence alignment as an instance of Eppstein's inverse parameter problem [20]. Recently, CRFs have been applied for optimizing edit distance for global protein sequence alignment [18].

4.7 Empirical evaluation

4.7.1 Dataset and Task

We used the GENIA Treebank beta corpus [30][2] for evaluation of our methods. The corpus consists of 500 parsed abstracts in MEDLINE with a total of 4529 sentences.

Although the Penn Treebank Wall Street Journal (WSJ) corpus is the de facto standard corpus for evaluating chunking and parsing performance, it lacks adequate structural information on coordinations, and therefore does not serve our purpose. Many

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

coordinations in the Penn Treebank are given a flat bracketing like (A, B, and C D), and thus we cannot tell which of ((A, B, and C) D) and ((A), (B), and (C D)) gives a correct alignment. The GENIA corpus, in contrast, distinguishes ((A, B, and C) D) and ((A), (B), and (C D)) explicitly, by providing more detailed bracketing. In addition, the corpus contains an explicit tag "COOD" for marking coordinations.

To avoid nested coordinations, which admittedly require techniques other than the one proposed in this thesis, we selected from the GENIA corpus sentences in which the conjunction "and" occurs just once. After this operation, the number of sentences reduced to 1668, from which we further removed 32 that are not associated with the "COOD" tag, and 3 more whose annotated tree structures contained an obvious errors. Of the remaining 1633 sentences, 1061 were coordinated noun phrases annotated with NP-COOD tags, 226 coordinated verb phrases (VP-COOD), 142 coordinated adjective phrases (ADJP-COOD), and so on. Because the number of VP-COOD, ADJP-COOD, and other types of coordinated phrases are too small to make a meaningful benchmark, we focus on coordinated noun phrases in this experiment.

The task hence amounts to identifying coordinated NPs and their constituent conjuncts in the 1633 sentences, all of which contain a coordination marker "and" but only 1061 of which are actually coordinated NPs.

4.7.2 Baselines

We used several publicly available full parsers as baselines: (i) the Bikel parser [5] version 0.9.9c with configuration file "bikel.properties (denoted as Bikel/Bikel), (ii) the Bikel parser in the Collins parser emulation mode (using "collins.properties file) (Bikel/Collins), and (iii) the Charniak and Johnson's reranking parser (Charniak-Johnson)[10]. We trained Bikel's parser and its Collins emulator with the GENIA corpus, WSJ (Penn Treebank), and the combination of the two. Charniak and Johnson's parser was used as distributed at Charniak's home page (and is WSJ trained).

Another baseline we used is chunkers based on linear-chain CRFs with the standard BIO labels. We trained two types of CRF-based chunkers by giving different BIO sequences, one for the conjunct bracketing and the other for coordination bracketing. The chunkers were implemented with T. Kudo's CRF++ package version 0.45. We varied its regularization parameters C among $C \in \{0.01, 0.1, 1, 10, 100, 1000\}$, and the best results among these are reported below.

Table 4.1: Features for the proposed methods

Substitute **(diagonal) nodes**
• Indicators of the word, POS and morphological attributes of x_i, x_j, (x_{i-1}, x_i), (x_i, x_{i+1}), (x_{j-1}, x_j), (x_j, x_{j+1}), and (x_i, x_j), respectively combined with the type of the node.
• For each of the word, POS and morphological attributes, an indicator of whether the respective attribute is identical in x_i and x_j, combined with the type of the node.
Delete **(vertical) nodes**
• Indicators of the word, POS and morphological attributes of x_i, x_j, x_{j-1}, (x_{i-1}, x_i), (x_i, x_{i+1}), and (x_{j-1}, x_j), respectively combined with the type of the node.
Insert **(horizontal) nodes**
• Indicators of the word, POS and morphological attributes of x_i, x_{i-1}, x_j, (x_{i-1}, x_i), (x_{j-1}, x_j), and (x_j, x_{j+1}), respectively combined with the type of the node.
Any arcs
• Indicators of the POS attribute of x_i, x_{i-1}, x_j, x_{j-1}, (x_{i-2}, x_{i-1}), (x_{i-1}, x_i), (x_i, x_{i+1}), (x_{j-2}, x_{j-1}), (x_{j-1}, x_j), (x_j, x_{j+1}), (x_{i-1}, x_{j-1}), (x_{i-1}, x_j), (x_i, x_{j-1}) and (x_i, x_j), respectively combined with the type of the arc.
Inside-Outside **and** ***Outside-Inside*** **arcs**
• Indicator of the distance $j - i$ between two words x_i and x_j, combined with the type of the arc.

4.7.3 Features

Let $x = (x_1, \ldots, x_n)$ be a sentence, with its member x_i a vector of attributes for the ith word. The attributes include word surface, part-of-speech (POS), and suffixes, among others.

Table 4.1 summarizes (i) the features assigned to the nodes whose corresponding edge in the original edit graph for x is emanating from row i and column j, and (ii) the features assigned to the arcs (consisting of two edges in the original edit graph) whose joint (the vertex between the two edges) is a vertex at row i and column j.

We also tested the path-based and box-based methods, and the CRF chunkers, both with and without suffix features.

Although this is not a requirement of our model or algorithms, every feature we use in this experiment is binary; if the condition associated with a feature is satisfied, the feature takes a value of 1; otherwise, it is 0. A condition typically asks whether or not specific attributes match those at a current node, arc, or their neighbors.

We used the POS tags from the GENIA corpus as the POS attribute. The morphological features include 3- and 4-gram suffixes and indicators of whether a word includes capital letters, hyphens, and digits.

For the baseline CRF-based chunkers, we assign the word (surface word), POS (from GENIA), and morphological features to nodes, and the POS features to edges. The feature set is identical to those used for our proposed methods, except for features defined on row-column combination (i.e., those defined over both i and j in Table 4.1), which cannot be incorporated as a local features in chunkers based on linear chain.

For the Bikel (and its Collins emulation) parsers which accepts POS tags output by external taggers upon testing, we gave them the POS tags from the GENIA corpus, for fair comparison with the proposed methods and CRF-based chunkers.

4.7.4 Evaluation criteria

We employed two evaluation criteria: (i) correctness of the conjuncts output by the algorithm, and (ii) correctness of the range of coordinations as a whole.

For the correctness of conjuncts, we further use two evaluation criteria. The first evaluation method ("pairwise evaluation") is based on the decomposition of coordinations into the canonical set of pairwise alignments, as described in Section refsec:ternally-and-larger-conjuctions. After the set of pairwise alignments is obtained, each pairwise alignment is transformed into a box surrounded by their boundaries. Using these boxes, we evaluate precision, recall, and F rate through the following definition. The precision measures how many of the boxes output by the algorithm exactly match those in the gold standard, and the recall is the percentage of the correct boxes found by the algorithm. The F rate is the harmonic mean of the precision and recall.

The second evaluation method ("chunk-based evaluation") for conjuncts is based on whether the algorithm correctly outputs the beginning and end of each conjunct, in the same manner as the chunking task. Here, we adopt the evaluation criteria for the CoNLL 99 NP bracketing task [3]; The precision equals how many of the NP conjuncts output by the algorithm are correct, and the recall is the percentage of the correct NP conjuncts found by the algorithm.

Of these two evaluation methods for conjuncts, it is harder to obtain a higher pairwise evaluation score than the chunk-based evaluation. To be counted as a true positive in the pairwise evaluation, two consecutive chunks must be output correctly by the algorithm.

For the correctness of the coordination range, we check if both the start of the first coordinated conjunct and the end of the last conjunct in the gold standard match those

[3] http://www.cnts.ua.ac.be/conll99/npb/

Table 4.2: Performance on conjunct bracketing. P: precision (%), R: recall (%), F: F rate.

	Pairwise evaluation			Chunk-bassed evaluation		
Method	P	R	F	P	R	F
Path-based	61.4	56.2	58.7	70.9	66.9	68.9
Path-based w/o word and suffix features	61.7	**58.8**	**60.2**	**71.2**	**69.7**	**70.5**
Path-based w/o arc features	49.1	50.1	49.6	59.2	62.9	61.0
Box-based	60.6	58.3	59.4	70.5	69.1	69.8
Box-based w/o word and suffix features	59.5	58.3	58.9	69.7	69.5	69.6
Box-based w/o arc features	49.3	50.0	49.7	57.9	61.1	59.5
Linear-chain CRF chunker (conjunct bracketing)	**62.6**	51.4	56.4	71.0	66.1	68.5
Bikel/Collins (trained with GENIA)	50.0	48.6	49.3	65.0	64.2	64.6
Bikel/Bikel (trained with GENIA)	50.1	47.8	49.0	63.9	61.3	62.6

output by the algorithm. The reason we evaluate coordination range is to compare our proposed method with the full parsers trained on WSJ (but applied to GENIA). Although WSJ and GENIA differ in the way conjuncts are annotated, they are mostly identical on how the range of coordinations are annotated, and hence comparison is feasible in terms of coordination range. For the baseline parsers, we regard the bracketing directly surrounding the coordination marker "and" as their output.

In [11], an F score of 75.5 is reported for the Bikel parser on coordination detection. Their evaluation is based on dependencies, which is different from our evaluation criteria based on boundaries. Generally speaking, our evaluation criterion seems stricter, as exemplified by the mismatch in Figures 7 and 8 of Clegg and Shepherd's paper [11]; in these figures, our evaluation criterion would result in zero true positive, whereas their evaluation counts the dependency arc from 'genes' to 'human' as one true positive.

4.7.5 Results and discussion

The results of conjunct bracketing and coordination bracketing are shown in Tables 4.2 and 4.3, respectively. These are the results of a five-fold cross validation. We ran the proposed methods until convergence or the cutoff iteration of $T = 10000$, whichever comes first.

Some baseline parsers are evaluated only for coordination boundaries because, as described in Section 4.7.1, they are trained with the Penn Treebank which differs from

Table 4.3: Performance on coordination bracketing. P: precision (%), R: recall (%), F: F rate.

	P	R	F
Path-based	58.2	55.3	56.7
Path-based w/o word and suffix features	57.7	**56.6**	**57.2**
Path-based w/o arc features	46.1	48.4	47.3
Box-based	55.6	54.4	55.0
Box-based w/o word and suffix features	54.8	54.6	54.7
Box-based w/o arc features	42.9	45.9	44.4
Linear-chain CRF chunker (conjunct bracketing)	43.9	46.7	45.3
Linear-chain CRF chunker (coordination bracketing)	**58.4**	51.0	54.5
Bikel/Collins (trained with GENIA)	44.0	45.4	44.7
Bikel/Collins (trained with WSJ)	42.3	43.2	42.7
Bikel/Collins (trained with GENIA+WSJ)	43.3	45.1	44.1
Bikel/Bikel (trained with GENIA)	44.8	45.4	45.1
Bikel/Bikel (trained with WSJ)	40.7	41.5	41.1
Bikel/Bikel (trained with GENIA+WSJ)	43.9	45.8	44.9
Charniak-Johnson reranking parser	48.3	45.2	46.7
Charniak (former version)	41.4	40.8	41.1

Table 4.4: Performance on coordination bracketing; path-based algorithms without using word and suffix features. P: precision (%), R: recall (%), F: F rate.

	P	R	F
Averaged perceptron	57.7	56.6	57.2
Averaged perceptron (L1-norm regularization)	57.4	**57.1**	57.3
Averaged perceptron (L2-norm regularization)	57.6	**57.1**	**57.4**
Averaged perceptron (lambda trick)	53.9	50.0	51.9
MIRA	51.3	51.9	51.6
Bayes point machine	**58.8**	55.7	57.2

the coordination annotation of the GENIA Treebank.

The path-based method (without words and suffixes) and box-based method (with full features) each achieved 2.0 and 1.3 point improvement over the CRF chunker in terms of the F score in chunk-based evaluation, 3.8 and 3.0 point improvement in pairwise evaluation and 2.7 and 0.5 point in coordination identification, respectively. Our methods also showed a performance considerably higher than the baseline parsers.

To investigate whether the features assigned to arcs are effective or not, we also evaluated the proposed methods by removing arc features altogether. Without the arc features, the F rate dropped around 10 point both in the path-based and box-based method.

The performance of the path-based method was better when the word and suffix features were removed, while the box-based method and CRF chunker performed better with these features.

Among the baseline parsers, Charniak and Johnson's reranking parser achieved the highest F-score, despite being trained on the WSJ corpus. This is surprising given that other parsers shows a significantly inferior performance when trained on the Penn Treebank. For the Bikel/Bikel and Bikel/Collins parsers, the F measure was highest when they were trained on the GENIA corpus alone. This should be the effect brought by the new features for coordination disambiguation used in Charniak and Johnson's parser.

We also examined the performance of variants of the perceptron algorithms, described in Section 2.6. All those are path-based and without using word and suffix features. In Table 4.4, "Averaged perceptron" denotes the same one as "Path-based w/o word and suffix features" in Table 4.3. The hyper parameter C are 1.0 for L1-norm, 0.0001 for L2-norm, and 1.0 for the lambda trick, those are determined by trial and errors. Note that MIRA is the online algorithm, whereas others are batch ones. For the Bayes point machine, we prepare eight inner systems which use in total 80% training data randomly selected in each iteration.

4.8 Summary

We have proposed a new coordination learning and detection method that can incorporate many different features, and automatically optimize their weights on training data.

In the experiment of Section 4.7, we obtained a performance that are superior to a linear-chain chunker and to the state-of-art full parsers.

We used only syntactic and morphological features, and did not use external similarity measures like thesauri and corpora, although they are reported to be effective for disambiguating coordinations. We note that it is easy to incorporate such external similarity measures as features in our model, thanks to its two-dimensional state space. The similarity of two words derived from external knowledge bases can be assigned to a *Substitute* node at the corresponding locations in the state space, in a straightforward manner. This is a topic we are currently working on.

We are also planning to reimplement our algorithms using CRFs instead of the averaged perceptron algorithm.

Chapter 5

Conclusions

In this thesis, we have focused on developing the systems for information extraction and coordination disambiguation. As for information extraction, we have seen that to mark terms regarding treatments and patients is not so difficult. However, if we would like to extract terms in consideration of more detailed contexts, such as sentence structures, we need parser outputs in high accuracy. Biomedical texts contain a great many of coordinated phrases, that in fact should be the obstacles for conventional parsers, as well as they likely to include important information for EBM. Then we have proposed a method for detecting and disambiguating coordinate phrases, a unique feature of which is that it employs a perceptron-like learning algorithm to adapt the substitution matrix to the training data drawn from the target language and domain. We have obtained a promising empirical result in detecting and disambiguating coordinated noun phrases in the GENIA corpus, despite using a relatively small number of training examples with minimal features.

As future work, one, but significant, remaining problem is to adapt nested coordinations. Our proposed method can not deal with those because it is impossible to represent a nested coordination as a path in the upper triangle state space. However, it is the fact that very often we come across expressions containing nested coordinations in biomedical texts. For instance, we return to the MEDLINE abstract shown in Figure 1.1. We find a nested coordination in the sentence:

> "Median times to progression and median survival times were ***6.1 months and 8.9 months in arm A and 7.2 months and 9.5 months in arm B*** , respectively ."

Again we emphasize that the author described clinically important information using

Table 5.1: Evaluation of Charniak and Johnson's state-of-the-art parsing accuracy (recall) with respect to the "COOD" tags in GENIA corpus, stratified by non nested or nested coordination

	total	non nested		nested	
COOD type	# of tags	# of tags	recall (%)	# of tags	recall (%)
NP	2591	1797	43.2	794	38.5
VP	485	312	60.6	173	63.6
ADJP	409	304	49.0	105	41.0
S	283	133	44.4	150	38.0
PP	209	124	50.0	85	45.9

the structure of nested coordination in the sentence.

According to our survey of 100 MEDLINE abstracts regarding clinical trials, 303 are nested among a total of 1053 coordinations. Moreover as for the GENIA corpus, we show in Table 5.1 that the rate of nested coordination is one third to one quarter (more than half with respect to COOD type "S"). We see also in Table 5.1 that nested coordination is harder to detect, except COOD type "VP".

We wish to construct the nested coordination analyzer that can deal with those, and we are convincing that the systems will be valuable especially in conducting EBM (i.e. meta-analysis or systematic review for clinical trials), and also make a great deal of contribution to the NLP community.

Bibliography

[1] R. Agarwal and L. Boggess. A simple but useful approach to conjunct identification. In *Proceedings of the 30th Annual Meeting of the Association for Computing Linguistics (ACL'92)*, pp. 15–21, 1992.

[2] Y. Altun, I. Tsochantaridis, and T. Hofmann. Hidden Markov Support Vector Machines. In *Machine Learning Proceedings of the Twentieth International Conference (ICML 2003)*, pp. 3–10, 2003.

[3] G. Andrew and J. Gao. Scalable training of L1-regularized log-linear models. In Z. Ghahramani ed., *Proceedings of the 24th Annual International Conference on Machine Learning (ICML 2007)*, pp. 33–40. Omnipress, 2007.

[4] A. Bies. Bracketing guidelines for treebank ii style penn treebank project, 1995.

[5] D. M. Bikel. Multilingual statistical parsing engine version 0.9.9c, 2005. http://www.cis.upenn.edu/~dbikel/software.html.

[6] C. Bishop. *Pattern Recognition And Machine Learning*. Springer, 2006.

[7] E. Buyko, K. Tomanek, and U. Hahn. Resolution of coordination ellipses in biological named entities using conditional random fields. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*, 2007.

[8] F. Chantree, A. Kilgarriff, A. de Roeck, and A. Willis. Disambiguating coordinations using word distribution information. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2005)*, Borovets, Bulgaria, 2005.

[9] E. Charniak. A maximum-entropy-inspired parser. In *Proceedings of the Second Meeting of North American Chapter of Association for Computational Linguistics (NAACL-2000)*, pp. 132–139, 2000.

[10] E. Charniak and M. Johnson. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL-2005)*, 2005.

[11] A. B. Clegg and A. J. Shepherd. Benchmarking natural-language parsers for biological applications using dependency graphs. *BMC Bioinformatics*, 8(24), 2007.

[12] M. Collins. Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, 2002.

[13] M. Collins. Head-driven statistical models for natural language parsing. *Comput. Linguist.*, 29(4):589–637, 2003.

[14] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *J. Mach. Learn. Res.*, 2:265–292, 2002.

[15] K. Crammer and Y. Singer. Ultraconservative online algorithms for multiclass problems. *J. Mach. Learn. Res.*, 3:951–991, 2003.

[16] M. Craven and J. Kumlien. Constructing biological knowledge bases by extracting information from text sources. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, pp. 77–86, 1999.

[17] N. Cristianini and J. Shawe-Taylor. *An introduction to support Vector Machines: and other kernel-based learning methods*. Cambridge University Press, New York, NY, USA, 2000.

[18] C. B. Do, S. S. Gross, and S. Batzoglou. CONTRAlign: discriminative training for protein sequence alignment. In *Proceedings of the Tenth Annual International Conference on Computational Molecular Biology (RECOMB 2006)*, 2006.

[19] C. Do, D. Woods, and S. Batzoglou. CONTRAfold: RNA secondary structure prediction without energy-based models. In *14th Annual International Conference on Intelligent Systems for Molecular Biology*, 2006.

[20] D. Eppstein. Setting parameters by example. *SIAM Journal on Computing*, 32(3):643–653, 2003.

[21] Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. In *COLT' 98: Proceedings of the eleventh annual conference on Computational learning theory*, pp. 209–217, New York, NY, USA, 1998. ACM.

[22] Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999.

[23] M. Goldberg. An unsupervised model for statistically determining coordinate phrase attachment. In *Association for Computational Linguistics*, pp. 610–601, University of Maryland, 1999.

[24] D. Gusfield. *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press, 1997.

[25] R. Herbrich, T. Graepel, and C. Campbell. Bayes point machines. *Journal of Machine Learning Research*, 1:245–279, 2001.

[26] L. Hirschman, A. Yeh, C. Blaschke, and A. Valencia. Overview of biocreative: critical assessment of information extraction for biology. *BMC Bioinformatics*, 6 Suppl 1, 2005.

[27] T. W. I. A. Sag, G. Gazdar and S. Weisler. Coordination and how to distinguish categories. *Natural Language and Linguistic Theory*, 3:117–171, 1985.

[28] E. M. B. Ivan A. Sag, Thomas Wasow. *Syntactic Theory: A Formal Introduction*. CSLI, Stanford, CA, 2003.

[29] T. Joachims. Learning to align sequences: A maximum-margin approach. online manuscript, August 2003.

[30] J.-D. Kim, T. Ohta, Y. Tateisi, and J. Tsujii. GENIA corpus: a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(Suppl. 1):i180–i182, 2003.

[31] T. Kudo and Y. Matsumoto. Chunking with support vector machines. In *NAACL '01: Second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies 2001*, pp. 1–8, 2001.

[32] T. Kudo and Y. Matsumoto. A boosting algorithm for classification of semi-structured text. In *In Proc. of Empirical Methods in Natural Language Processing*, pp. 301–308, 2004.

[33] S. Kurohashi. Analyzing coordinate structures including punctuation in English. In *Proceedings of The Fourth International Workshop on Parsing Technologies*, pp. 136–147, 1995.

[34] S. Kurohashi and M. Nagao. Dynamic programming method for analyzing conjunctive structures in Japanese. In *COLING-92*, pp. 170–176, Nantes, France, 1992.

[35] S. Kurohashi and M. Nagao. A syntactic analysis method of long Japanese sentences based on the detection of conjunctive structures. *Computational Linguistics*, 20:507–534, 1994.

[36] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning (ICML-2001)*, pp. 282–289. Morgan Kaufmann, 2001.

[37] D. D. Lewis. Data extraction as text categorization: an experiment with the muc-3 corpus. In *MUC3 '91: Proceedings of the 3rd conference on Message understanding*, pp. 245–255, 1991.

[38] Y. Li, H. Zaragoza, R. Herbrich, J. Shawe-Taylor, and J. S. Kandola. The perceptron algorithm with uneven margins. In *ICML '02: Proceedings of the Nineteenth International Conference on Machine Learning*, pp. 379–386, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[39] A. McCallum, K. Bellare, and F. Pereira. A conditional random field for discriminatively-trained finite-state string edit distance. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI-2005)*, 2005.

[40] P. Nakov and M. Hearst. Using the web as an implicit training set: application to structural ambiguity resolution. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language (HLT/EMNLP)*, pp. 835–842, Vancouver, 2005.

[41] A. Okumura and K. Muraki. Symmetric pattern matching analysis for English coordinate structures. In *Proceedings of the Fourth Conference on Applied Natural Language Processing*, pp. 41–46, 1994.

[42] I. C. on Harmonization of Technical Requirements for Registration of Pharmaceuticals for Human Use. *ICH E9: Statistical Principles for Clinical Trials*. 1998.

[43] L. Pachter and B. Sturmfels. Parametric inference for biological sequence analysis. *Proceedings of the National Academy of Sciences of the USA*, 101(46):16138–16143, 2004.

[44] J. Pestian, C. Brew, P. Matykiewicz, D. Hovermale, N. Johnson, K. B. Cohen, and W. Duch. A shared task involving multi-label classification of clinical free text. In ACL ed., *Proceedings of ACL BioNLP*, Prague, June 2007. Association for Computational Linguistics.

[45] P. Resnik. Semantic similarity in a taxonomy: an information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 11:95–130, 1999.

[46] E. Riloff and W. Lehnert. Information extraction as a basis for high-precision text classification. *ACM Trans. Inf. Syst.*, 12(3):296–333, 1994.

[47] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

[48] D. Sackett, W. Rosenberg, J. Gray, R. Haynes, and W. Richardson. Evidence based medicine: what it is and what it isn't. *BMJ*, 312(7023):71–2, 1996.

[49] P. Schachter. Constraints on coordination. *Language*, 53:86–103, 1977.

[50] B. Scholkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[51] W. Schuette, T. Blankenburg, W. Guschall, I. Dittrich, M. Schroeder, H. Schweisfurth, A. Chemaissani, C. Schumann, N. Dickgreber, T. Appel, and D. Ukena. Multicenter randomized trial for stage iiib/iv non-small-cell lung cancer using every-3-week versus weekly paclitaxel/carboplatin. *Clin Lung Cancer*, 7(5):338–43, 2006.

[52] F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of the Human Language Technology Conference North American Chapter of Association for Computational Linguistics (HLT-NAACL 2003)*, pp. 213–220, Edmonton, Alberta, Canada, 2003. Association for Computational Linguistics.

[53] S. Straus, W. Richardson, P. Glasziou, and R. Haynes. *Evidence-based medicine: how to practice and teach, third edition*. Elsevier Churchill Livingstone, 2005.

[54] F. Sun, D. Fernández-Baca, and W. Yu. Inverse parametric sequence alignment. *Journal of Algorithms*, 53:36–54, 2004.

[55] C. Sutton and A. McCallum. *An Introduction to Conditional Random Fields for Relational Learning*. MIT Press, 2006.

[56] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, p. 104, New York, NY, USA, 2004. ACM.

[57] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research (JMLR)*, 6:1453 – 1484, September 2005.

[58] O. Uzuner, Y. Luo, and P. Szolovits. Evaluating the state-of-the-art in automatic de-identification. *J Am Med Inform Assoc*, 2007.

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

[60] C.-N. Yu, T. Joachims, R. Elber, and J. Pillardy. Support vector training of protein alignment models. In *Proceeding of the International Conference on Research in Computational Molecular Biology (RECOMB)*, pp. 253–267, 2007.

[61] H. Yu and E. Agichtein. Extracting synonymous gene and protein terms from biological literature. *Bioinformatics*, pp. 340–349, 2003.

List of Publications

Journal Papers

1. Kazuo Hara, Masashi Shimbo, Yuji Matsumoto : "An Alignment-Based Method for Learning Coordinate Phrases in Biomedical Abstracts", *Transactions of the Japanese Society for Artificial Intelligence*, Vol. 22, No. 3, pp. 248-255, May 2007. (in Japanese).

2. Kazuo Hara and Yuji Matsumoto: "Extracting clinical design information from MEDLINE abstracts", *New Generation Computing*, Vol. 25, No. 4, pp. 263-275, August 2007.

International Conference/Workshop Papers

1. Kazuo Hara, Yuji Matsumoto: "Information Extraction and Sentence Classification applied to Clinical Trial MEDLINE Abstracts", *In Proceedings of the 2005 International Joint Conference of InCoB, AASBi and KSBI (BIOINFO 2005)*, pp. 85-90, Busan, September 2005.

2. Kazuo Hara, Masashi Shimbo, Yuji Matsumoto : "Towards Biomedical Information Extraction Using Local Alignments", *The International Workshop on Data-Mining and Statistical Science (DMSS 2006)*, pp. 167-172, Sapporo, September 2006. (in Japanese).

3. Kazuo Hara: "Applying a SVM Based Chunker and a Text Classifier to the Deid Challenge", *i2b2 Workshop on Challenges in Natural Language Processing for Clinical Data*, Washington, D.C., USA, November 2006. Available as a JAMIA on-line data supplement to the article; "Evaluating the State-of-the-Art in Automatic De-identification" by Ozlem Uzuner, Yuan Luo, and Peter Szolovits, June 2007.

4. Masashi Shimbo, Kazuo Hara: "A Discriminative Learning Model for Coordinate Conjunctions", *In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp.610-619, Prague, Czech, June 2007.

Other Publications

1. Kazuo Hara, Yuji Matsumoto: "Information Extraction from MEDLINE Abstracts of Clinical Trials", *IPSJ SIG Technical Reports*, 2004-ICS-138, pp.135-139, Hanoi, Vietnam, December 2004.

2. Kazuo Hara and Yuji Matsumoto: "Information Extraction and Sentence Classification Applied to Clinical Trial MEDLINE Abstracts", *In Proceedings of the 11th Annual Meeting of the Association for Natural Language Processing*, pp.101-104, March 2005. (in Japanese).