Learning Chinese Language Structures with Multiple Views

Weiwei Sun

Saarbrücken Graduate School of Computer Science
Department of Computational Linguistics

University of Saarland

Thesis for obtaining the title of *Doctor of Engineering* of the Faculties of Natural Sciences and Technology of Saarland University

Saarbrücken, Germany
April, 2012

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form in einem Verfahren zur Erlangung eines akademischen Grades vorgelegt.

Name:	Weiwei Sun
Date:	2012-4-17
Place:	Saarbrücken, Germany

Abstract

Motivated by the inadequacy of single view approaches in many areas in NLP, we study multi-view Chinese language processing, including word segmentation, part-of-speech (POS) tagging, syntactic parsing and semantic role labeling (SRL), in this thesis. We consider three situations of multiple views in statistical NLP: (1) Heterogeneous computational models have been designed for a given problem; (2) Heterogeneous annotation data is available to train systems; (3) Supervised and unsupervised machine learning techniques are applicable.

First, we comparatively analyze successful single view approaches for Chinese lexical, syntactic and semantic processing. Our analysis highlights the diversity between heterogenous systems built on different views, and motivates us to improve the state-of-the-art by combining or integrating heterogeneous approaches. Second, we study the annotation ensemble problem, i.e. learning from multiple data sets under different annotation standards. We propose a series of generalized stacking models to effectively utilize heterogeneous labeled data to reduce approximation errors for word segmentation and parsing. Finally, we are concerned with bridging the gap between unsupervised and supervised learning paradigms. We introduce feature induction solutions that harvest useful linguistic knowledge from large-scale unlabeled data and effectively use them as new features to enhance discriminative learning based systems.

For word segmentation, we present a comparative study of word-based and character-based approaches. Inspired by the diversity of the two views, we design a novel stacked sub-word tagging model for joint word segmentation and POS tagging, which is robust to integrate different models, even models trained on heterogeneous annotations. To benefit from unsupervised word segmentation, we derive expressive string knowledge from unlabeled data which significantly enhances a strong supervised segmenter.

For POS tagging, we introduce two linguistically motivated improvements:

(1) combining syntax-free sequential tagging and syntax-based chart parsing results to better capture syntagmatic lexical relations and (2) integrating word clusters acquired from unlabeled data to better capture paradigmatic lexical relations.

For syntactic parsing, we present a comparative analysis for generative PCFG-LA constituency parsing and discriminative graph-based dependency parsing. To benefit from the diversity of parsing in different formalisms, we implement a previously introduced stacking method and propose a novel Bagging model to combine complementary strengths of grammar-free and grammar-based models. In addition to the study on the syntactic formalism, we also propose a reranking model to explore heterogenous treebanks that are labeled under different annotation scheme. Finally, we continue our efforts on combining strengths of supervised and unsupervised learning, and evaluate the impact of word clustering on different syntactic processing tasks.

Our work on SRL focus on improving the full parsing method with linguistically rich features and a chunking strategy. Furthermore, we developed a partial parsing based semantic chunking method, which has complementary strengths to the full parsing based method. Based on our work, Zhuang and Zong [2010] successfully improve the state-of-the-art by combining full and partial parsing based SRL systems.

Zusammenfassung

Motiviert durch die Unzulänglichkeit der Ansätze mit dem einzigen Ansicht in vielen Bereichen in NLP, untersuchen wir Chinesische Sprache Verarbeitung mit mehrfachen Ansichten, einschließlich Wortsegmentierung, Part-of-Speech (POS)-Tagging und syntaktische Parsing und die Kennzeichnung der semantische Rolle (SRL) in dieser Arbeit. Wir betrachten drei Situationen von mehreren Ansichten in der statistischen NLP: (1) Heterogene computergestützte Modelle sind für ein gegebenes Problem entwurft, (2) Heterogene Annotationsdaten sind verfügbar, um die Systeme zu trainieren, (3) überwachten und unüberwachten Methoden des maschinellen Lernens sind zur Verfügung gestellt.

Erstens, wir analysieren vergleichsweise erfolgreiche Ansätze mit einzigen Ansicht für chinesische lexikalische, syntaktische und semantische Verarbeitung. Unsere Analyse zeigt die Unterschiede zwischen den heterogenen Systemen, die auf verschiedenen Ansichten gebaut werden, und motiviert uns, die state-of-the-Art durch die Kombination oder Integration heterogener Ansätze zu verbessern. Zweitens, untersuchen wir die Annotation Ensemble Problem, d.h. das Lernen aus mehreren Datensätzen unter verschiedenen Annotation Standards. Wir schlagen eine Reihe allgemeiner Stapeln Modelle, um eine effektive Nutzung heterogener Daten zu beschriften, und um Approximationsfehler für Wort Segmentierung und Analyse zu reduzieren. Schließlich sind wir besorgt mit der Überbrückung der Kluft zwischen unüberwachten und überwachten Lernens Paradigmen. Wir führen Induktion Feature-Lösungen, die nützliche Sprachkenntnisse von großflächigen unmarkierter Daten ernte, und die effektiv nutzen als neue Features, um die unterscheidenden Lernen basierten Systemen zu verbessern.

Für die Wortsegmentierung, präsentieren wir eine vergleichende Studie der Wort-basierte und Charakter-basierten Ansätzen. Inspiriert von der Vielfalt der beiden Ansichten, entwerfen wir eine neuartige gestapelt Sub-Wort-Tagging-Modell für gemeinsame Wort-Segmentierung und POS-Tagging, die robust ist, um verschiedene Modelle zu integrieren, auch Modelle auf heterogenen Annotationen geschult. Um den unbeaufsichtigten Wortsegmentierung zu profitieren, leiten wir ausdrucksstarke Zeichenfolge Wissen

von unmarkierten Daten. Diese Methode hat eine überwachte Methode erheblich verbessert.

Für POS-Tagging, führen wir zwei linguistisch motiviert Verbesserungen: (1) die Kombination von Syntaxfreie sequentielle Tagging und Syntaxbasierten Grafik-Parsing-Ergebnisse, um syntagmatische lexikalische Beziehungen besser zu erfassen (2) die Integration von Wortclusteren von nicht markierte Daten, um die paradigmatische lexikalische Beziehungen besser zu erfassen.

Für syntaktische Parsing präsentieren wir eine vergleichenbare Analyse für generative PCFG-LA Wahlkreis Parsing und diskriminierende Graphenbasierte Abhängigkeit Parsing. Um aus der Vielfalt der Parsen in unterschiedlichen Formalismen zu profitieren, setzen wir eine zuvor eingeführte Stacking-Methode und schlagen eine neuartige Schrumpfbeutel-Modell vor, um die ergänzenden Stärken der Grammatik und Grammatik-free-basierte Modelle zu kombinieren. Neben dem syntaktischen Formalismus, wir schlagen auch ein Modell, um heterogene reranking Baumbanken, die unter verschiedenen Annotationsschema beschriftet sind zu erkunden. Schließlich setzen wir unsere Bemühungen auf die Bündelung von Stärken des überwachten und unüberwachten Lernen, und bewerten wir die Auswirkungen der Wort-Clustering auf verschiedene syntaktische Verarbeitung Aufgaben.

Unsere Arbeit an SRL ist konzentriert auf die Verbesserung der vollen Parsingsmethode mit linguistischen umfangreichen Funktionen und einer Chunkingstrategie. Weiterhin entwickelten wir eine semantische Chunkingmethode basiert auf dem partiellen Parsing, die die komplementäre Stärken gegen die die Methode basiert auf dem vollen Parsing hat. Basiert auf unserer Arbeit, Zhuang and Zong [2010] hat den aktuelle Stand erfolgreich verbessert durch die Kombination von voll-und partielle-Parsing basierte SRL Systeme.

Acknowledgements

First and foremost, I thank my advisor, Prof. Hans Uszkoreit. Prof. Uszkoreit was personally responsible for bringing me to Saarbrücken, and offered exceptional academic freedom during my study while at the same time made sure I was on the right track. His comments on my work have improved both my results and the readability of the dissertation.

I am also indebted to many collaborators during my PhD study, especially Yi Zhang, Rui Wang and Jia Xu. They helped me much not only in research but also in daily life. I thank Regine Bader for proofreading my ACL-2011 paper. I would like to acknowledge our secretaries, Christina Deeg, Corinna Johanns and Sandra Loch. They help me a lot to organize many things.

I thank my parents for their unconditional love and endless support throughout all these years. I dedicate this thesis to them.

Finally, I would like to thank the graduate school of computer science for its support during the submission of this dissertation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The application of statistical learning techniques to natural language processing (NLP) has been remarkably successful over the past two decades. The wide availability of linguistic corpora has played a critical role in their success, but acquiring sufficient quantities of usefully labeled training examples is still a major bottleneck for many supervised NLP algorithms. Two promising methods to address the annotation bottleneck are *co-training*, a variant of semi-supervised learning, in which two (or more) learners label most reliable pseudo training examples for each other, and *co-testing*, a variant of active learning, in which two (or more) learners find training examples that are the most informative by disagreement for the human to label. There are some successful application of both co-training and co-testing for several NLP tasks, such as syntactic parsing [Hwa et al., 2003; Osborne and Baldridge, 2004; Sarkar, 2001] and information extraction [Collins and Singer, 1999; Liao and Grishman, 2011].

The success of both co-training and co-testing results from multiple sufficiently independent views, based on which labeled and unlabeled data are explored by mutually training a set of classifiers defined in each view. These two *multi-view learning* approaches are inspired by a general principle: The agreement rate of multiple hypotheses based on different views lower-bounds the error rate of either hypothesis. By maximizing the agreement rate, the error rate can be minimized. Broadly speaking, multi-view learning is not restricted to the semi-supervised and active learning cases, and has, explicitly or implicitly, been applied in many other very different approaches, although many authors do not seem to be aware of the multi-view aspect. By utilizing the consensus maximization principle, multi-view learning can be advantageous when compared to learning with only a single view especially when the weaknesses of one view complement the strengths of the other.

Previous research efforts on NLP have mainly focused on designing and developing individual models with single views, and have met with success in a majority of tasks: word segmentation, POS tagging, syntactic parsing, machine translation, just to name a few. Normally, there have been a considerable number of methods to resolve one problem. Take English Penn Treebank guided parsing—one of the biggest breakthroughs in the last two decades—for example. First, different grammar formalisms, e.g. probabilistic context-free grammars [Charniak, 2000; Collins, 2003; Klein and Manning, 2003], tree-adjoining grammars [Carreras et al., 2008; Chiang, 2000; Shen and Joshi, 2005], and dependency grammars [Eisner, 1996; McDonald et al., 2005; Yamada and Matsumoto, 2003], are explored. Second, in one grammar formalism, different approaches are proposed and well implemented. For example, based on the context free grammar, both lexicalized models, such as Collins and Charniak parsers [Charniak, 2000; Collins, 2003], and unlexicalized models, such as the Berkeley parser [Petrov et al., 2006; Petrov and Klein, 2007] achieve good results. Finally, even in the same approach, there are still different strategies to follow. For instance, both shift-reduce [Sagae and Lavie, 2006a] and cascaded chunking [Tsuruoka et al., 2009] models are evaluated for incremental parsing.

Single views could be adequate if for every processing task, we could have a few perfect views that could be precise enough. Unfortunately, we are not even close to finding such perfect views, if they do exist. On the other hand, NLP systems built on particular single views normally capture different properties of an original problem, and therefore differ in predictive powers. Moreover, our motivating examples, i.e. co-training and co-testing, suggest that multi-view processing is an encouraging solution since it may combines complementary strengths of different views. In this thesis, we empirically investigate learning natural language structures with *multiple views*. Traditional multi-view learning describes the setting of learning from data where observations are represented by multiple independent sets of features. Different from such setting, we extend the idea and study different views with respect to computational models, linguistic annotations and learning paradigms, which are three typical situations in NLP. We claim that multiple, distinct, heterogeneous views are needed to process natural languages in all levels, to efficiently construct linguistic resources, to enhance existing computational models, as well as to inspire novel ideas.

This thesis focuses on the automatic statistical processing of the Chinese language. The process of comprehending a Chinese sentence involves structuring a sequence of characters lexically, syntactically and semantically to arrive at a representation of the sentence's meaning. The problems studied in this thesis include word segmentation, POS tagging, constituency as well as dependency parsing, and semantic role labeling. First, we use existing approaches as a starting point to comparatively analyze currently successful models for lexical, syntactic and semantic processing. The results of analysis is further exploited to advance the state-of-the-art by combining or integrating heterogeneous methods. Second, we study the annotation ensemble problem, i.e. learning from multiple data sets created according to different annotation standards. We propose *generalized stacking* models to effectively utilize heterogeneous labeled data. Finally, we are concerned with bridging the gap between unsupervised and supervised learning paradigms. We introduce *feature induction* solutions that harvest useful linguistic knowledge from large-scale unlabeled data and effectively use them as new features to enhance language processing systems based on discriminative learning.

1.1 About Multiple Views in NLP

1.1.1 A General Framework for Data-driven Text Processing

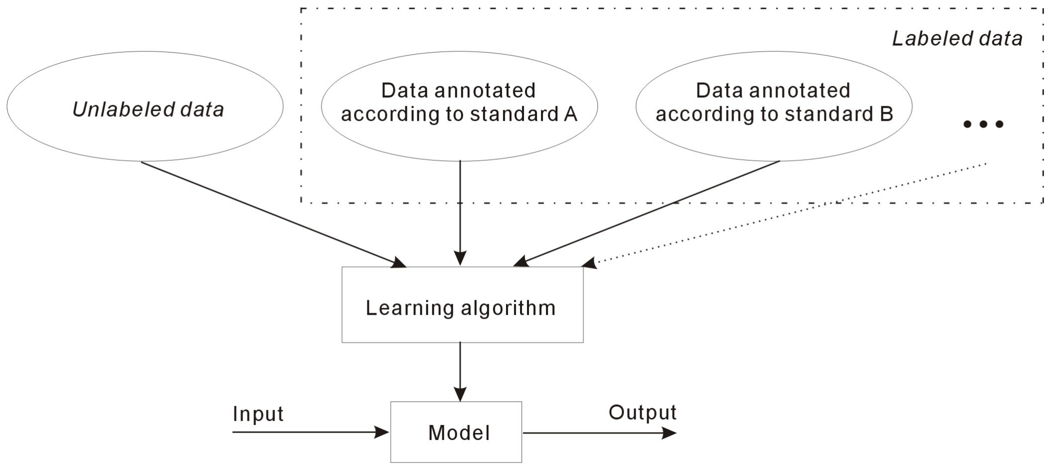

Labeled data
Unlabeled data
Data annotated according to standard A
Data annotated according to standard B
...
Learning algorithm
Input
Model
Output

Figure 1.1: Outline of generic data-driven NLP framework.

Figure 1.1 graphically displays the general framework for most data-driven NLP systems. First, a system must define a *learning algorithm* that takes as input the *training data*, which is either labeled texts with informative linguistic structures or unlabeled raw texts, and outputs a *model*. Sometimes, multiple heterogeneous annotated corpora that are labeled according to different standards are available. The learned model is the main part of a NLP system. When a new sentence x is given to the NLP system, the system uses the parameter specifications in the model to produce a linguistic structure y.

1.1.2 Three Categories of Multi-views

According to the aforementioned general framework, we distinguish between different *view*s for data-driven NLP from three aspects: computational model, linguistic annotation and learning paradigm. First, given a training data set, different models can be taken as different views. Second, for one type of task, heterogeneous annotations that yield predictors with different outputs can be taken as different views. Finally, different learning paradigms, namely unsupervised language acquisition and supervised language processing, can be taken as different views.

Heterogeneous Computational Models: For a number of NLP tasks, distinct, heterogeneous models have been proposed for solutions, each of which is based on a particular view of a problem. Take Chinese word segmentation for example. There are two dominant approaches for word segmentation. The first one naturally formulates the problem as finding words contained in a given sentence one-by-one. The second one transforms the segmentation problem into a character classification problem, of which the target classes are word boundary labels. The different start points make the two types of models behave very differently, and especially have different error distributions.

Heterogeneous Linguistic Annotations: A majority of data-driven NLP systems rely on large-scale, manually annotated corpora. These corpora with considerable information are important to train statistical systems but very expensive to build. Nowadays, for many tasks, multiple heterogeneous annotated corpora have been built and publicly available. For Chinese lexical processing, both the PKU's People's Daily data and the Penn Chinese Treebank provide word boundary and POS information. For parsing, the HIT-IR dependency Treebank and the Tsinghua Treebank are two alternative corpora of the Penn Treebank. However, the annotation *formalisms* or *schemes* in different projects are usually different, since the underlying linguistic theories vary and have different ways to explain the same language phenomena. Each linguistic annotation standard with its associated labeled corpus can be taken as a particular view.

Heterogeneous Learning Paradigms: In data-driven NLP, we are interested in automatically learning something from labeled and unlabeled corpora. Generally speaking, typical approaches in NLP to the natural language learning problem fall into two categories. In the first case, we basically build strong systems on labeled data by applying highly developed supervised training techniques. In the second case, we acquire linguistic knowledge from raw texts in a primarily unsupervised fashion. Both achieve encouraging results for some tasks. A good example for the first case is syntactic parsing, while a good example for the second case is word alignment. These two learning paradigms can be conventionally taken as two views for learning language structures.

1.1.3 Advantages of Multi-view Processing

On one hand, each view alone can yield a reasonably good predictor in many cases, but is inadequate to interpret every linguistic phenomenon. On the other hand, some linguistic properties that are not captured by one model, can be potentially captured by other models. As a result, NLP systems can take advantage of complementary strengths of multiple views. Below we sketch some possible directions.

- **Model Selection:** Comparative analysis of different views gives better understanding of the goodness and badness of different solutions. This can help select an appropriate solution from a set of candidate models for a particular problem.
- **Inspiration for Novel Models:** Comparative analysis can (at least partially) interpret why a particular view works or does not work for a given task. This can help us design new models by overcoming the shortcomings of each view.
- **System Ensemble:** We could use multiple systems to obtain better predictive performance than could be obtained from any of the constituent systems. Multiple views can increase the diversity of base systems which is of the central role of system ensemble.
- **Agreement-based semi-supervised Learning:** Agreement-based semi-supervised learning is a general approach for learning from both labeled and unlabeled data, where the agreement among multiple learners is exploited for learning. This general approach can benefit from multiple views to explore unlabeled data, or benefit from unlabeled data to explore the diversity of multiple views.

- **Disagreement-based Active Learning:** Similar to agreement-based semi-supervised learning, multiple views can also help detect informative examples by disagreement for active learning and therefore reduce labeling efforts for corpus construction.

1.1.4 View Integration

Apart from achieving better understanding of Chinese language processing through comparative study, we are also (or even more) interested in enhancing individual systems by combining or integrating different views. There are many conceivable ways to do so. In this thesis, we present a unified framework for view integration. Each view alone can yield a predictor that can be taken as a mechanism to produce morphology, lexical, syntactic or semantic structures for given texts. With different views, we can construct multiple heterogeneous systems. These systems may produce the same type of linguistic analysis but with different error distributions, may produce similar linguistic analysis which holds the same high level linguistic principles but differs in details, or just produce some non-directly comparable but relevant information. We leverage post-inference to integrate the outputs from systems designed by single views. This framework is general and robust, in the sense that we assume almost nothing about the individual systems and take them as black boxes.

Formally speaking, our idea is to include two "levels" of processing. The first level includes one or more predictors $f^m_1, ..., f^m_{K_m}, f^a_1, ..., f^a_{K_a}, f^l_1, ..., f^l_{K_l}$ that are independently built on different views. When views are about different computational models, the associated predictors $f^m_1, ..., f^m_{K_m}$ have the same input space $\mathcal{X}$ and the same output space $\mathcal{Y}$. When views are about different linguistic annotations or learning paradigms, the associated predictors $f^a_1, ..., f^a_{K_a}, f^l_1, ..., f^l_{K_l}$ have the same input space $\mathcal{X}$ but different output spaces $\mathcal{Y}^a_1, ..., \mathcal{Y}^a_{K_a}, \mathcal{Y}^l_1, ..., \mathcal{Y}^l_{K_l}$. The second level processing consists of an inference function h that takes as input $\langle \mathbf{x}, f^m_1(\mathbf{x}), ..., f^m_{K_m}(\mathbf{x}), f^a_1(\mathbf{x}), ..., f^a_{K_a}(\mathbf{x}), f^l_1(\mathbf{x}), ..., f^l_{K_l}(\mathbf{x})\rangle$ and outputs a final prediction $h(\mathbf{x}, f^m_1(\mathbf{x}), ..., f^l_{K_l}(\mathbf{x}))$.

Learning-free Inference: When the relations between heterogeneous systems can be explicitly expressed as hard constraints, i.e. the outputs are directly comparable, we can use a learning-free post-inference to pick up good analysis from the output pool. By *learning-free*, we mean no machine learning procedure is involved in the selection of outputs. The simplest case is the combination of heterogeneous methods, where the target outputs of different systems are exactly the same. A more complex example is the combination of a constituency parser and a dependency parser. If we assume that a constituency structure is adequate to be transformed to its associated dependency structure, we can still use a word-by-word voting method to combine the results from a dependency parser with the transformed dependency results from a constituency parser to achieve better dependency analyzing.

Learning-based Inference: When the relations between heterogeneous systems are hard to be expressed as constraints, i.e. the outputs are not comparable, we can still use discriminative learning techniques to integrate their outputs. The discriminative nature of a learning-based post-inference procedure allows it to define arbitrary features from rich heterogeneous structures, and to automatically identify and explore informative features for output selection.

Table 1.1 lists some properties of the three categories of heterogeneous views. When the views are about computational models, the output spaces are the same one, so the outputs of different systems are directly comparable. In this case, we can apply both learning-based and learning-free inference to get the final result. When the views are about linguistic annotations, the output spaces are not directly comparable. In this case, we can apply learning-based inference to get the final result. If the assistant annotations are adequate to be converted to the target annotation, we can still employ learning-free inference by first transforming between heterogeneous annotations. When the views are about different machine learning paradigms, the output spaces are not comparable. In this case, only learning-based inference is applicable. When the supervised processing system is itself a discriminative one, we can directly incorporate the heterogeneous linguistic knowledge into the system as new features.

View category	Output	Post-inference	
		Learning-based	Learning-free
Computational model	Directly comparable	YES	YES
Linguistic annotation	Not directly comparable	YES	?
Learning paradigm	Not comparable	YES	NO

Table 1.1: Properties of different views in NLP.

1.2 The Problems Investigated in This Thesis

In this thesis, we focus on multi-view Chinese language processing, including learning lexical, syntactic and shallow semantic structures. Before we move on to the main body of the thesis, we give a brief introduction to the tasks we investigate.

Word Segmentation: The Chinese language has a number of characteristics that make Chinese language processing particularly challenging and intellectually rewarding. For example, written Chinese text does not have marked word boundaries like English and other Western languages. To find the basic language units, word segmentation, of which the goal is to transform a Chinese sentence from a character sequence to a word sequence, is a necessary initial step for Chinese language processing.

Syntactic Parsing: As one of the core issues of NLP, syntactic parsing is the process of analyzing a sequence of words to determine its grammatical structure. Two popular formalisms to express syntactic relations are constituency and dependency representations. A constituency grammar arranges sentences into a hierarchy of nested phrases which determine the construction of each phrase, while a dependency grammar formalizes syntactic structure as a directed tree of bilexical dependencies, which determines relations between head words and their dependents. It is generally accepted that finding syntactic structures is useful in determining the meaning of a sentence. Therefore most NLP applications could certainly benefit from high-accuracy parsing.

Semantic Role Labeling: Semantic role labeling (SRL) is the process of assigning semantic roles to constituents in a sentence according to their relationship to predicates expressed in the sentence. It consists of the detection of the semantic arguments associated with a target predicate and their classification into their specific roles. Such sentence-level semantic analysis of text is concerned with the characterization of events and is therefore important to understand the essential meaning of natural language sentences – *who* did *what* to *whom*, for *whom* or *what*, *how*, *where*, *when* and *why*?

1.3 Main Contributions

We present a series of theoretical and empirical comparative analysis for a number of state-of-the-art heterogeneous methods to resolve the Chinese language structure learning problems, including,

- word-based and character-based methods for word segmentation,
- discriminative sequential tagging and generative chart parsing methods for POS tagging,

- generative PCFG-LA models and discriminative graph-based methods for constituency and dependency parsing,
- full and shallow parsing based methods for SRL.

Generally, to get a good hybrid solver, the component learners should be as more accurate as possible, and as diverse as possible. Our comparative analysis highlights the diversity between different systems built on different views, and therefore motivates our research on view integration. Amongst many conceivable ways, this thesis specially focus on the aforementioned *post-inference* method to combine outputs produced by heterogeneous systems. We propose several effective models and improve state-of-the-art accuracy for Chinese language processing, including,

- a Bagging model to combine word-based and character-based word segmentation methods,
- a stacked sub-word tagging model for joint word segmentation and POS tagging, which is robust to integrate not only heterogeneous models, but also heterogeneous annotation data,
- a Bagging model for POS tagging which combine the complementary strengths of syntax-free sequential tagging and syntax-based chat parsing,
- a Bagging model for dependency parsing which combine the complementary strengths of a discriminative, grammar-free parser and a generative, grammar-based parser,
- a parse reranking model to explore heterogeneous treebanks for constituency parsing.

The last topic we explore is *unsupervised language acquisition for supervised language processing*. Rather than utilize an extra post-inference, we directly incorporate linguistic knowledge acquired from unlabeled data into discriminative processors. Our work includes,

- deriving useful and expressive string knowledge from unlabeled data to enhance strong supervised word segmenters,
- utilizing word clusters to improve different syntactic processing tasks, including POS tagging, text chunking and dependency parsing.

1.4 Outline of the Thesis

The remainder of this thesis consists of three main parts and is structured as follows.

In the first part, we present our work on word segmentation.

- Chapter 2 provides a comparative study of two dominant model views for word segmentation.
- Chapter 3 presents a stacked sub-word tagging model for joint word segmentation and POS tagging, which is robust to integrate different models, even models trained on heterogeneous annotations.
- Chapter 4 describes a semi-supervised method to enhance a supervised word segmenter via harvesting string knowledge from unlabeled data.

Then, we introduce our work on syntactic parsing in the second part.

- Chapter 5 provides a comparative study for POS tagging and syntactic parsing in different formalisms.
- Chapter 6 exploits heterogenous treebanks to improve constituency parsing.
- Chapter 7 applies unsupervised lexical acquisition to POS tagging, text chunking and dependency parsing.

The third part is our work on shallow semantic parsing.

- Chapter 8 introduces full and partial parsing based semantic chunking methods for SRL and presents a comparative analysis.

Chapter 9 concludes by summarizing the thesis and providing ideas for future work.

Part I

Word Segmentation

Chapter 2

Comparing and Combining Word-based and Character-based Segmenters

This chapter introduces the Chinese word segmentation problem, which is a fundamental task of Chinese language processing. Supervised learning with specifically defined training data has become a dominant paradigm. Our discussion is under this setting. We present a theoretical and empirical comparative analysis of the two dominant categories of approaches in word segmentation: word-based models and character-based models. From a machine learning view, the two approaches formulate segmentation as semi-Markov and Markov tagging respectively. We show that, in spite of similar performance overall, the two models produce different distributions of segmentation errors, in a way that can be explained by theoretical properties of the two models. The analysis is further exploited to improve segmentation accuracy by integrating a word-based segmenter and a character-based segmenter.

Parts of this chapter are originally published in [Sun, 2010c].

2.1 Background

2.1.1 The Problem

In language, words are relatively independent carriers of meaning that can be codified in the lexicon and be described syntactically as the smallest substitutable units of a sentence. Chinese sentences are written in continuous sequences of characters without space characters as explicit word delimiters. To find the basic language units, word segmentation, of which the goal is to transform a Chinese sentence from a character sequence to a word sequence, is a necessary intial step for Chinese language processing.

2.1.2 Previous Work

There are two dominant models for Chinese word segmentation. The first one is what we call "word-based" approach, where the basic predicting units are words themselves. This kind of segmenters sequentially decides whether the local sequence of characters make up a word. This word-by-word approach ranges from naive maximum matching [Chen and Liu, 1992] to complex solution based on semi-Markov conditional random fields (CRF) [Andrew, 2006]. The second is the "character-based" approach, where basic processing units are characters which compose words. Segmentation is formulated as a classification problem to predict whether a character locates at the beginning of, inside or at the end of a word. This character-by-character method was first proposed in [Xue, 2003], and a number of machine learning algorithms have been exploited, including maximum entropy classification [Ng and Low, 2004], structured perceptron [Jiang et al., 2009], CRFs [Tseng et al., 2005a], and discriminative latent variable CRFs [Sun et al., 2009b].

State-of-the-art segmenters nearly all leverage discriminative sequential tagging. It is easy to formulate the two kinds of methods as semi-Markov and Markov tagging problems. This chapter is concerned with the behavior of different segmentation models in general. We present a detailed analysis that reveals important differences of the two methods. First, we give a theoretical comparative analysis of the two models. Then we implement two statistical segmenters and empirically study several factors that influence the performance of the two types of algorithms. Our analysis will indicate that the two types of approaches exhibit different behaviors, and each segmentation model has strengths and weaknesses. We further consider integrating word-based and character-based models in order to exploit their complementary strengths and thereby improve segmentation accuracy beyond what is possible by either model in isolation. We present a *Bootstrap Aggregating* model to combine multiple segmentation systems.

2.2 State-of-the-Art

First of all, we distinguish two kinds of "words": (1) Words in dictionary are word types; (2) Words in sentences are word tokens. The goal of word segmentation is to identify word tokens in a running text, where a large dictionary (i.e. list of word types) and annotated corpora may be available. From the view of *token*, we divide segmentation models into two main categories: word-based models and character-based models. There are two key issues of a segmentation model: (1) How to decide whether a local sequence of characters is a word? (2) How to do disambiguation if ambiguous segmentation occurs? For each model, we separately discuss the strategies for word prediction and segmentation disambiguation.

2.2.1 Word-Based Method: Semi-Markov Tagging

It may be the most natural idea for segmentation to find word tokens one by one. This kind of segmenters read the input sentences from left to right, predict whether current piece of continuous characters is a word token. After one word is found, segmenters move on and search for next possible word. There are different strategies for the word prediction and disambiguation problems. Take for example maximum matching, which was a popular algorithm at the early stage of research [Chen and Liu, 1992]. For word prediction, if a sequence of characters appears in a dictionary, it is taken as a word candidate. For segmentation disambiguation, if more than one word types are matched, the algorithm chooses the longest one.

In the last several years, machine learning techniques are employed to improve word-based segmentation, where the above two problems are solved in a uniform semi-Markov tagging model. Given a sequence of characters $\mathbf{c} \in \mathcal{C}^n$ (n is the number of characters), denote a segmented sequence of words $\mathbf{w} \in \mathcal{W}^m$ (m is the number of words, i.e. m varies with $\mathbf{w}$), and a function GEN that enumerates a set of segmentation candidates GEN($\mathbf{c}$) for $\mathbf{c}$. In general, a segmenter solves the following "argmax" problem:

$$\begin{aligned}
\hat{\mathbf{w}} &= \arg\max_{\mathbf{w}\in\text{GEN}(\mathbf{c})} \theta^\top \Phi(\mathbf{c}, \mathbf{w}) && (2.1)\\
&= \arg\max_{\mathbf{w}\in\text{GEN}(\mathbf{c})} \theta^\top \sum_{i=1}^{|\mathbf{w}|} \phi(\mathbf{c}, w_{[1:i]}) && (2.2)
\end{aligned}$$

where Φ and ϕ are global and local feature maps and θ is the parameter vector to learn. The inner product $\theta^\top \phi(\mathbf{c}, w_{[1:i]})$ can been seen as the confidence score of whether w_i is a word. The disambiguation takes into account confidence score of each word, by using the sum of local scores as its criteria. Markov assumption is necessary for computation, so ϕ is usually defined on a limited history. Perceptron and semi-

Markov CRFs were used to estimate θ in previous work [Andrew, 2006; Zhang and Clark, 2007].

Our introduction here is according to the properties of segmentation models, not the research history. Note that the first disambiguation method with machine learning techniques was introduced under the character-based scheme.

2.2.2 Character-Based Method: Markov Tagging

Most previous data-driven segmentation solutions took an alternative, character-based view. This approach observes that by classifying characters as different positions in words, e.g. *word-initial*, *word-middle*, *word-final*, etc., segmentation can be treated as a Markov sequential tagging problem, assigning labels to the characters in a sentence indicating whether a character c_i is a single character word (S) or the begin (B), middle (I) or end (E) of a multi-character word. For word prediction, word tokens are inferred based on the character classes. For example, the target label representation of the book title "国家的囚犯：赵紫阳总理的秘密日记/Prisoner of the State: The Secret Journal of Premier Zhao Ziyang" is as follows.

国	家	的	囚	徒	：	赵	紫	阳	总	理	的	秘	密	日	记
B	E	S	B	E	S	B	I	E	B	E	S	B	E	B	E

The main difficulty of this model is character ambiguity that most Chinese characters can occur in different positions within different words. Linear models are also popular for character disambiguation (i.e. segmentation disambiguation). Denote a sequence of character labels $\mathbf{y} \in \mathcal{Y}^n$, a linear model is defined as:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}\in\mathcal{Y}^{|\mathbf{c}|}} \theta^\top \Psi(\mathbf{c}, \mathbf{y}) \tag{2.3}$$

$$= \arg\max_{\mathbf{y}\in\mathcal{Y}^{|\mathbf{c}|}} \theta^\top \sum_{i=1}^{|\mathbf{c}|} \psi(\mathbf{c}, y_{[1:i]}) \tag{2.4}$$

Note that local feature map ψ is defined only on the sequence of characters and their labels. Several discriminative models have been exploited for parameter estimation, including perceptron, CRFs, and discriminative latent variable CRFs [Jiang et al., 2009; Sun et al., 2009b; Tseng et al., 2005a].

2.3 Theoretical Comparison

Theoretically, the two types of models are different. We compare them under four aspects.

2.3.1 Internal Structure of Words

Chinese words have internal structures. In most cases, a Chinese character is a morpheme which is the smallest meaningful unit of the language. Though we cannot exactly infer the meaning of a word from its character components, the character structure is still meaningful [Sun et al., 2009a]. Partially characterizing the internal structures of words, one advantage of character-based models is the ability to induce new words. E.g., character "者/person" is usually used as a suffix meaning "one kind of people". If a segmenter never sees "工作者/worker" in training data, it may still rightly recognize this word by analyzing the prefix "工作/work" with label *BI* and the suffix "者" with label *E*. This feature may be helpful in CWS for generalizing to new words. In contrast, current word-based models only utilize the weighted features as word prediction criteria, and thus word formation information is not well explored. For more details about Chinese word fomation in NLP, see [Sun et al., 2009a].

2.3.2 Linearity and Nonlinearity

A majority of structured prediction models are linear models in the sense that the score functions are linear combination of parameters. Both previous solutions for word-based and character-based systems utilize linear models. However, both principly linear models incur nonlinearity to some extent. In general, a sequence classification itself involves nonlinearity because the features of current token usually encode previous state information which is a linear combination of features of previous tokens. The interested readers may consult [Liang et al., 2008] for preliminary discussion about the nonlinearity in structured models. This kind of nonlinearity exists in both word-based and character-based models. The word-based solution, such as word-based perceptron and semi-Markov CRFs, is a linear model. The word prediction acts on features in a linear way. In addition, in most character-based models, a word should take a *S* label or start with a *B* label, end with *E* label, and only have *I* label inside. This inductive way for word prediction actually behaves nonlinearly. Only strings with label sequence like *BI*E* or *S* are predicted as words.

2.3.3 Dynamic Tokens or Static Tokens

Since word-based models take the sum of part score of each individual word token, it increases the upper bound of the whole score to segment more words. As a result, word-based segmenter tends to segment words into smaller pieces. A difficult case occurs when a word token w consists of some sub-strings that cound be word tokens. In such cases a word-based segmenter more easily splits the word into individual words. For example, in the phrase "四千三百/4300 米/meter (4300 meters)", the numeral "四千三百" consists of two individual strings "四千 (4000)" and "三百(300)" which are numeral word typs. A word-based segmenter could more more easily make a mistake to segment two word tokens. This phenomenon is very common in named entities.

2.3.4 Word Token or Word Type Features

In character-based models, features are usually defined by the character information in the neighboring n-character window. Despite a large set of valuable features that could be expressed, it is slightly less natural to encode predicted word token information. On the contrary, taking words as dynamic tokens, it is very easy to define word token features in a word-based model. Word-based segmenters hence have greater representational power. Despite of the lack of word token representation ability, character-based segmenters can use word type features by looking up a dictionary. For example, if a local sequence of characters following current token matches a word in a dictionary; these word types can be used as features. If a string matches a word type, it has a very high probability (ca. 90%) to be a word token. So word type features are a good approximation of word token features.

2.4 Empirical Comparision

The primary purpose of this study is to characterize the different properties of the two methods. We present a series of experiments that relate segmentation performance to a set of properties of input words. We argue that the results can be correlated to specific theoretical aspects of each model.

2.4.1 Baseline Systems

For empirical analysis, we implement segmenters in word-based and character-based architectures respectively. We introduce them from three aspects: basic models, parameter estimation and feature selection.

2.4.1.1 Models

For both word-based and character-based segmenters, we use linear models introduced in the section above. We use first order (Semi-)Markov models for training and test. In particular, for word-based segmenter, the local feature map $\phi(\mathbf{c}, w_{[1:i]})$ is defined only on $\mathbf{c}, w_{i-1}$ and w_i, and thereby Eq. 2.2 is defined as

$$\hat{\mathbf{w}} = \arg \max_{\mathbf{w} \in \text{GEN}(\mathbf{c})} \theta^\top \sum_{i=1}^{|\mathbf{w}|} \phi(\mathbf{c}, w_{i-1}, w_i) \tag{2.5}$$

This model has a first-order Semi-Markov structure. For decoding, Zhang and Clark [2007] used a beam search algorithm to get approximate solutions, and Sarawagi and Cohen [2004] introduced a Viterbi style algorithm for exact inference. We use this exact inference algorithm in our segmenter at both training and test time.

For our character-based segmenter, the local feature map $\psi(\mathbf{c}, y_{[1:i]})$ is defined on $\mathbf{c}, y_{i-1}$ and y_i, and Eq. 2.4 is defined as

$$\hat{\mathbf{y}} = \arg \max_{\mathbf{y} \in \mathcal{Y}^{|\mathbf{c}|}} \theta^\top \sum_{i=1}^{|\mathbf{c}|} \psi(\theta, y_{i-1}, y_i) \tag{2.6}$$

In our character-based segmenter, we also use a Viterbi algorithm for decoding.

2.4.1.2 Learning

We adopt *Passive-Aggressive* (PA) framework [Crammer et al., 2006], a family of margin based online learning algorithms, for the parameter estimation. It is fast and easy to implement. Algorithm 3 illustrates the learning procedure. The parameter vector $\mathbf{w}$ is initialized to $(0, ..., 0)$. A *PA* learner processes all the instances (t is from 1 to n) in each iteration (I). If current hypothesis ($\mathbf{w}$) fails to predict $\mathbf{x}_t$, the learner updates $\mathbf{w}$ through calculating the loss l_t and the difference between $\Phi(\mathbf{x}_t, \mathbf{y}_t^*)$ and $\Phi(\mathbf{x}_t, \mathbf{y}_t)$ (line 5-7). There are three variants in the update step. We here only present the PA-II rule[1], which performs best in our experiments. This update step is based on

[1]See the original paper for more details.

```
  input : Data $\{(\mathbf{x}_t, \mathbf{y}_t), t = 1, 2, ..., n\}$
1 Initialize: $\mathbf{w} \leftarrow (0, ..., 0)$
2 for $I = 1, 2, ...$ do
3     for $t = 1, ..., n$ do
4         Predict: $\mathbf{y}_t^* = \arg\max_{\mathbf{y} \in \text{GEN}(\mathbf{x}_t)} \mathbf{w}^\top \Phi(\mathbf{x}_t, \mathbf{y})$
5         Suffer loss: $l_t = \rho(\mathbf{y}_t, \mathbf{y}_t^*) + \mathbf{w}^\top \Phi(\mathbf{x}_t, \mathbf{y}_t^*) - \mathbf{w}^\top \Phi(\mathbf{x}_t, \mathbf{y}_t)$
6         Set: $\tau_t = \frac{l_t}{||\Phi(\mathbf{x}_t, \mathbf{y}_t^*) - \Phi(\mathbf{x}_t, \mathbf{y}_t)||^2 + 0.5C}$
7         Update: $\mathbf{w} \leftarrow \mathbf{w} + \tau_t(\Phi(\mathbf{x}_t, \mathbf{y}_t) - \Phi(\mathbf{x}_t, \mathbf{y}_t^*))$
8     end
9 end
```

Algorithm 1: The *PA* learning procedure.

analytical solutions to simple constrained optimization problems. For more details, please refer to the original paper.

The PA algorithm utilizes a paradigm of cost-sensitive learning to resolve structured prediction. A cost function ρ is necessary to calculate the loss l_t (line 5). For every pair of labels $(\mathbf{y}^*, \mathbf{y})$, users should define a cost $\rho(\mathbf{y}^*, \mathbf{y})$ associated with predicting $\mathbf{y}^*$ when the correct label is $\mathbf{y}$. ρ should be defined differently for different purposes. There are two natural costs for segmentation: (1) sum of the number of wrong and missed word predictions and (2) sum of the number of wrongly classified characters. We tried both cost functions for both models. We find that the first one is suitable for a word-based segmenter and the second one is suitable for a character-based segmenter. We do not report segmentation performance for the weeker combination of the problem structure and the cost function in this thesis. C (in line 6) is the slack variable. In our experiments, the segmentation performance is not sensitive to C. In the following experiments, we set $C = 1$.

2.4.1.3 Features

Developing features has been shown crucial to advancing the state-of-the-art statistical solutions of in a number of NLP tasks. It is also important for word segmentation. The features used in our segmenters are mainly borrowed from previous research, which are elaborated on in the following.

Word-based Segmenter

For the convenience of illustration, we denote a candidate word token w_i with a context $c_{j-1}[_{w_{i-1}} c_j ... c_k][_{w_i} c_{k+1} ... c_l] c_{l+1}$.

The character features includes,

Boundary character unigram: c_j, c_k, c_{k+1}, c_l and c_{l+1}; *Boundary character bigram*: $c_k c_{k+1}$ and $c_l c_{l+1}$.

Inside character unigram: c_s $(k+1 < s < l)$; *Inside character bigram*: $c_s c_{s+1}$ $(k+1 < s < l)$.

Length of current word.

Whether c_{k+1} and c_{k+1} are *identical.*

Combination Features: c_{k+1} and c_l,

The word token features includes,

Word Unigram: previous word w_{i-1} and current word w_i; *Word Bigram*: $w_{i-1}w_i$.

The *identity* of w_i, if it is a *Single character word.*

Combination Features: w_{i-1} and length of w_i, w_i and length of w_{i-1}. c_{k+1} and length of w_i, c_l and length of w_i.

Character-based Segmenter

We use the exact same feature templates described in [Sun et al., 2009b]. We denote a candidate character token c_i with a context $...c_{i-1}c_ic_{i+1}...$. The features are divided into two types: character features and word type features. The character-based features are indicator functions that fire when the bracketing label takes some value and some predict of the input (at a certain position) corresponding to the label is satisfied. Note that the word type features are indicator functions that fire when the local character sequence matches a word unigram or bigram. Dictionaries containing word unigrams and bigrams was collected from the training data.

We use the predicate templates as follows:

- *character unigram*: c_s $(i-3 < s < i+3)$
- *character bigram*: $c_s c_{s+1}$ $(i-3 < s < i+3)$
- Whether c_s and c_{s+1} are *identical*, for $i-2 < s < i+2$.
- Whether c_s and c_{s+2} are *identical*, for $i-4 < s < i+2$.

The latter two feature templates are designed to detect character or word reduplication, a morphological phenomenon that can influence word segmentation in Chinese.

The word type features are indicator functions that fire when the local character sequence matches a word unigram or bigram. The dictionary containing word and bigram information was collected from the training data. They includes,

- The *identity* of the string $c_{[s:i]}$ $(i-6 < s < i)$, if it matches a word from the list of unigram words;
- the *identity* of the string $c_{[i:s]}$ $(i < s < i+6)$, if it matches a word; multiple features could be generated.
- The *identity* of the bigram $c_{[s:i-1]}c_{[i:t]}$ $(i-6 < s, t < i+6)$, if it matches a word bigram from the list of unigram words.
- The *identity* of the bigram $c_{[j:i]}c_{[i+1:k]}$ $(i-6 < s, t < i+6)$, if it matches a word bigram; multiple features could be generated.

2.4.2 Setting

The data sets from the international Chinese word segmentation bakeoffs are popular of Chinese word segmentation research. In this chapter, we used the data provided by the second SIGHAN Bakeoff [Emerson, 2005] to test the two segmentation models. The data contains four corpora from different sources: Academia Sinica Corpus (AS), City University of Hong Kong (CU), Microsoft Research Asia (MSR), and Peking University (PKU). Experiments have shown that there is only about 75% agreement among native speakers regarding the correct word segmentation (Sproat et al., 1996). There is no fixed standard for Chinese word segmentation. Also, specific NLP tasks may require different segmentation criteria. For example, "赵紫阳/Person name: Zhao Zhiyang" should be treated as two words (surname and given name) in the PKU data, but one word in the MSR data. The four data sets above are annotated with different standards. To catch general properties, we do experiments on all the four data sets. For the generation of word token features used in the character-based model, we extracted a unigram and bigram word list from the training data as the dictionary. Three metrics were used for evaluation: precision (P), recall (R) and balanced F-score (F) defined by 2PR/(P+R). For more detailed information on the corpora and these metrics, refer to [Emerson, 2005].

2.4.3 Results

2.4.3.1 Baseline Performance

Table 2.1 shows the performance of our two segmenters. Numbers of iterations are respectively set to 15 and 20 for our word-based segmenter and character-based segmenter. The word-based segmenter performs slightly worse than the character-based

	Model	P(%)	R(%)	F
AS	Character	94.8	94.7	94.7
	Word	93.5	94.8	94.2
CU	Character	95.5	94.6	95.0
	Word	94.4	94.7	94.6
MSR	Character	96.1	96.5	96.3
	Word	96.0	96.3	96.1
PKU	Character	94.6	94.9	94.8
	Word	94.7	94.3	94.5

Table 2.1: Baseline performance.

segmenter. This is different from the experiments reported in [Zhang and Clark, 2007]. We think the main reason is that we use a different learning architecture.

2.4.3.2 Word Frequency Factors

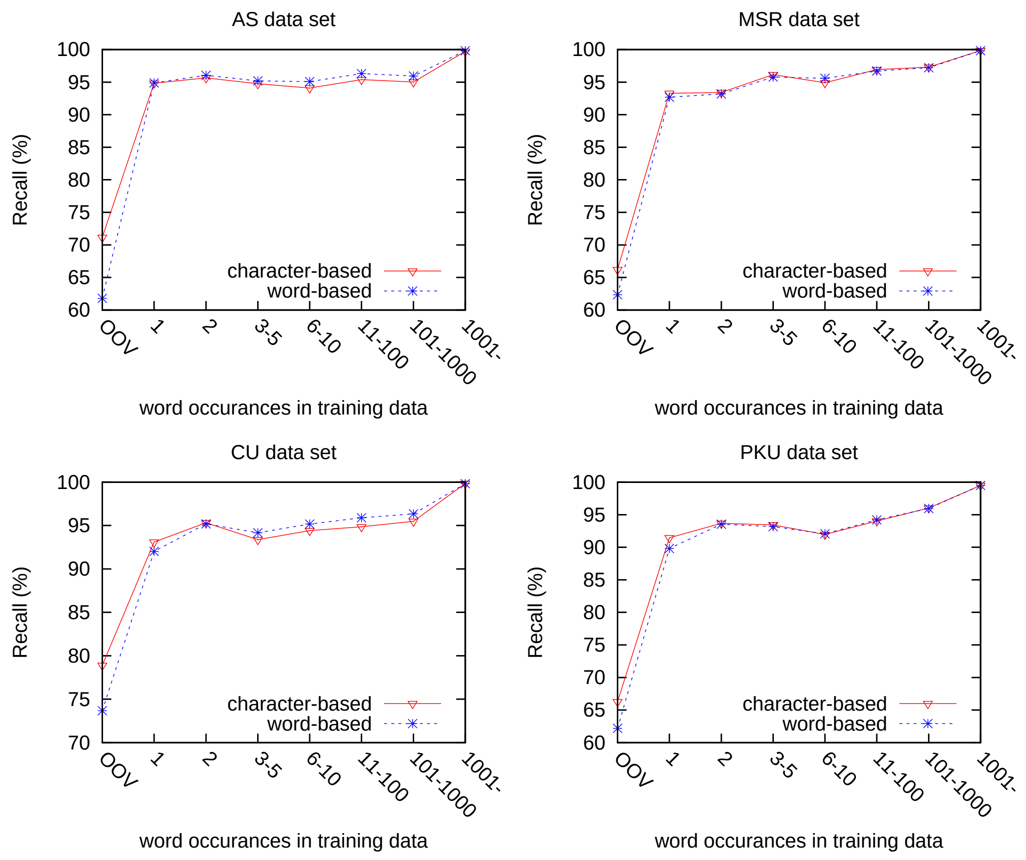


Figure 2.1: Segmentation recall relative to gold word frequency.

Our theoretical analysis also suggests that the character-based method has stronger word induction ability because it focuses more on word internal structures and thereby

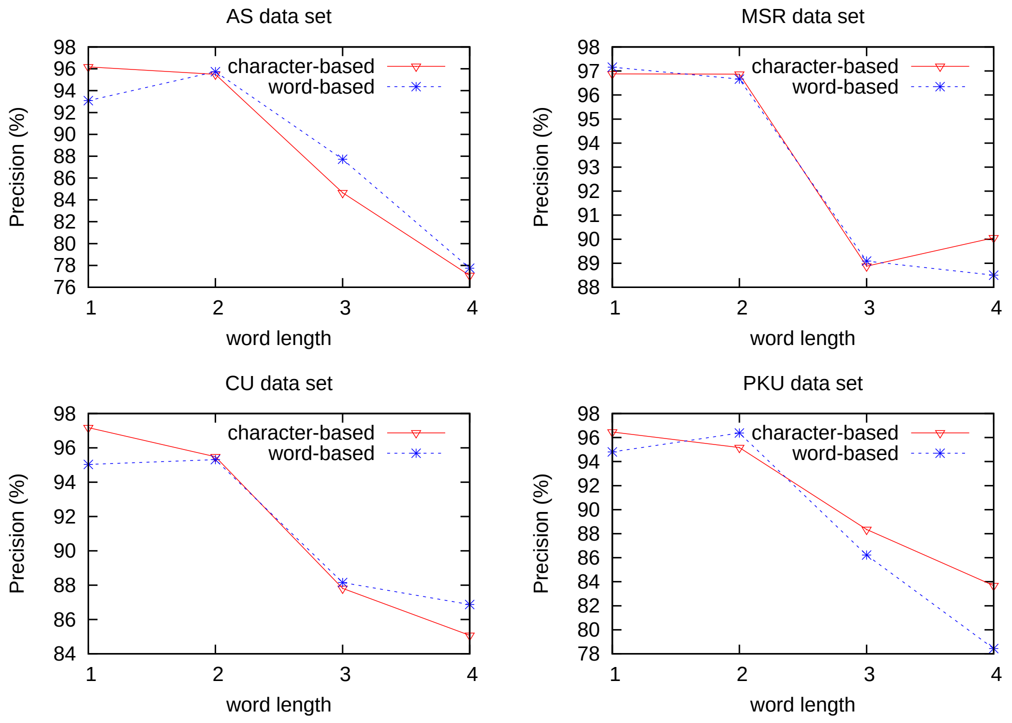


Figure 2.2: Segmentation precision relative to gold word length in training data.

expresses more nonlinearity. To test the word induction ability, we present the recall relative to word frequency. If a word appears in a training data many times, the learner usually works in a "memorizing" way. On the contrary, infrequent words should be correctly recognized in a somehow "inductive" way. Figure 2.1 shows the recall change relative to word frequency in each training data. Note that, the words with frequency 0 are out-of-vocabulary (OOV) words. We can clearly see that character-based model outperforms word-based model for infrequent word, especially OOV words, recognition. The "memorizing" ability of the two models is similar; on the AS and CU data sets, the word-based model performs slightly better. Neither model is robust enough to reliably segment unfamiliar words. The recall of OOV words is much lower than in-vocabulary words. This is still very far away from real-world application where any varieties of Chinese texts must be successfully segmented.

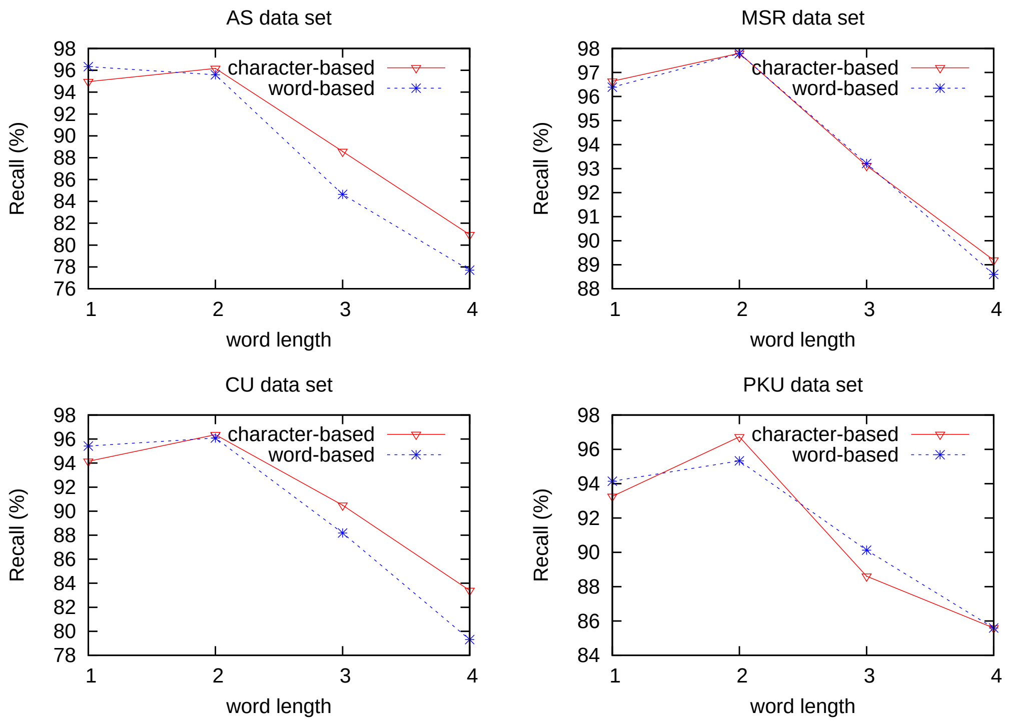


Figure 2.3: Segmentation recall relative to gold word length in training data.

2.4.3.3 Length Factors

Table 2.2 shows the statistics of word counts relative to word length on each test data sets. There are much less words with length more than 4. Analysis on long words may not be statistical significant, so we only present length factors on small words (length is less than 5). Figure 2.3 shows the precision/recall of both segmentation models relative sentence length. We can see that word-based model tends to predict more single character words, but making more mistakes. Since about 50% word tokens are single-character words, this is one main source of error for word-segmenter. In other words, word-based model segment sentences into more frages. This can be explained by theoretical properties of dynamic token prediction discussed in Sec. 2.3.3. The basic predicting unit is dynamic rather than static, and the score is the sum of all local scores of all basic units. The score of a word boundary assignment in a word-based segmenter is defined like $\theta^\top \sum_{i=1}^{|\mathbf{w}|} \phi(\mathbf{c}, w_{[1:i]})$. The upper bound of this score varies with the length $|\mathbf{w}|$. If a segmentation result is with more fragments, i.e. $|\mathbf{w}|$ is

Length	AS	CU	MSR	PKU
1	61254	19116	48092	45911
2	52268	18186	49472	49861
3	6990	2682	4652	5132
4	1417	759	2711	2059
5(+)	690	193	1946	656

Table 2.2: Word length statistics on test sets.

larger, the upper bound of its score is higher. As a result, in many cases, a word-based segmenter prefers shorter words, which may cause errors.

2.4.3.4 Feature Factors

We would like to measure the effect of features empirically. In particular, we do not use dynamic *word token features* in our word-based segmenter, and *word type features* in our character-based segmenter as comparison with "standard" segmenters. The difference in performance can be seen as the contribution of *word features.* There are obvious drops in both cases. Though it is not a fair comparison, word token features seem more important, since the numerical decrease in the word-based experiment is larger. We also show the different performence of a character-based segmenter with and without *lexicon features.* Note that *word features* are actually word *token* features, while *lexicon features* are word *type* features.

	word-based		character-based	
	−	+	−	+
AS	93.1	94.2	94.1	94.7
CU	92.6	94.6	94.2	95.0
MSR	95.7	96.1	95.8	96.3
PKU	93.3	94.5	94.4	94.8

Table 2.3: F-score of two segmenters, with (−) and without (+) *word token/type features.*

2.5 Combination

The above analysis indicates that the theoretical differences cause different error distributions. The error analysis further suggests that there is still space for improvement, just by combining the two existing models. Here, we introduce a classifier ensemble method for system combination.

	P(%)	R(%)	F	ER (%)
AS	96.6	96.9	96.7	37.7
CU	97.4	97.1	97.3	46.0
MSR	97.5	97.7	97.6	35.1
PKU	96.8	96.2	96.5	32.7

Table 2.4: Upper bound for combination. The error reduction (ER) rate is a comparison between the F-score produced by the oracle combination system and the character-based system (see Table 2.1).

2.5.1 Upper Bound of System Combination

To get an upper bound of the improvement that can be obtained by combining the strengths of each model, we have performed an oracle experiment. For every word in each gold standard segmented text, we check whether it is rightly predicted by either system. We think the optimal combination system should choose the right prediction when the two segmenters do not agree with each other. There is a *gold segmenter* that generates gold-standard segmentation results. In the oracle experiment, we let the three segmenters, i.e. baseline segmenters and the gold segmenter, vote. The three segmenters output three segmentation results, which are further transformed into IOB2 representation [Ramshaw and Marcus, 1995]. Namely, each character has three B or I labels. We assign each character an oracle label which is chosen by at least two segmenters. When the baseline segmenters agree with each other, the gold segmenter cannot change the segmentation no matter if it is right or wrong. In the situation that the two baseline segmenters disagree, the vote given by the gold segmenter will decide the right prediction. This kind of optimal performance is presented in Table 3.4. Compared these results with Table 2.1, we see a significant increase in accuracy for the four data sets. There is still much room for improvement. The upper bound of error reduction with system combination is over 30%.

2.5.2 Segmenter Ensemble via Bagging

Bootstrap aggregating (Bagging) is a machine learning ensemble meta-algorithm to improve classification and regression models in terms of stability and classification accuracy [Breiman, 1996a]. It also reduces variance and helps to avoid overfitting. Although it is usually applied to decision tree models, it can be used with any type of model. Bagging is a special case of the model averaging approach. Given a training set D of size n, Bagging generates m new training sets D_i of size $n' \leq n$, by sampling examples from D uniformly. By sampling with replacement it is likely that some

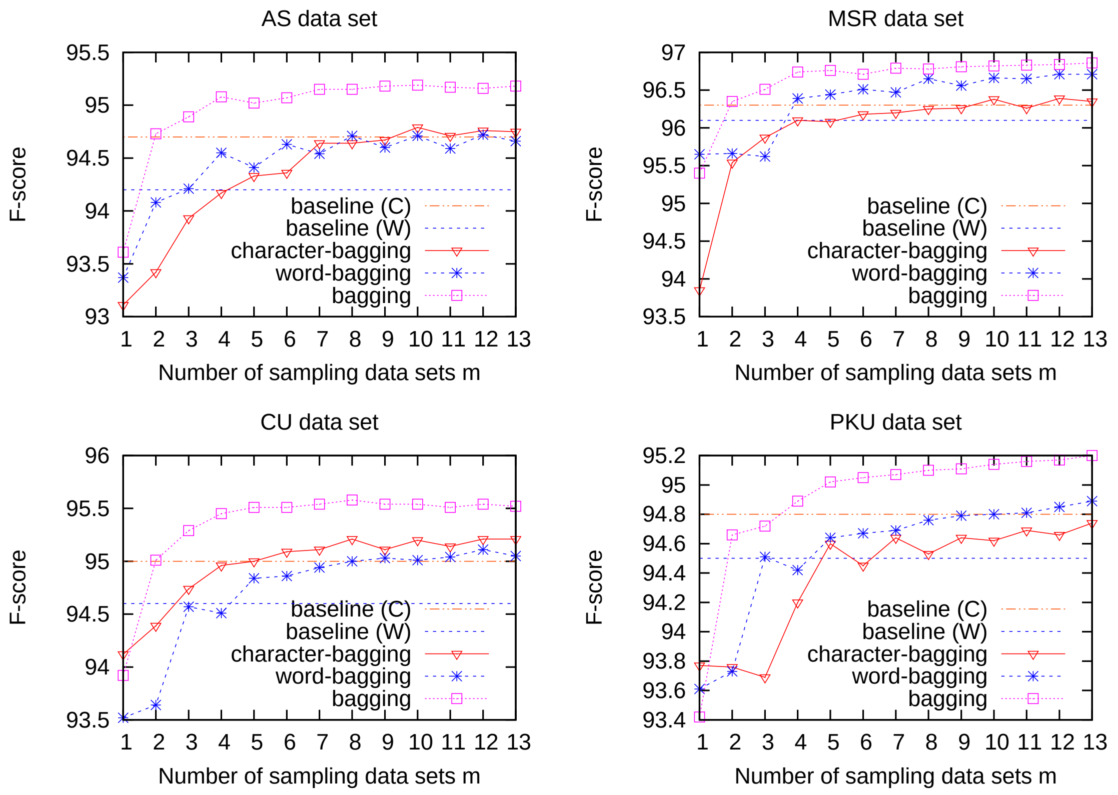


Figure 2.4: F-score of bagging models with different numbers of sampling data sets. *Character-bagging* means that the bagging system built on the single character-based segmenter. *Word-bagging* is named in the same way.

examples will be repeated in each D_i. If $n' = n$, then for large n the set D_i expected to have 63.2% of the unique examples of D, the rest being duplicates. This kind of sample is known as a bootstrap sample. The m models are fitted using the above m bootstrap samples and combined by voting (for classification) or averaging the output (for regression).

Note that Bagging is not useful for improving linear models, since the method averages several predictors. However, although the learning models of our segmenters are called linear models, they both involve nonlinearity (See Sec. 2.3.2). In addition, the two segmenters are in different architecture. The final prediction is not decided directly by the inner product of the parameters and features. This nonlinearity property makes the basic assumption of Bagging algorithm work.

We propose a Bagging model to combine multiple segmentation systems. In the training phase, given a training set D of size n, our model generates m new training sets D_i of size $63.2\% \times n$ by sampling examples from D without replacement. Namely no example will be repeated in each D_i. Each D_i is separately used to train a word-

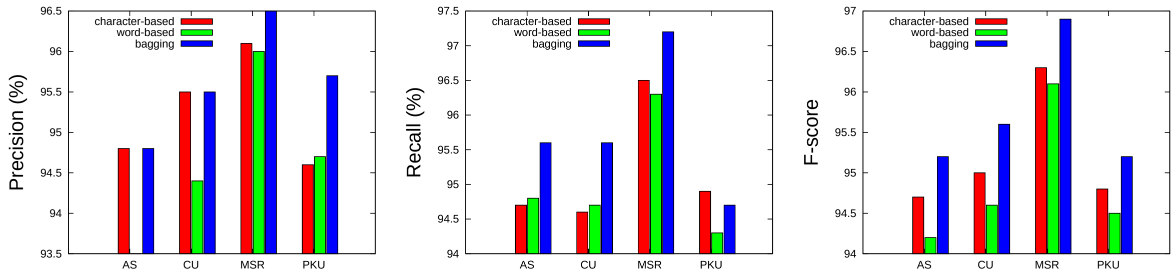


Figure 2.5: Precision/Recall/F-score of different models.

based segmenter and a character-based segmenter. Using this strategy, we can get $2m$ *weak* segmenters. Note that the sampling strategy is different from the standard one. Our experiment shows that there is no significant difference between the two sampling strategies in terms of accuracy. However, the *non-placement* strategy is more efficient. In the segmentation phase, the $2m$ models outputs $2m$ segmentation results, which are further transformed into IOB2 representation. In other words, each character has $2m$ B or I labels. The final segmentation is the voting result of these $2m$ labels. Note that since $2m$ is an even number, there may be equal number of B and I labels. In this case, our system prefer B to reduce error propagation.

2.5.3 Evaluation

We evaluate our combination model on the same datasets used above. Figure 2.4 shows the influence of m in the bagging algorithm. Because each new data set D_i in bagging algorithm is generated by a random procedure, the performance of all bagging experiments are not the same. To give a more stable evaluation, we repeat 5 experiments for each m and show the averaged F-score. We can see that the bagging model taking two segmentation models as basic systems consistently outperform the baseline systems and the bagging model taking either model in isolation as basic systems. An interesting phenomenon is that the bagging method can also improve word-based models. In contrast, there is no significant change in character-based models.

Figure 2.5 shows the precision, recall, F-score of the two baseline systems and our final system for which we generate $m = 15$ new data sets for bagging. We can see significant improvements on the four datasets in terms of the balanced F-score. The improvement of precision and recall are not consistent. The improvement of AS and CU datasets is from the recall improvement; the improvement of PKU datasets is from the precision improvement. We think the different performance is mainly because the four datasets are annotated by using different standards.

Table 2.5 summarizes the performance of our final system and other systems reported in a majority of previous work. The left most column indicates the reference of previous systems that represent state-of-the-art results. The comparison of the accuracy between our integrating system and the state-of-the-art segmentation systems in the literature indicates that our combination system is competitive with the best systems, obtaining the highest reported F-scores on three data sets.

	AS	CU	MSR	PKU
[Zhang et al., 2006]	95.1	95.1	97.1	95.1
[Zhang and Clark, 2007]	94.6	95.1	97.2	94.5
[Andrew, 2006]	N/A	N/A	97.2	N/A
[Sun et al., 2009b]	N/A	94.6	**97.3**	**95.2**
This paper	**95.2**	**95.6**	96.9	**95.2**

Table 2.5: Segmentation performance presented in previous work and of our combination model.

2.6 Conclusion and Discussion

Our theoretical and empirical analysis highlights the fundamental differences between word-based (semi-Markov tagging) and character-based (Markov tagging) models, which enlighten us to design new models. The above analysis indicates that the theoretical differences cause different error distributions. The two approaches are either based on a particular view of segmentation. The analysis is helpful to design new solutions for segmentation. Our analysis points out several drawbacks of each one. It may be helpful for both models to overcome their shortcomings. We may naturally ask what other methods may prove fruitful. For example, one weakness of word-based model is its word induction ability which is partially caused by its neglect of internal structure of words. A word-based model may be improved by solving this problem. On the other hand, character-based segmenters hope to find a way to utilize dynamic word token information. For example, Zhang et al. [2006] proposed a subword-based model, in which the basic predicting unit is larger than a character yet smaller than a word.

While the two mechanisms overlap in their numerical overall results, they are not redundant. Each segmentation model has strengths and weaknesses for certain design problems. We may construct a single system integrating the strengths of each segmenter. In this chapter, we try this direction by using an ensemble learning technique. The question "How to combine systems in different architectures" is currently a hot topic in a majority of NLP tasks. System combination strategies can be roughly divided into two categories: (1) learning-based post-inference and (2) learning-free post-inference. For example, in dependency parsing, several methods are proposed to integrate transition-based and graph-based parsers [Surdeanu and Manning, 2010]. Previous work pays much attention to incorporating features that use one system as main problem solver and the main solver use features generated from other systems [Nivre and McDonald, 2008; Torres Martins et al., 2008]. This kind of combination method involves learning in the training phase: A meta-learner is trained to provide combination decisions. The other kind of integration architecture is to directly combine outputs of different systems, such as voting. Note that this kind of combination method may involve complex inference procedure. For example, a re-parsing technique was successfully developed to combine the outputs provided by multiple parsers in [Sagae and Lavie, 2006b]. In their method, dependency structures given by different parsers are first used to create a weighted graph. Finding the optimal dependency structure is formulated as a maximum spanning tree (MST) inference problem over this graph.

The Bagging-based combination method proposed in this chapter is a learning-free inference method. In the next chapter, we will present a learning-based inference, i.e. sub-word tagging, to enhance word segmentation through combining three tagging methods.

Chapter 3

Stacked Sub-word Tagging for Joint Word Segmentation and POS Tagging

The large combined search space of joint word segmentation and Part-of-Speech (POS) tagging makes efficient decoding very hard. As a result, effective high order features representing rich contexts are inconvenient to use. In this chapter, we propose a novel stacked sub-word tagging model for this task, concerning both efficiency and effectiveness. Our solution is a two step process. First, multiple heterogeneous solvers are trained to produce coarse segmentation and POS information. Second, the outputs of the predictors are merged into sequences of largest non-overlapped strings, which are further bracketed and labeled with POS tags by a fine-grained sub-word tagger. The coarse-to-fine search scheme is efficient, while in the sub-word tagging step rich contextual features can be approximately derived. We also study the annotation ensemble problem and show that sub-word tagging a robust solution, in the sense that the coarse-grained solvers can be trained on heterogeneous annotations. Evaluation on the Penn Chinese Treebank and People's Daily data shows that our model yields significant improvements over the best system reported in the literature.

This chapter is originally published in [Sun, 2011] and [Sun and Wan, 2012].

3.1 Background

3.1.1 The Problem

Word segmentation and part-of-speech (POS) tagging are fundamental steps for more advanced Chinese language processing tasks, such as parsing and semantic role labeling. Joint approaches that resolve the two tasks simultaneously have received much attention in recent research. Previous work has shown that joint solutions led to accuracy improvements over pipelined systems by avoiding segmentation error propagation and exploiting POS information to help segmentation. A challenge for joint approaches is the large combined search space, which makes efficient decoding and structured learning of parameters very hard. Moreover, the representation ability of models is limited since using rich contextual word features makes the search intractable. To overcome such efficiency and effectiveness limitations, approximate inference and reranking techniques have been explored in previous work [Jiang et al., 2008b; Zhang and Clark, 2010].

Given a sequence of characters $\mathbf{c} = (c_1, ..., c_{\#\mathbf{c}})$, the task of word segmentation and POS tagging is to predict a sequence of word and POS tag pairs $\mathbf{y} = (\langle w_1, p_1 \rangle, \langle w_{\#\mathbf{y}}, p_{\#\mathbf{y}} \rangle)$, where w_i is a word, p_i is its POS tag, and a "#" symbol denotes the number of elements in each variable. In order to avoid error propagation and make use of POS information for word segmentation, the two tasks should be resolved jointly. Previous research has shown that the integrated methods outperformed pipelined systems [Jiang et al., 2008a; Ng and Low, 2004; Zhang and Clark, 2008a]. A major challenge for such joint systems is the large search space faced by the decoder. Decoding can be inefficient.

3.1.2 Character-Based and Word-Based Methods

Similar to word segmentation, both word-based (semi-Markov tagging) and character-based (Markov tagging) methods are popular for joint word segmentation and POS tagging. Word segmentation can be viewed as a bracketing problem, while joint segmentation and tagging can be viewed as a labeled bracketing problem.

In the "word-based" approach, the basic predicting units are words themselves. This kind of solver sequentially decides whether the local sequence of characters makes up a word as well as its possible POS tag. In particular, a word-based solver reads the input sentence from left to right, predicts whether the current piece of continuous characters is a word token and which class it belongs to. Solvers may use previously predicted words and their POS information as clues to find a new word. After one word is found and classified, solvers move on and search for the next possible word. This word-by-word method for segmentation was first proposed in [Zhang and Clark, 2007], and was then further used in POS tagging in [Zhang and Clark, 2008a].

In the "character-based" approach, the basic processing units are characters which compose words, and joint segmentation and tagging is formulated as the classification of characters into POS tags with boundary information. For example, the label *B-NN* indicates that a character is located at the begging of a noun. Using this method, POS information is allowed to interact with segmentation. This character-by-character method for segmentation was first proposed in [Xue, 2003], and was then further used in POS tagging in [Ng and Low, 2004]. One main disadvantage of this model is the difficulty in incorporating the whole word information. Note that the hybrid approach described in [Kruengkrai et al., 2009; Nakagawa and Uchimoto, 2007] is also a character-based approach, since the word information used is word *type* information.

3.1.3 Stacked Learning

Stacked generalization is a meta-learning algorithm that was first proposed in [Wolpert, 1992] and [Breiman, 1996b]. The idea is to include two "levels" of predictors. The first level includes one or more predictors $g_1, ...g_K : \mathbb{R}^d \rightarrow \mathbb{R}$; each receives input $\mathbf{x} \in \mathbb{R}^d$ and outputs a prediction $g_k(\mathbf{x})$. The second level consists of a single function $h : \mathbb{R}^{d+K} \rightarrow \mathbb{R}$ that takes as input $\langle \mathbf{x}, g_1(\mathbf{x}), ..., g_K(\mathbf{x}) \rangle$ and outputs a final prediction $\hat{y} = h(\mathbf{x}, g_1(\mathbf{x}), ..., g_K(\mathbf{x}))$.

Training is done as follows. The training data $S = \{(\mathbf{x}_t, \mathbf{y}_t) : t \in [1, T]\}$ is split into L equal-sized disjoint subsets $S_1, ..., S_L$. Then functions $\mathbf{g}_1, ..., \mathbf{g}_L$ (where $\mathbf{g}_l = \langle g_1^l, ..., g_K^l \rangle$) are separately trained on $S - S_l$, and are used to construct the augmented data set $\hat{S} = \{(\langle \mathbf{x}_t, \hat{\mathbf{y}}_t^1, ..., \hat{\mathbf{y}}_t^K \rangle, \mathbf{y}_t) : \hat{\mathbf{y}}_t^k = g_k^l(\mathbf{x}_t) \text{ and } \mathbf{x}_t \in S_l\}$. Finally, each g_k is trained on the original data set and the second level predictor h is trained on $\hat{S}$. The intent of the *cross-validation* scheme is that $\mathbf{y}_t^k$ is similar to the prediction produced by a predictor which is learned on a sample that does not include $\mathbf{x}_t$.

Stacked learning has been applied as a system ensemble method in several NLP tasks, such as named entity recognition [Wu et al., 2003] and dependency parsing [Nivre and McDonald, 2008]. This framework is also explored as a solution for learning *non-local* features in [Torres Martins et al., 2008]. In the machine learning research, stacked learning has been applied to structured prediction [Cohen and Carvalho, 2005]. In this work, stacked learning is used to acquire extended training data for sub-word tagging.

3.1.4 Annotation Ensemble

A majority of data-driven NLP systems relies on large-scale, manually annotated corpora. These corpora are important to train statistical systems but very expensive to build. Nowadays, for many NLP tasks, multiple heterogeneous annotated corpora have been built and are publicly available. For example, the Penn Treebank is popular to train PCFG-based parsers, while the Redwoods Treebank is well known for HPSG research; the Propbank is favored to build general semantic role labeling systems, while the FrameNet is attractive for predicate-specific labeling. However, the annotation schemes in different projects are usually different, since the underlying linguistic theories vary and have different ways to explain the same language phenomena.

The co-existence of heterogeneous annotation data, i.e. labeled data in different representations, presents a new challenge to the consumers of such resources. While many state-of-the-art statistical NLP systems are not bound to specific annotation standards, almost all of them assume homogeneous annotation in the training corpus. Therefore, such heterogeneous resources cannot be simply put together while training systems. In this chapter, we address the question about annotation ensemble—learning from instances that have multiple independent representations—which is a natural, yet non-standard new problem setting. There has been a feature-engineering solution for segmentation and POS tagging [Jiang et al., 2009]. Different from their work, we incorporate heterogeneous taggers into our sub-word tagging model, which more explicitly explores the relation between heterogenous annotations.

3.2 A Stacked Sub-word Tagging Model

3.2.1 Method

In this chapter, we propose a novel stacked sub-word model for joint word segmentation and POS tagging, concerning both efficiency and effectiveness. Our work is motivated by several characteristics of this problem. First of all, a majority of words are easy to identify in the segmentation problem. For example, a simple maximum matching segmenter can achieve an f-score of about 90. We will show that it is possible to improve the efficiency and accuracy by using different strategies for different words. However, previous approaches treat all possible words equally. The basic strategy in this work is to identify *simple* and *difficult* words first and to integrate them into a sub-word level. To identify *simple* words, we borrow ideas from system ensemble.

Second, segmenters designed with different views have complementary strength. We argue that the agreements and disagreements of different solvers can be used to construct an intermediate sub-word structure for joint segmentation and tagging. Since the sub-words are large enough in practice, the decoding for POS tagging over sub-words is efficient.

Finally, the Chinese language is characterized by the lack of morphology that often provides important clues for POS tagging, and the POS tags contain much syntactic information, which need context information within a large window for disambiguation. For example, Huang et al. [2007] showed the effectiveness of utilizing syntactic information to rerank POS tagging results. As a result, the capability to represent rich contextual features is crucial to a Chinese POS tagger. In this work, we use a representation-efficiency tradeoff through stacked learning, a way of approximating rich *non-local* features.

Given multiple word segmentations of one sentence, we formally define a sub-word structure that maximizes the agreement of non-word-break positions. Based on the sub-word structure, joint segmentation and tagging is addressed as a two step process: (1) coarse-grained word segmentation and tagging, and (2) fine-grained sub-word tagging. The workflow is shown in Figure 3.1. In the first phase, one word-based segmenter ($\mathrm{Seg_W}$) and one character-based segmenter ($\mathrm{Seg_C}$) are trained to produce word boundaries. Additionally, a *local* character-based joint segmentation and tagging solver ($\mathrm{SegTag_L}$) is used to provide word boundaries as well as inaccurate POS information. Here, the word *local* means the labels of nearby characters are not used as features. In other words, the local character classifier assumes that the tags of characters are independent of each other. In the second phase, our system first combines the three segmentation and tagging results to get sub-words which maximize the agreement about word boundaries. Finally, a fine-grained sub-word tagger (SubTag) is applied to bracket sub-words into words and also to obtain their POS tags.

In our model, segmentation and POS tagging interact with each other in two processes. First, although Seg_L is locally trained, it resolves the two sub-tasks simultaneously. Therefore, in the sub-word generating stage, segmentation and POS tagging help each other. Second, in the sub-word tagging stage, the bracketing and the classification of sub-words are jointly resolved as one sequence labeling problem.

Our experiments on the Penn Chinese Treebank will show that the word-based and

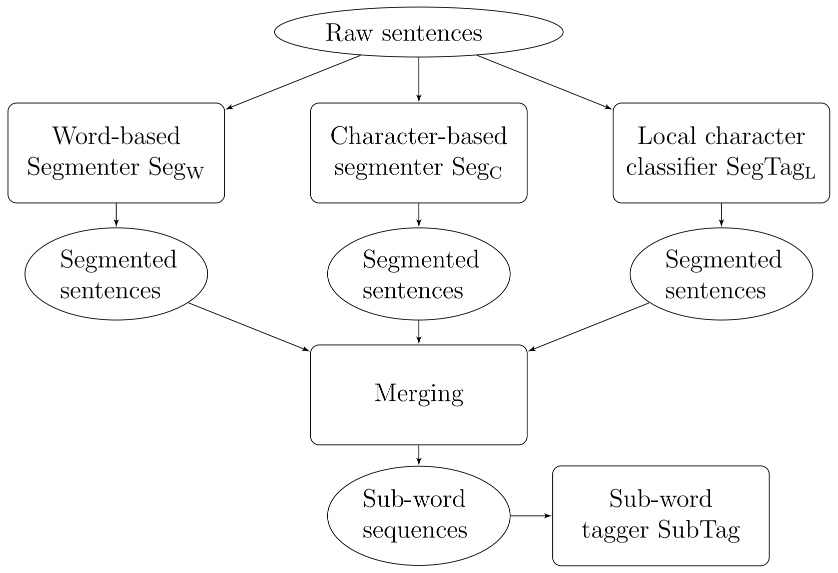

Raw sentences
Word-based Segmenter Seg_W
Character-based segmenter Seg_C
Local character classifier SegTag_L
Segmented sentences
Segmented sentences
Segmented sentences
Merging
Sub-word sequences
Sub-word tagger SubTag

Figure 3.1: Workflow of the stacked sub-word model.

character-based segmenters and the local tagger on their own produce high quality word boundaries. As a result, the oracle performance to recover words from a sub-word sequence is very high. The quality of the final tagger relies on the quality of the sub-word tagger. If a high performance sub-word tagger can be constructed, the whole task can be well resolved. The statistics will also empirically show that sub-words are significantly larger than characters and only slightly smaller than words. As a result, the search space of the sub-word tagging is significantly shrunken, and exact Viterbi decoding without approximately pruning can be efficiently processed. This property makes nearly all popular sequence labeling algorithms applicable.

Zhang et al. [2006] described a sub-word based tagging model to resolve word segmentation. To get the pieces which are larger than characters but smaller than words, they combine a character-based segmenter and a dictionary matching segmenter. Our contributions include (1) providing a formal definition of our sub-word structure that is based on multiple segmentations and (2) proposing a stacking method to acquire sub-words.

3.2.2 The Coarse-grained Solvers

In the former chapter, we systematically described the implementation of two state-of-the-art Chinese word segmenters in word-based and character-based architectures, respectively. In this chapter, we introduce two simple but important refinements: (1)

	以	总	成	绩	3	5	5	.	3	5	分	居	领	先	地	位
Answer:	[P]	[JJ]	[NN	]	[			CD		]	[M]	[VV]	[JJ	]	[NN	]
Seg_W:	[]	[]	[	]	[		]	[		]	[	]	[	]	[	]
Seg_C:	[]	[]	[	]	[						]	[]	[	]	[	]
SegTag_L:	[P]	[JJ]	[NN	]	[	CD		]	[NT]	[CD]	[NT]	[VV]	[VV	]	[NN	]
Sub-words:	[P]	[JJ]	[NN	]	[	B-CD	]	[I-CD]	[NT]	[CD]	[NT]	[VV]	[VV	]	[NN	]

Figure 3.2: An example phrase: 以总成绩３５５．３５分居领先地位 (Being in front with a total score of 355.35 points).

to shuffle the sample orders in each iteration and (2) to average the parameters in each iteration as the final parameters.

We use a local classifier to predict the POS tag with positional information for each character. Each character can be assigned one of two possible boundary tags: "B" for a character that begins a word and "I" for a character that occurs in the middle of a word. We denote a candidate character token c_i with a fixed window $c_{i-2}c_{i-1}c_ic_{i+1}c_{i+2}$. The following features are used:

- character unigrams: c_k $(i-2 \leq k \leq i+2)$
- character bigrams: c_kc_{k+1} $(i-2 \leq k \leq i+1)$

To resolve the classification problem, we use the linear SVM classifier LIBLINEAR[1]. Since the local classifier does not taken into account the labels of the nearby words, two consecutive labels may be inconsistent.

Idiom In linguistics, idioms are usually presumed to be figures of speech not fully obeying the principle of compositionality. As a result, it is very hard to recognize out-of-vocabulary idioms for word segmentation. However, the lexicon of idioms can be taken as a close set, which helps resolve the problem well. We collect 12992 idioms[2] from several online Chinese dictionaries. For both word-based and character-based segmentation, we first match every string of a given sentence with idioms. Every sentence is then split into smaller pieces which are separated by idioms. Statistical segmentation models are later performed on these smaller character sequences.

3.2.3 Generating Sub-word Sequences

A majority of words are easy to identify in the segmentation process. We favor the idea treating different words using different strategies. In this work we try to identify *simple* and *difficult* words first and to integrate them into a sub-word level. Inspired by previous work, we constructed this sub-word structure by using multiple solvers designed from different views. If a piece of continuous characters is consistently segmented by multiple segmenters, it will not be separated in the sub-word tagging step. The intuition is that strings which are consistently segmented by the different segmenters tend to be correct predictions. In our experiment on the Penn Chinese Treebank, the accuracy is 98.59% on the development data which is defined in the next section. The key point for the intermediate sub-word structures is to maximize the agreement of the three coarse-grained systems. In other words, the goal is to make merged sub-words as large as possible but not overlap with any predicted word produced by the three coarse-grained solvers. In particular, if the position between two continuous characters is predicted as a word boundary by any segmenter, this position is taken as a separation position of the sub-word sequence. This strategy makes sure that it is still possible to *re*-segment the strings of which the boundaries are disagreed with by the coarse-grained segmenters in the fine-grained tagging stage.

The formal definition is as follows. Given a sequence of characters $\mathbf{c} = (c_1, ..., c_{\#\mathbf{c}})$, let $c[i:j]$ denote a string that is made up of characters between c_i and c_j (including c_i and c_j), then a partition of the sentence can be written as $c[0:e_1], c[e_1+1:e_2], ..., c[e_m:\#\mathbf{c}]$. Let $s_k = \{c[i:j]\}$ denote the set of all segments of a partition. Given multiple partitions of a character sequence $\mathcal{S} = \{s_k\}$, there is one and only one merged partition $s_{\mathcal{S}} = \{c[i:j]\}$ s.t.

1. $\forall c[i:j] \in s_{\mathcal{S}}, \forall s_k \in \mathcal{S}, \exists c[s:e] \in s_k, s \leq i \leq j \leq e$.
2. $\forall \mathcal{C}'$ satisfies the above condition, $|\mathcal{C}'| > |\mathcal{C}|$.

The first condition makes sure that all segments in the merged partition can be only embedded in but do not overlap with any segment of any partition from $\mathcal{S}$. The second condition promises that segments of the merged partition achieve maximum length.

[1] LIBLINEAR is available at http://www.csie.ntu.edu.tw/~cjlin/liblinear/. We also tried some other popular classifiers and find that a crucial aspect in character classification is the representation of tokens with features, rather than the particular choice of classification algorithm.

[2] This resource is publicly available at http://www.coli.uni-saarland.de/~wsun/idioms.txt.

Figure 3.2 is an example to illustrate the procedure of our method. The lines Seg_{W}, Seg_{C} and SegTag_{L} are the predictions of the three coarse-grained solvers. The open and square bracket and the close square bracket respectively indicate the beginning and the end of a word. For the three words at the beginning and the two words at the end, the three predictors agree with each other. So these five words are kept as sub-words. For the character sequence "３５５．３５分居", the predictions are very different. Because there are no word break predictions among the first three characters "３５５", they together are taken as one sub-word. For the other five characters, either the left position or the right position is segmented as a word break by at least one predictor, so the merging processor separates them and takes each one as a single sub-word. The last line shows the merged sub-word sequence with their inaccurate POS tags. The coarse-grained POS tags with positional information are derived from the labels provided by SegTag_{L}.

3.2.4 Features

Bracketing sub-words into words is formulated as an IOB-style sequential classification problem. Each sub-word may be assigned with one POS tag as well as two possible boundary tags: "B" for the beginning position and "I" for the middle position. A tagger is trained to classify sub-word by using the features derived from its contexts.

The sub-word level allows our system to utilize features in a large context, which is very important for POS tagging of the morphologically poor language. Features are formed making use of sub-word contents, their IOB-style inaccurate POS tags. In the following description, "C" refers to the content of the sub-word, while "T" refers to the IOB-style POS tags. For convenience, we denote a sub-word with its context $...s_{i-2}s_{i-1}s_is_{i+1}s_{i+2}...$, where s_i is the current token. We denote l_C, l_T as the sizes of the window.

- Unigram features: $\text{C}(s_k)$ $(-l_C \le k \le l_C)$, $\text{T}(s_k)$ $(-l_T \le k \le l_T)$
- Bigram features: $\text{C}(s_k)\text{C}(s_{k+1})$ $(-l_C \le k \le l_C - 1)$, $\text{T}(s_k)\text{T}(s_{k+1})$ $(-l_T \le k \le l_T - 1)$
- $\text{C}(s_{i-1})\text{C}(s_{i+1})$ (if $l_C \ge 1$), $\text{T}(s_{i-1})\text{T}(s_{i+1})$ (if $l_T \ge 1$)
- $\text{T}(s_{i-2})\text{T}(s_{i+1})$ (if $l_T \ge 2$)
- In order to better handle unknown words, we also extract morphological features: character n-gram prefixes and suffixes for n up to 3.

C(s_{i-1})="成绩"; C(s_i)="３５５"; C(s_{i+1})="．"; T(s_{i-1})="NN";T(s_i)="B-CD"; T(s_{i+1})="I-CD";
C(s_{i-1})C(s_i)="成绩_３５５"; C(s_i)C(s_{i+1})="３５５_．"; T(s_{i-1})T(s_i)="NN_B-CD"; T(s_i)T(s_{i+1})="B-CD_I-CD";
C(s_{i-1})C(s_{i+1})="成绩_．"; T(s_{i-1})T(s_{i+1})="B-NN_I-CD";
Prefix(1)="３"; Prefix(2)="３５"; Prefix(3)="３５５" Suffix(1)="５"; Suffix(2)="５５"; Suffix(3)="３５５"

Table 3.1: An example of features used for sub-word tagging.

Take the sub-word "３５５" in Figure 3.2 for example, when l_C and l_T are both set to 1, all features used are listed in Table 3.1.

In the following experiments, we will vary window sizes l_C and l_T to find out the contribution of context information for the disambiguation. A first order Max-Margin Markov Networks model is used to resolve the sequence tagging problem. The SVM-HMM[1] implementation is chosen for the experiments in this work. We use the basic linear model without applying any kernel function.

3.2.5 Stacked Learning for Parameter Estimation

The three coarse-grained solvers Seg$_\mathrm{W}$, Seg$_\mathrm{C}$ and SegTag$_\mathrm{L}$ are directly trained on the original training data. When these three predictors are used to produce the training data, the performance is perfect. However, this does not hold when these models are applied to the test data. If we directly apply Seg$_\mathrm{W}$, Seg$_\mathrm{C}$ and SegTag$_\mathrm{L}$ to extend the training data to generate sub-word samples, the extended training data for the sub-word tagger will be very different from the data in the run time, resulting in poor performance.

One way to correct the training/test mismatch is to use the stacking method, where a K-fold *cross-validation* on the original data is performed to construct the training data for sub-word tagging. Algorithm 3 illustrates the learning procedure. First, the training data $S = \{(\mathbf{c}_t, \mathbf{y}_t)\}$ is split into L equal-sized disjoint subsets $S_1, ..., S_L$. For each subset S_l, the complementary set $S - S_l$ is used to train three coarse solvers Seg$_\mathrm{W}{}^l$, Seg$_\mathrm{C}{}^l$ and SegTag$_\mathrm{L}{}^l$, which process the S_l and provide inaccurate predictions. Then the inaccurate predictions are merged into sub-word sequences and S_l is extended to S'_l. Finally, the sub-word tagger is trained on the whole extended data set S'.

[1]Available at http://www.cs.cornell.edu/People/tj/svm_light/svm_hmm.html.

```
input : Data $S = \{(\mathbf{c}_t, \mathbf{y}_t), t = 1, 2, ..., n\}$
1 Split $S$ into L partitions $\{S_1, ...S_L\}$
2 for $l = 1, ..., L$ do
3     Train $\mathrm{Seg_W}^l$, $\mathrm{Seg_C}^l$ and $\mathrm{SegTag_L}^l$ using $S - S_l$.
4     Predict $S_l$ using $\mathrm{Seg_W}^l$, $\mathrm{Seg_C}^l$ and $\mathrm{SegTag_L}^l$.
5     Merge the predictions to get sub-words training sample $S'_l$.
6 end
7 Train the sub-word tagger SubTag using $S'$.
```
Algorithm 2: The stacked learning procedure for the sub-word tagger.

3.3 Experiments and Analysis

3.3.1 Setting

Previous studies on joint Chinese word segmentation and POS tagging have used the Penn Chinese Treebank (CTB) in experiments. We follow this setting in this paper. We use CTB 5.0 as our main corpus and define the training, development and test sets according to [Jiang et al., 2008a,b; Kruengkrai et al., 2009; Zhang and Clark, 2010]. Table 3.2 shows the statistics of our experimental settings.

Data set	CTB files	#sent.	#words
Training	1-270	18,089	493,939
	400-931		
	1001-1151		
Devel.	301-325	350	6821
Test	271-300	348	8008

Table 3.2: Training, development and test data on CTB 5.0

Three metrics are used for evaluation: precision (P), recall (R) and balanced f-score (F) defined by 2PR/(P+R). Precision is the relative amount of correct words in the system output. Recall is the relative amount of correct words compared to the gold standard annotations. For segmentation, a token is considered to be correct if its boundaries match the boundaries of a word in the gold standard. For the whole task, both the boundaries and the POS tag have to be correctly identical.

3.3.2 Performance of the Coarse-grained Solvers

Table 3.3 shows the performance on the development data set of the three coarse-grained solvers. In this paper, we use 20 iterations to train $\mathrm{Seg_W}$ and $\mathrm{Seg_C}$ for all experiments. Even only locally trained, the character classifier SegTag_L still significantly outperforms the two state-of-the-art segmenters Seg_W and Seg_C. This good performance indicates that the POS information is very important for word segmentation. Since we have odd number (3) segmentation system, we can directly combine these systems by voting. Following the voting method introduced in Chapter 2, we first transform the outputs the segmenters into IOB2 representation. In other words, each character has 3 B or I labels. The final segmentation is the voting result of these $2m$ labels. The last line show segmentation performance of the voting system. On this data set, we cannot see any improvement by voting.

Devel.	Task	P	R	F
Seg_W	Seg	94.55%	94.84%	94.69
Seg_C	Seg	95.10%	94.38%	94.73
SegTag_L	Seg	95.67%	95.98%	95.83
	Seg&Tag	87.54%	91.29%	89.38
Voting	Seg	96.06%	95.03%	95.54

Table 3.3: Performance of the coarse-grained solvers on the development data.

3.3.3 Statistics of Sub-words

Since the base predictors to generate coarse information are two word segmenters and a local character classifier, the coarse decoding is efficient. If the length of sub-words is too short, i.e. the decoding path for sub-word sequences are too long, the decoding of the fine-grained stage is still hard. The average length of sub-words on the development set is 1.64, while the average length of words is 1.69. The number of all IOB-style POS tags is 59 (when using 5-fold cross-validation to generate stacked training samples). The number of all POS tags is 35. Empirically, the decoding over sub-words is $\frac{1.69}{1.64} \times (\frac{59}{35})^{n+1}$ times as slow as the decoding over words, where n is the order of the Markov model. When a first order Markov model is used, this number is 2.93. These statistics empirically suggest that the decoding over sub-word sequence can be efficient.

On the other hand, the sub-word sequences are not perfect in the sense that they do not promise to recover all words because of the errors made in the first step. Similarly, we can only show the empirical upper bound of the sub-word tagging. The oracle performance of the final POS tagging on the development data set is shown in Table 3.4. The upper bound indicates that the coarse search procedure does not lose too much.

Task	P	R	F
Seg&Tag	99.50%	99.09%	99.29

Table 3.4: Upper bound of the sub-word tagging on the development data.

One main disadvantage of character-based approach is the difficulty to incorporate word features. Since the sub-words are on average close to words, sub-word features are good approximations of word features.

3.3.4 Rich Contextual Features Are Helpful

Table 3.5 shows the effect that features within different window size has on the sub-word tagging task. In this table, the symbol "C" means sub-word content features while the symbol "T" means IOB-style POS tag features. The number indicates the length of the window. For example, "C:±1" means that the tagger uses one preceding sub-word and one succeeding sub-word as features. From this table, we can clearly see the impact of features derived from neighboring sub-words. There is a significant increase between "C:±2" and "C:±1" models. This confirms our motivation that longer history and future features are crucial to the Chinese POS tagging problem. It is the main advantage of our model that making rich contextual features applicable. In all previous solutions, only features within a short history can be used due to the efficiency limitation.

The performance is further slightly improved when the window size is increased to 3. Using the labeled bracketing f-score, the evaluation shows that the "C:±3 T:±1" model performs the same as the "C:±3 T:±2" model. However, the sub-word classification accuracy of the "C:±3 T:±1" model is higher, so in the following experiments and the final results reported on the test data set, we choose this setting.

This table also suggests that the IOB-style POS information of sub-words does not contribute. We think there are two main reasons: (1) The POS information provided by the local classifier is inaccurate; (2) The structured learning of the sub-word tagger can use *real predicted* sub-word labels during its decoding time, since this learning algorithm does inference during the training time. It is still an open question whether more accurate POS information in rich contexts can help this task. If the answer is *YES*, how can we efficiently incorporate these features?

Devel.		P	R	F
C:0	T:0	92.52%	92.83%	92.67
C:±1	T:0	92.63%	93.27%	92.95
C:±1	T:±1	92.62%	93.05%	92.83
C:±2	T:±0	93.17%	93.86%	93.51
C:±2	T:±1	93.27%	93.64%	93.45
C:±2	T:±2	93.08%	93.61%	93.34
C:±3	T:±0	93.12%	93.86%	93.49
C:±3	T:±1	93.34%	93.96%	93.65
C:±3	T:±2	93.34%	93.96%	93.65

Table 3.5: Performance of the stacked sub-word model ($K = 5$) with features in different window sizes.

3.3.5 Stacked Learning Is Helpful

Table 3.6 compares the performance of "C:±3 T:±1" models trained with no stacking as well as different folds of cross-validation. We can see that although it is still possible to improve the segmentation and POS tagging performance compared to the local character classifier, the whole task just benefits only a little from the sub-word tagging procedure if the stacking technique is not applied. The stacking technique can significantly improve the system performance, both for segmentation and POS tagging. This experiment confirms the theoretical motivation of using stacked learning: simulating the test-time setting when a sub-word tagger is applied to a new instance. There is not much difference between the 5-fold and the 10-fold cross-validation.

Devel.	Task	P	R	F
No stacking	Seg	95.75%	96.48%	96.12
	Seg&Tag	91.42%	92.13%	91.77
$K = 5$	Seg	96.42%	97.04%	96.73
	Seg&Tag	93.34%	93.96%	93.65
$K = 10$	Seg	96.67%	97.11%	96.89
	Seg&Tag	93.50%	94.06%	93.78

Table 3.6: Performance on the development data. No stacking and different folds of cross-validation are separately applied.

3.3.6 Comparison to the State-of-the-Art

Table 3.7 summarizes the performance of our final system on the test data and other systems reported in a majority of previous work. The final results of our system are achieved by using 10-fold cross-validation "C:±3 T:±1" models. The left most column indicates the reference of previous systems that represent state-of-the-art results. The comparison of the accuracy between our stacked sub-word system and the state-of-the-art systems in the literature indicates that our method is slightly better than the best systems. Our system obtains the highest f-score performance on both segmentation and the whole task, resulting in error reductions of 14.1% and 5.5% respectively.

Test	Seg	Seg&Tag
[Jiang et al., 2008a]	97.85	93.41
[Jiang et al., 2008b]	97.74	93.37
[Kruengkrai et al., 2009]	97.87	93.67
[Zhang and Clark, 2010]	97.78	93.67
Our system	**98.17**	**94.02**

Table 3.7: F-score performance on the test data.

3.3.7 Results on the CTB 6.0

We conduct further experiments using the CTB 6.0, which is larger than the previous experimental data sets. The corpus was collected during different time periods from different sources with a diversity of topics. In order to obtain a representative split of data sets, we define the training, development and test sets according to the Chinese sub-task of the CoNLL 2009 shared task[1]. The core of the CoNLL 2009 shared task is to predict syntactic and semantic dependencies and their labeling. To evaluate Chinese dependency parsing, the organizers extract labeled dependencies from manually annotated treebanks. Here, we follow this division of the CTB since this setting considers many data annotation details, and provides more balanced data to train and evaluate Chinese language processing algorithms. Note that CoNLL 2009 does not utilize all annotated data available. Table 3.8 shows the statistics of this experimental setting.

Data set	#sent.	#words	#char.
Training	22277	609060	1004266
Devel.	1762	49620	83670
Test	2557	73152	121008

Table 3.8: Training, development and test data on CTB 6.0

The sequence labeling toolkit, SVM-HMM, used in previous experiments is a very

[1]http://ufal.mff.cuni.cz/conll2009-st/index.html

"expensive" algorithm. It is not suitable to be applied to large-scale data set due to the limit of memory. In the following experiments, we use a CRF learning toolkit, wapiti[1] [Lavergne et al., 2010]. Since more labeled data is available for training, extending window sizes of unigram and bigram features may further improve the tagging accuracy. Among several parameter estimation methods provided by wapiti, our auxiliary experiments indicate that the "rprop-" method works best. We use this algorithm and let other setting default to train a good sub-word tagger. We re-tune this parameter and find that window size 3 works best for the local classifier. We implement a "C:±3 T:±1" model with 10-fold cross-validation for sub-word tagging. The final results are reported in the Table 3.9.

Devel.	Task	P	R	F
Seg_W	Seg	95.11%	95.44%	95.27
Seg_C	Seg	95.53%	95.77%	95.65
SegTag_L	Seg	94.89%	95.27%	95.08
	Seg&Tag	86.15%	89.50%	87.79
Voting	Seg	95.92%	96.20%	96.06
SubTag	Seg	95.67%	96.18%	95.92
	Seg&Tag	90.81%	91.30%	91.06
Test	**Task**	**P**	**R**	**F**
Seg_W	Seg	94.68%	94.57%	94.63
Seg_C	Seg	95.09%	94.95%	95.02
SegTag_L	Seg	94.36%	94.42%	94.39
	Seg&Tag	85.78%	88.74%	87.24
Voting	Seg	95.52%	95.38%	95.45
SubTag	Seg	95.25%	95.42%	95.34
	Seg&Tag	90.29%	90.47%	90.38

Table 3.9: F-score performance on the CTB 6.0.

In the last part of the previous chapter (2.6), We discussed the *learning-free* and *learning-based* post-inference strategies for NLP system combination. Under our definition, our voting method to combine segmentation systems falls into the first category, while our stacked sub-word tagging method implements the second strategy. The fine-grained sub-word tagger can be viewed as a *meta*-learner to provide the final decisions of word boundaries based on the coarse-grained predictions.

Surdeanu and Manning [2010] present a systematic comparative study of different system combination methods to enhance English dependency parsing. In their experiments, the simplest voting scheme works quite well, even better than most of

[1]http://wapiti.limsi.fr/

more complex solutions. Although dependency parsing is very different from joint segmentation and tagging, their experiment at least show that meta-learning does not necessarily outperform voting. Our experiments on the different versions of the CTB also confirm this point. Meta-learning works better sometimes, while simple voting works better other times. There are many other ways to combine different models. For example, *log-linear* combination is very effective in statistical machine translation; dual decomposition can efficiently and effectively search the combinatorial optimization of different sub-models. The fundamental assumption that make sure system combination is possible to improve performances in terms of accuracy is the diversity. Our comparative analysis presented in the last chapter shows the diversity of word-based and character-based views. To some extent, this guarantees that both voting and meta-learning could enhance individual systems.

3.4 Reducing Approximation and Estimation Errors with Heterogeneous Annotations

For Chinese word segmentation and POS tagging, supervised learning has become a dominant paradigm. Much of the progress is due to the development of both corpora and machine learning techniques. Although several institutions to date have released their segmented and POS tagged data, acquiring sufficient quantities of high quality training examples is still a major bottleneck. The annotation schemes of existing lexical resources are different, since the underlying linguistic theories vary. Despite of the existence of multiple resources, such data cannot be simply put together for training systems, because almost all of statistical NLP systems assume homogeneous annotation. Therefore, it is not only interesting but also important to study how to fully utilize heterogeneous resources to improve Chinese lexical processing.

3.4.1 Two Essential Characteristics of Heterogeneous Annotations

There are two main types of errors in statistical NLP: (1) the approximation error that is due to the intrinsic suboptimality of a model and (2) the estimation error that is due to having only finite training data. Take Chinese word segmentation for example. Our previous analysis (Chapter 2) shows that one main intrinsic disadvantage of character-based model is the difficulty in incorporating the whole word information, while one main disadvantage of word-based model is the weak ability to express word formation. In both models, the significant decrease of the prediction accuracy of out-of-vocabulary (OOV) words indicates the impact of the estimation error.

There are two essential characteristics of heterogeneous annotations that can be utilized to reduce both approximation and estimation errors.

- On one hand, heterogeneous annotations are (similar but) *different* considering different annotation schemata. As a result, systems respectively trained on heterogeneous annotation data can produce different analysis. Auxiliary features from heterogeneous analysis can be derived for disambiguation, and therefore the approximation error can be reduced.

- On the other hand, heterogeneous annotations are (different but) *similar* in the sense that the corresponding linguistic analysis is highly correlated. An auxiliary corpus can be converted with a high precision for model re-training, and therefore the estimation error can be reduced. Note that different annotated corpora are usually based on different texts.

3.4.2 Diversity Analysis

In this chapter, we focus on two representative popular corpora for Chinese lexical processing: (1) the Penn Chinese Treebank (CTB) and (2) the PKU's People's Daily data (PPD). To analyze the diversity between their annotation standards, we pick up 200 sentences from CTB and manually label them according to the PPD standard. Especially, we employ a PPD-style segmentation and tagging system to automatically label these 200 sentences. As a linguistic expert who deeply understands the PPD annotation standard, the author manually checks the automatic analysis and correct its errors.

These 200 sentences are segmented as 3886 and 3882 words respectively according to the CTB and PPD standards. The average lengths of word tokens are almost the same. However, the word boundaries or the definitions of words are different. 3561 word tokens are consistently segmented by both standards. In other words, 91.7% CTB word tokens share the same word boundaries with 91.6% PPD word tokens. Among these 3561 words, there are 552 punctuations that are simply consistently segmented. If punctuations are filtered out to avoid overestimation of consistency, 90.4% CTB words have same boundaries with 90.3% PPD words. The boundaries of words that are differently segmented are *compatible*. Among all annotations, just one cross-bracketing occurs. The statistics indicates that the two heterogenous segmented

AD	⇒	d:149; c:11; ad:6; z:4; a:3; v:2; n:1; r:1; m:1; f:1; t:1;
AS	⇒	u:44;
BA	⇒	p:2; d:1;
CC	⇒	c:73; p:5; v:2;
CD	⇒	m:134;
CS	⇒	c:3; d:1;
DEC	⇒	u:83;
DEG	⇒	u:123;
DEV	⇒	u:7;
DT	⇒	r:15; b:1;
ETC	⇒	u:9;
JJ	⇒	a:43; b:13; n:3; vn:3; d:2; j:2; f:2; t:2; z:1;
LB	⇒	p:1;
LC	⇒	f:51; Ng:3; v:1; u:1;
M	⇒	q:101; n:11; v:1;
MSP	⇒	c:2; u:1;
NN	⇒	n:738; vn:135; v:26; j:19; Ng:5; an:5; a:3; r:3; s:3; Ag:2; nt:2; f:2; q:2; i:1; t:1; nz:1; b:1;
NR	⇒	ns:170; nr:65; j:23; nt:21; nz:7; n:2; s:1;
NT	⇒	t:98;
OD	⇒	m:41;
P	⇒	p:133; v:4; c:2; Vg:1;
PN	⇒	r:53; n:2;
PU	⇒	w:552;
SP	⇒	u:1;
VA	⇒	a:57; i:4; z:2; ad:1; b:1;
VC	⇒	v:32;
VE	⇒	v:13;
VV	⇒	v:382; i:5; a:3; Vg:2; vn:2; n:2; p:2; w:1;

Table 3.10: Mapping between CTB-style tags and PPD-style tags.

corpora are systematically different, and confirms the aforementioned two properties of heterogeneous annotations.

Table 3.10 is the mapping between CTB-style tags and PPD-style tags. For the definition and illustration of these tags, please refers to the annotation guidelines[1]. The statistics after colons are how many times this POS tag pair appears among the 3561 words that are consistently segmented. From this table, we can see that (1) there is no one-to-one mapping between their heterogeneous word classification but (2) the mapping between heterogeneous tags is not very uncertain. This simple

[1]Available at http://www.cis.upenn.edu/~chinese/posguide.3rd.ch.pdf and http://www.icl.pku.edu.cn/icl_groups/corpus/spec.htm.

analysis indicates that the two POS tagged corpora also hold the two properties of heterogeneous annotations. The differences between the POS annotation standards are systematic. The annotations in CTB are treebank-driven, and thus consider more functional (dynamic) information of basic lexical categories. The annotations in PPD are lexicon-driven, and thus focus on more *static* properties of words. Limited to the document length, we only illustrate the annotation of verbs and nouns for better understanding of the differences.

- The CTB tag **VV** indicates common verbs that are mainly labeled as verbs (v) too according to the PPD standard. However, these words can be also tagged as nominal categories (a, vn, n). The main reason is that there are a large number of Chinese adjectives and nouns that can be realized as predicates without linking verbs.

- The tag **NN** indicates common nouns in CTB. Some of them are labeled as verbal categories (vn, v). The main reason is that a majority of Chinese *verbs* could be realized as subjects and objects without form changes.

3.4.3 Reducing the Approximation Error via Stacking

3.4.3.1 Annotation Ensemble as System Integration

Each annotation data set alone can yield a predictor that can be taken as a mechanism to produce structured texts. With different training data, we can construct multiple heterogeneous systems, each of which independently associates its own structure, i.e. a word sequence, with the surface string. These systems produce similar linguistic analysis that holds the same high level linguistic principles but differs in details. To facilitate the description, we name one of these analysis as *target*, and others as *complementary* analysis. A very natural idea to take advantage of heterogeneous structures is to design a model which can predict a more accurate *target* structure based on the input, the less accurate *target* structure and relevant *complementary* structures.

This idea is very close to stacked learning that is a well developed technique for model ensemble. Formally speaking, this idea is to include two "levels" of processing. The first level includes one or more base predictors $f_1, ..., f_K$ that are independently built on heterogeneous training data. The second level processing consists of an inference function h that takes as input $\langle \mathbf{x}, f_1(\mathbf{x}), ..., f_K(\mathbf{x})\rangle$ and outputs a final prediction $h(\mathbf{x}, f_1(\mathbf{x}), ..., f_K(\mathbf{x}))$. The only difference between model ensemble and *annotation* *ensemble* is that the output spaces of model ensemble are the same while the output spaces of annotation ensemble are different. This framework is general and fexible, in the sense that it assumes almost nothing about the individual systems and takes them as black boxes.

In the following we will introduce a novel sub-word tagging model which is built on coarse-grained taggers that are trained with heterogeneous labeled data. To compare with Jiang et al. [2009]'s previous work, we also implement a similar feature-based stacking model. Since not only the features but also the word boundary structures are utilized for prediction refinement, we call our sub-word tagger a structure-based stacking model.

3.4.3.2 A Character-based Joint Model

With IOB2 representation [Ramshaw and Marcus, 1995], the problem of joint segmentation and tagging can be regarded as a character classification task. Previous work shows that the character-based approach is an effective method for Chinese lexical processing. Both of our feature- and structure-based stacking models employ base character-based taggers to generate multiple segmentation and tagging results. Our base tagger use a discriminative sequential classifier to predict the POS tag with positional information for each character. Each character can be assigned one of two possible boundary tags: "B" for a character that begins a word and "I" for a character that occurs in the middle of a word. We denote a candidate character token c_i with a fixed window $c_{i-2}c_{i-1}c_ic_{i+1}c_{i+2}$. The following features are used for classification:

- Character unigrams: c_k $(i - l \leq k \leq i + l)$
- Character bigrams: c_kc_{k+1} $(i - l \leq k < i + l)$

3.4.3.3 Feature-based Stacking

Jiang et al. [2009] introduced a feature-based stacking solution for annotation ensemble. In their solution, an auxiliary tagger CTag_{ppd} is trained on a complementary corpus, i.e. PPD, to assist the target CTB-style tagging. To refine the character-based tagger CTag_{ctb}, PPD-style character labels are directly incorporated as new features. The stacking model relies on the ability of discriminative learning method to explore informative features, which play central role to boost the tagging performance. To compare their feature-based stacking model and our structure-based model, we implement a similar system $\text{CTag}_{\text{ppd}\rightarrow\text{ctb}}$. Apart from character uni/bigram features, the PPD-style character labels are used to derive the following features to enhance our CTB-style tagger:

- Character label unigrams: c_k^{ppd} $(i - l^{\text{ppd}} \leq k \leq i + l^{\text{ppd}})$
- Character label bigrams: $c_k^{\text{ppd}} c_{k+1}^{\text{ppd}}$ $(i - l^{\text{ppd}} \leq k < i + l^{\text{ppd}})$

In the above descriptions, l and l^{ppd} are the window sizes of features, which can be tuned on development data.

3.4.3.4 Structure-based Stacking

The feature-based stacking is insufficient to fully utilize all information provided by a heterogeneous system. In this paper, we study structured-based stacking for joint word segmentation and POS tagging. In our solution, heterogeneous word structures are used not only to generate features but also to derive a sub-word structure which can better resolve the whole task. The design of the previous stacked sub-word tagging model is motivated by the diversity of heterogeneous models, while our current concern is to explore the diversity of heterogeneous annotations.

The workflow of our structure-based stacking system is shown in Figure 3.3. In the first phase, one character-based CTB-style tagger (CTag_{ctb}) and one character-based PPD-style tagger (CTag_{ppd}) are respectively trained to produce heterogenous word boundaries. In the second phase, this system first combines the two segmentation and tagging results to get sub-words which maximize the agreement about word boundaries. Finally, a fine-grained sub-word tagger (STag_{ctb}) is applied to bracket sub-words into words and also to label their POS tags. Note that we choose character-based taggers as coarse-grained processors just for simplicity.

To train the sub-word tagger STag_{ctb}, features are formed making use of both CTB-style and PPD-style POS tags provided by the character-based taggers. In the following description, "C" refers to the content of a sub-word; "T_{ctb}" and "T_{ppd}" refers to the positional POS tags generated from CTag_{ctb} and CTag_{ppd}; l_C, l_T^{ctb} and l_T^{ppd} are the window sizes. For convenience, we denote a sub-word with its context $...s_{i-1}s_is_{i+1}...$, where s_i is the current token. The following features are applied:

- Unigram features: $\text{C}(s_k)$ $(i - l_C \leq k \leq +l_C)$, $\text{T}_{\text{ctb}}(s_k)$ $(i - l_T^{\text{ctb}} \leq k \leq i + l_T^{\text{ctb}})$, $\text{T}_{\text{ppd}}(s_k)$ $(i - l_T^{\text{ppd}} \leq k \leq i + l_T^{\text{ppd}})$
- Bigram features: $\text{C}(s_k)\text{C}(s_{k+1})$ $(i - l_C \leq k < i + l_C)$, $\text{T}_{\text{ctb}}(s_k)\text{T}_{\text{ctb}}(s_{k+1})$ $(i - l_T^{\text{ctb}} \leq k < i + l_T^{\text{ctb}})$, $\text{T}_{\text{ppd}}(s_k)\text{T}_{\text{ppd}}(s_{k+1})$ $(i - l_T^{\text{ppd}} \leq k < i + l_T^{\text{ppd}})$

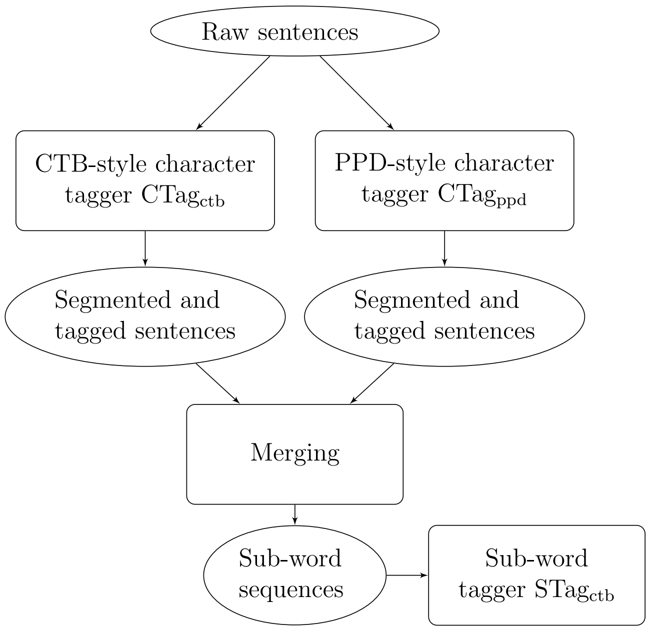

Raw sentences
CTB-style character tagger CTag$_{\text{ctb}}$
PPD-style character tagger CTag$_{\text{ppd}}$
Segmented and tagged sentences
Segmented and tagged sentences
Merging
Sub-word sequences
Sub-word tagger STag$_{\text{ctb}}$

Figure 3.3: Sub-word tagging based on heterogeneous taggers.

- $\text{C}(s_{i-1})\text{C}(s_{i+1})$ (if $l_C \geq 1$), $\text{T}_{\text{ctb}}(s_{i-1})\text{T}_{\text{ctb}}(s_{i+1})$ (if $l_T^{\text{ctb}} \geq 1$), $\text{T}_{\text{ppd}}(s_{i-1})\text{T}_{\text{ppd}}(s_{i+1})$ (if $l_T^{\text{ppd}} \geq 1$)
- Word formation features: character n-gram prefixes and suffixes for n up to 3.

Cross-validation CTag$_{\text{ctb}}$ and CTag$_{\text{ppd}}$ are directly trained on the original training data, i.e. the CTB and PPD data. Cross-validation technique has been proved necessary to generate the training data for sub-word tagging, since it deals with the training/test mismatch problem. To construct training data for the new heterogeneous sub-word tagger, a 10-fold *cross-validation* on the original CTB data is performed too.

3.4.4 Reducing the Estimation Error via Corpus Conversion

It is possible to acquire high quality labeled data for a specific annotation standard by exploring existing heterogeneous corpora, since the annotations are normally highly compatible. Moreover, the exploitation of additional (pseudo) labeled data aims to reduce the estimation error and enhances a NLP system in a different way from stacking. We therefore expect the improvements are not much overlapping and the combination of them can give a further improvement.

The stacking models can be viewed as annotation converters: They take as input complementary structures and produce as output target structures. In other words, the stacking models actually learn statistical models to transform the lexical representations. We can acquire informative extra samples by processing the PPD data with our stacking models. Though the converted annotations are imperfect, they are still helpful to reduce the estimation error.

Character-based Conversion The feature-based stacking model $\text{CTag}_{\text{ppd}\rightarrow\text{ctb}}$ maps the input character sequence $\mathbf{c}$ and its PPD-style character label sequence to the corresponding CTB-style character label sequence. This model by itself can be taken as a corpus conversion model to transform a PPD-style analysis to a CTB-style analysis. By processing the auxiliary corpus D_{ppd} with $\text{CTag}_{\text{ppd}\rightarrow\text{ctb}}$, we acquire a new labeled data set $D'_{ctb} = D_{ppd\rightarrow ctb}^{CTag_{ppd\rightarrow ctb}}$. We can re-train the $CTag_{ctb}$ model with both original and converted data $D_{ctb} \cup D'_{ctb}$.

Sub-word-based Conversion Similarly, the structure-based stacking model can be also taken as a corpus conversion model. By processing the auxiliary corpus D_{ppd} with STag_{ctb}, we acquire a new labeled data set $D''_{ctb} = D_{ppd\rightarrow ctb}^{STag_{ctb}}$. We can re-train the STag_{ctb} model with $D_{ctb} \cup D''_{ctb}$. If we use the gold PPD-style labels of D''_{ctb} to extract sub-words, the new model will overfit to the gold PPD-style labels, which are unavailable at test time. To avoid this training/test mismatch problem, we also employ a 10-fold cross validation procedure to add noise.

It is not a new topic to convert corpus from one formalism to another. A well known work is transforming Penn Treebank into resources for various deep linguistic processing, including `LTAG` [Xia, 1999], `CCG` [Hockenmaier and Steedman, 2007], `HPSG` [Miyao et al., 2004] and `LFG` [Cahill et al., 2002]. Such work for corpus conversion mainly leverages rich sets of hand-crafted rules to convert corpora. The construction of linguistic rules is usually time-consuming and the rules are not full coverage. Compared to rule-based conversion, our statistical converters are much easier to built and empirically perform well.

3.5 Evaluation of Annotation Ensemble

3.5.1 Setting

Our adaptation experiments are conducted on the PKU's People's Daily data[1] and the CTB 5.0 data. These two corpora are segmented and tagged following different standards. The CTB data is used for target tagging, while The annotation of the People's Daily of January in 1998 is used as a heterogeneous resource. This setup for annotation ensemble follows Jiang et al. [2009]'s experiments to lead to a fair comparison. The training, development and test data sets are defined as the same as in Section 3.3.1. To learn sequential classifiers, we the CRF toolkit wapiti.

3.5.2 Results of Stacking

Table 3.11 summarizes the segmentation and tagging performance of the baseline and different stacking models. The baseline of the character-based joint solver is competitive, and achieves an f-score of 92.93. By using the character labels from a heterogeneous solver (which is trained on the PPD data set), the performance of this character-based system is improved to 93.46. This result confirms the importance of a heterogeneous structure. Our structure-based stacking solution is effective and outperforms the feature-based stacking. By better exploiting the heterogeneous word boundary structures, our sub-word tagging model achieves an f-score of 94.03 (l_T^{ctb} and l_T^{pku} are tuned on the development data and both set to 1).

Devel.	P	R	F
CTag_{ctb}	93.28%	92.58%	92.93
$\text{CTag}_{\text{ppd}\rightarrow\text{ctb}}$	93.89%	93.46%	93.67
STag_{ctb}	94.07%	93.99%	94.03

Table 3.11: Performance of different stacking models on the development data.

The contribution of the auxiliary tagger is two-fold. On one hand, the heterogeneous solver provides structural information, which is the basis to construct the sub-word sequence. On the other hand, this tagger provides additional POS information, which is helpful for disambiguation. To evaluate these two contributions, we do another experiment by just using the heterogeneous word boundary structures without the POS information. The f-score of this type of sub-word tagging is 93.73. This result indicates that both the word boundary and POS information are helpful.

[1] This corpus is publicly available at `http://icl.pku.edu.cn/icl_res/`.

3.5.3 Learning Curves

We do additional experiments to evaluate the effect of heterogeneous features as the amount of PPD data is varied. Table 3.12 summarizes the f-score change. The feature-based model works well only when a considerable amount of heterogeneous data is available. When a small set is added, the performance is even lower than the baseline (92.93). The structure-based stacking model is more robust and obtains consistent gains regardless of the size of the complementary data.

		PPD → CTB	
#CTB	#PPD	CTag	STag
18104	7381	92.21	93.26
18104	14545	93.22	93.82
18104	21745	93.58	93.96
18104	28767	93.55	93.87
18104	35996	93.67	94.03
9052	9052	92.10	92.40

Table 3.12: F-scores relative to sizes of training data. Sizes (shown in column #CTB and #PPD) are numbers of sentences in each training corpus.

3.5.4 Results of Annotation Conversion

The stacking models can be viewed as data-driven annotation converting models. However they are not trained on "real" labeled samples. Although the target representation (CTB-style analysis in our case) is gold standard, the input representation (PPD-style analysis in our case) is labeled by a automatic tagger CTag_{ppd}. To make clear whether these stacking models trained with noisy inputs can *tolerant* perfect inputs, we evaluate the two stacking models on our manually converted data. The accuracies presented in Table 3.13 indicate that though the conversion models are learned by applying noisy data, they can refine target tagging with gold auxiliary tagging. Another interesting thing is that the gold PPD-style analysis does not help the sub-word tagging model as much as the character tagging model.

	Auto PPD	Gold PPD
$\text{CTag}_{\text{ppd}\rightarrow\text{ctb}}$	93.69	95.19
STag_{ctb}	94.14	94.70

Table 3.13: F-scores with gold PPD-style tagging on the manually converted data.

3.5.5 Results of Re-training

Table 3.14 shows accuracies of re-trained models. Note that a sub-word tagger is built on character taggers, so when we re-train a sub-word system, we should consider whether or not re-training base character taggers. The error rates decrease as automatically converted data is added to the training pool, especially for the character-based tagger CTag$_{\text{ctb}}$. When the base CTB-style tagging is improved, the final tagging is improved in the end. The re-training does not help the sub-word tagging much; the improvement is very modest.

$CTag_{ctb}$	$STag_{ctb}$	P	R	F
$D_{ctb} \cup D'_{ctb}$	- -	94.46%	94.06%	94.26
$D_{ctb} \cup D'_{ctb}$	D_{ctb}	94.61%	94.43%	94.52
D_{ctb}	$D_{ctb} \cup D''_{ctb}$	94.05%	94.08%	94.06
$D_{ctb} \cup D'_{ctb}$	$D_{ctb} \cup D''_{ctb}$	94.71%	94.53%	94.62

Table 3.14: Performance of re-trained models on the development data.

3.5.6 Comparison to the State-of-the-Art

Table 3.15 summarizes the tagging performance of different systems. The baseline of the character-based tagger is competitive, and achieve an f-score of 93.41. By better using the heterogeneous word boundary structures, our sub-word tagging model achieves an f-score of 94.36. Both character and sub-word tagging model can be enhanced with automatically converted corpus. With the pseudo labeled data, the performance goes up to 94.11 and 94.68. These results are also better than the best published result on the same data set that is reported in [Jiang et al., 2009].

Test	P	R	F
[Jiang et al., 2009]	- -	- -	94.02
[Wang et al., 2011]	- -	- -	94.18[1]
Character model	93.31%	93.51%	93.41
+Re-training	93.93%	94.29%	94.11
Sub-word model	94.10%	94.62%	94.36
+Re-training	**94.42%**	**94.93%**	**94.68**

Table 3.15: Performance of different systems on the test data.

[1]This result is achieved with much unlabeled data, which is different from our setting.

3.6 Conclusion

Inspired by the comparative analysis presented in last chapter, we design a novel stacked sub-word tagging model for joint word segmentation and POS tagging. We define a sub-word structure which maximizes the agreement of multiple segmentations provided by heterogeneous segmenters. We show that this sub-word structure could explore the complementary strengths of different systems designed with different views. Moreover, the POS tagging can be efficiently and effectively resolved over sub-word sequences. Exploiting diversity among different systems plays a central role in the success of our new model. By observing two essential characteristics of heterogeneous annotation data, we propose to use our new model to explore the diversity between different labeled corpora. A new sub-word tagging model together with corpus conversion is implemented and evaluated. Experiments show that our approach is superior to the existing approaches reported in the literature.

Chapter 4

Harvesting String Knowledge for Word Segmentation

This chapter investigates improving supervised word segmentation accuracy with unlabeled data. Both large-scale in-domain data and small-scale document text are considered. We present a unified solution to include features derived from unlabeled data to a discriminative learning model. For the large-scale data, we derive string statistics from Gigaword to assist a character-based segmenter. In addition, we introduce the idea about transductive, document-level segmentation, which is designed to improve the system recall for out-of-vocabulary (OOV) words which appear more than once inside a document. Novel features result in relative error reductions of 13.8% and 15.4% in terms of F-score and the recall of OOV words respectively. Our work can be viewed as a good example to leverage feature induction to bridge the gap between supervised language processing and unsupervised language acquisition.

This chapter is joint work with Jia Xu, originally published in [Sun and Xu, 2011].

4.1 Background

4.1.1 The Problem: Combining Supervised and Unsupervised NLP

Machine learning has become an indispensable tool for NLP researchers. Highly developed supervised training techniques have led to state-of-the-art performance for many NLP tasks. Unfortunately, given the limited availability of labeled data, and the non-trivial cost of human annotation, progress on supervised learning often yields diminishing returns. Unsupervised learning, on the other hand, is not bound by the same data resource limits. While labeled data is expensive to obtain, unlabeled data is essentially free in comparison. It exists simply as raw text from sources such as the Internet. The amount of unlabeled linguistic data available to us is much larger and growing much faster than the amount of labeled data. However, unsupervised learning is significantly harder than supervised learning and, although intriguing, has not been able to produce consistently successful results for most NLP tasks.

It is becoming increasingly important to leverage both types of data resources, labeled and unlabeled, to achieve the best performance in challenging NLP problems. Many semi-supervised learning methods, e.g. transductive SVM, graph-based methods, have been originally developed for binary classification problems. NLP problems often pose new challenges to these techniques, involving more complex structure that can violate many of the underlying assumptions. On the other hand, a number of *easy-to-implement* methods have been proposed, e.g. self-training and co-training, but their effectiveness on NLP tasks is not always clear. For example, bootstrapping methods typically assume a very small amount of labeled data and have not always shown to improve state-of-the-art performance when a large amount of labeled data is available, such as POS tagging [Clark et al., 2003].

We believe that it is important to explicitly investigate why and how auxiliary unlabeled data can truly improve NLP tasks. The following aspects motivate us to search for a robust semi-supervised solution that can help high-resource tasks.

- Flexibility: We favor the solutions which are easy to apply for problems with different structures (e.g. word sequences, syntactic trees or forests, N-best lists).
- Linguistic knowledge: We favor the idea exploiting NLP-specific background knowledge to aid semi-supervised learning.
- Scalability: NLP data-sets are often large, even for non-English tasks. We favor methods that can be applied to large-scale data (both labeled and unlabeled) sets.
- Effectiveness: We still expect gains even when high-performance supervised systems can be built. For example, we hope that semi-supervised learning can improve a supervised system that is already more than 95% accurate.

4.1.2 The Method: Feature Induction

In this chapter, we focus on a general framework for semi-supervised NLP, i.e. feature induction. Feature induction is a simple yet effective semi-supervised learning method for NLP. The basic strategy for taking advantage of unlabeled data is to derive informative features from large-scale unlabeled data and use them in discriminative supervised models. This “feature-engineering” approach has been successfully applied to named entity recognition (NER) [Lin and Wu, 2009; Miller et al., 2004], dependency parsing [Koo et al., 2008], query classification [Lin and Wu, 2009]. Miller et al. [2004] and Koo et al. [2008] demonstrated the effectiveness of using word clusters as features in discriminative learning. Following their ideas, Turian et al. [2010] compared different word clustering algorithms and evaluated their impacts on both NER and text chunking. Moreover, Lin and Wu [2009] present a simple and scalable algorithm for clustering tens of millions of phrases and use the resulting phrase clusters as features to enhance two applications: NER and query classification. Their experimental results show that phrase-based clusters offer significant improvements for NLP applications.

One of the advantages of the feature induction approach is that the learning algorithm is decoupled from the process of generating features. In other words, the construction of unlabeled data features is separated from training. This decoupling gives us the flexibility of using any algorithm to create different linguistic features that might be useful. Linguistic knowledge can explicitly motivate us to design good features based on unlabeled data. Moreover, models trained with features from unlabeled data are more compact and easier to interpret than more complex learning techniques, such as transductive SVMs. Feature induction increases the complexity of an original discriminative model only with new features, which are normally in a very small set. This property make this method efficient and scalable to most discriminative NLP systems. Finally, when good and task-related linguistic features are derived, they are reasonably expected to be useful clues for disambiguation.

4.2 Three Types of Unlabeled Data

We distinguish three types of unlabeled data, namely large-scale in-domain data, out-of-domain data and small-scale document text. Both large-scale in-domain and out-of-domain data are popular for enhancing NLP tasks. Learning from these two types of unlabeled data normally involves semi-supervised learning. The difference between them is that out-of-domain data is usually used for domain adaptation. For a number of NLP tasks, there are relatively large amounts of labeled training data. In this situation, supervised learning can provide competitive results, and it is difficult to improve them any further by using extra unlabeled data. Chinese word segmentation is one of this kind of tasks, since several large-scale manually annotated corpora are publicly available. In this chapter, we first exploit unlabeled in-domain data to improve strong supervised models. We leave domain adaptation for our future work.

We introduce the third type of unlabeled data with a *transductive learning*, *document-level* view. Many applications of word segmentation involve processing a whole document, such as information retrieval. In this situation, the text of the current document can provide additional useful information to segment a sentence. Take the word "氨纶丝/elastane" for example[1]. As a translated terminology word, it lacks compositionality. Moreover, this word appears rarely in general texts. As a result, if it does not appear in the training data, it is very hard for statistical models to recognize this word. Nevertheless, when we deal with an article discussing an elastane company, this word may appear more than once in this article, and the document information can help recognize this word. This idea is closely related to transductive learning in the sense that the segmentation model knows something about the problem it is going to resolve. We are also concerned with enhancing word segmentation with the document information.

We present a unified "feature engineering" approach for learning segmentation models from both labeled and unlabeled data. Our method is a simple two-stage process. First, we use unannotated corpus to extract string and document information, and then we use these information to construct new statistics-based and document-based feature mapping for a discriminative word segmenter. We are relying on the ability of discriminative learning method to identify and explore informative features, which play a central role to boost the segmentation performance. This simple solution has been shown effective for named entity recognition [Miller et al., 2004] and dependency parsing [Koo et al., 2008]. In their implementations, word clusters derived from unlabeled data are imported as features to discriminative learning approaches.

4.3 Feature Design

4.3.1 Baseline Features

Key to our approach is to allow informative features derived from unlabeled data to assist the segmenter. In our experiments, we employed three different feature sets: a baseline feature set which draws upon "normal" information from training data, a statistics-based feature set that uses statistical information derived from a large-scale

[1]This example is from an article indexed as chtb_0041 in the Penn Chinese Treebank corpus.

in-domain corpus, and a document-based feature set that uses information encoded in the surrounding text.

We already introduce a set of good feature templates for purely supervised character-based segmentation in Chapter 2. For the experiments in this chapter, our baseline feature set includes them all, as well as one idiom feature:

- Does c_i locate at the beginning of, inside or at the end of an idiom? If the string $c_{[s:i]}$ $(s < i)$ matches an item from the idiom lexicon, the feature template receives a string value "E". Similarly, we can define when this feature ought to be set to "B" or "I". Note that all idioms are larger than one character, so there is no "S" feature here.

4.3.2 Statistics-based Features

In order to distill information from unlabeled data, we borrow ideas from some previous research on unsupervised word segmentation. The statistical information acquired from a relatively large amount of unlabeled data are designed as features correlated with the position where a character locates in a word token. These features are based on three widely used criteria.

4.3.2.1 Mutual Information

Empirical mutual information is widely used in NLP. Informally, mutual information compares the probability of observing x and y together with the probabilities of observing x and y independently. If there is a genuine association between x and y, the $I(x, y) = \log \frac{p(x,y)}{p(x)p(y)}$ should be greater than 0.

Some previous work claimed that the larger the mutual information between two consecutive strings, the higher the possibility of the two strings being combined together. We adopt this idea in our character-based segmentation model. The empirical mutual information between two character bigrams is computed by counting how often they appear in the large-scale unlabeled corpus. Given a Chinese character string $c_{[i-2:i+1]}$, the mutual information between substrings $c_{[i-2:i-1]}$ and $c_{[i:i+1]}$ is computed as:

$$MI(c_{[i-2:i-1]}, c_{[i:i+1]}) = \log \frac{p(c_{[i-2:i+1]})}{p(c_{[i-2:i-1]})p(c_{[i:i+1]})}$$

For each character c_i, we incorporate the MI of the character bigrams into our model. They include,

- $MI(c_{[i-2:i-1]}, c_{[i:i+1]})$,

- $MI(c_{[i-1:i]}, c_{[i+1:i+2]})$.

4.3.2.2 Accessor Variety Features

When a string appears under different linguistic environments, it may carry a meaning. This principle is introduced as the *accessor variety* criterion for identifying meaningful Chinese words in [Feng et al., 2004]. This criterion evaluates how independently a string is used, and thus how likely it is that the string can be a word. Given a string s, which consists of l ($l \geq 2$) characters, we define the *left accessor variety* of $L^l_{av}(s)$ as the number of distinct characters that precede s in a corpus. Similarly, the *right accessor variety* $R^l_{av}(s)$ is defined as the number of distinct characters that succeed s.

We first extract all strings whose length are between 2 and 4 from the unlabeled data, and calculate their accessor variety values. For each character c_i, we then incorporate the following information into our model,

- Accessor variety of strings with length 4: $L^4_{av}(c_{[i:i+3]})$, $L^4_{av}(c_{[i+1:i+4]})$, $R^4_{av}(c_{[i-3:i]})$, $R^4_{av}(c_{[i-4:i-1]})$;
- Accessor variety of strings with length 3: $L^3_{av}(c_{[i:i+2]})$, $L^3_{av}(c_{[i+1:i+3]})$, $R^3_{av}(c_{[i-2:i]})$, $R^3_{av}(c_{[i-3:i-1]})$;
- Accessor variety of strings with length 2: $L^2_{av}(c_{[i:i+1]})$, $L^2_{av}(c_{[i+1:i+2]})$, $R^2_{av}(c_{[i-1:i]})$, $R^2_{av}(c_{[i-2:i-1]})$.

4.3.2.3 Punctuation Features

Punctuation marks are symbols that indicate the structure and organization of written language, as well as intonation and pauses to be observed when reading aloud. Punctuation marks can be taken as perfect word delimiters. The preceding and succeeding strings of punctuations carry additional wordbreak information, since punctuations should be segmented as a word. Note that such information is biased because not all words can appear before or after punctuations. For example, punctuations can not be followed by particles, such as "了", "着" and "过" which are indicators of aspects. Nevertheless, our experiments will show this kind of information is still useful for word segmentation.

When a string appears many times preceding or succeeding punctuations, there tends to be wordbreaks succeeding or preceding that string. To utilize the wordbreak information provided by punctuations, we extract all strings with length $l(2 \leq l \leq 4)$

which precede or succeed punctuations in the unlabeled data. We define the *left punctuation variety* of $L^l_{pv}(s)$ as the number of times a punctuation precedes s in a corpus. Similarly, the *right punctuation variety* $R^l_{pv}(s)$ is defined as the number of how many times a punctuation succeeds s. These two variables evaluate how likely a string can be separated at its start or end positions.

We first gather all strings surrounding punctuations in the unlabeled data, and calculate their punctuation variety values. The length of each string is also restricted between 2 and 4. For each character c_i, we import the following information into our model,

- Punctuation variety of strings with length 4: $L^4_{pv}(c_{[i:i+3]})$, $R^4_{pv}(c_{[i-3:i]})$;
- Punctuation variety of strings with length 3: $L^3_{pv}(c_{[i:i+2]})$, $R^3_{pv}(c_{[i-2:i]})$;
- Punctuation variety of strings with length 2: $L^2_{pv}(c_{[i:i+1]})$, $R^2_{pv}(c_{[i-1:i]})$.

Punctuations can be viewed as *mark-up's* of Chinese text. Our motivation to use the punctuation information to assist a word segmenter is similar to [Spitkovsky et al., 2010] in a way to explore "artificial" word (or phrase) break symbols. In their work, four common HTML tags are successfully used as raw phrase bracketings to improve unsupervised dependency parsing.

4.3.2.4 Binary or Numeric Features

The derived information introduced above is all expressed as real values. The natural way to incorporate these statistics into a discriminative learning model is to directly use them as numeric features. However, our experiments show that this simple choice does not work well. The reason is that these statistics actually behave non-linearly to predict character labels. For each type of statistics, one weight alone cannot capture the relation between its value and the possibility that a string forms a word. Instead, we represent these statistics as discrete features.

For the mutual information, this is done by rounding down decimal number. The integer part of each MI value is used as a string feature. For the accessor variety and punctuation variety information, since their values are integer, we can directly use them as string features. The accessor variety and punctuation variety could be very large, so we set thresholds to cut off large values to deal with the data sparse problem. Specially, if an accessor variety value is greater than 50, it is incorporated as a feature "> 50"; if the value is greater than 30 but not greater than 50, it is incorporated as a feature "$30 - 50$"; else the value is individually incorporated as a string feature. For example, if the left accessory variety of a character bigram $c_{[i:i+1]}$ is 29, the binary feature "$L^2_{av}(c_{[i:i+1]})$=29" will be set to 1, while other related binary features such as "$L^2_{av}(c_{[i:i+1]}) = 15$" or "$L^2_{av}(c_{[i:i+1]}) > 50$" will be set to 0. Similarly, we can discretize the punctuation variety features. However, we only set one threshold, 30, for this value. These thresholds can be tuned by using held-out data.

4.3.3 Document-based Features

It is meaningless to derive statistics of a document and use it for word segmentation, since most documents are relatively short, and values are statistically unreliable. Our experiments confirm this idea. Instead, we propose the following binary features which are based on the *string count* in the given document that is simply the number of times a given string appears in that document. For each character c_i, our document-based features include,

- Whether the string count of $c_{[s:i]}$ is equal to that of $c_{[s:i+1]}$ $(i - 3 \leq s \leq i)$. Multiple features are generated for different string length.
- Whether the string count of $c_{[i:e]}$ is equal to that of $c_{[i-1:e]}$ $(i \leq e \leq i + 3)$. Multiple features are generated for different string length.

The intuition is as follows. The string counts of $c_{[s:i]}$ and $c_{[s:i+1]}$ being equal means that when $c_{[s:i]}$ appears, it appears inside $c_{[s:i+1]}$. In this case, $c_{[s:i]}$ is not independently used in this document, and this feature suggests the segmenter not assign a "S" or "E" label to the character c_i. Similarly, the string counts of $c_{[i:e]}$ and $c_{[i-1:e]}$ being equal means $c_{[i:e]}$ is not independently used in this document, and this feature suggests segmenter not assign a "S" or "B" label to c_i. We do not directly use the string counts to prevent a bias towards longer documents.

4.4 Experiments and Analysis

4.4.1 Setting

The SIGHAN Bakeoffs provide several large-scale labeled data for the research on Chinese word segmentation. Although these data sets are labeled on continuous run texts, they do not contain the document boundary information. CTB is a segmented, POS tagged, and fully bracketed corpus in the constituency formalism. It is also an popular data set to evaluate word segmentation methods, such as [Jiang et al., 2009;

Sun, 2011]. CTB is a collection of documents which are separately annotated. This annotation style allows us to calculate the so-called document-based features and to further evaluate our approach. In this chapter, we use CTB 6.0 as our main corpus and follow the CoNLL 2009 shared task to define the training, development and test sets[1]. In Chapter 3, we showed the statistics of this setting in Table 3.8.

In previous Chapters 3, 5 and 8, the CTB 6.0 data used comes from the CoNLL 2009 shared task or its associated parts. Here, the CTB data is extracted from the file list provided by the organizer of the shared task. There is a small difference between the data provided by the shared task and the data extracted from the file list: The development data extracted from the file list includes one more sentence (namely 1973 sentences in total). This small difference does not affect much and it is still reasonable to compare the experimental results in this section to the previous reported results.

Chinese Gigaword is a comprehensive archive of newswire text data that has been acquired over several years by the Linguistic Data Consortium (LDC). The large-scale unlabeled data we use in our experiments comes from the Chinese Gigaword (LDC2005T14). We choose the Mandarin news text, i.e. Xinhua newswire. This data covers all news published by Xinhua News Agency (the largest news agency in China) from 1991 to 2004, which contains over 473 million characters.

F-score is used as the accuracy measure. Define precision p as the percentage of words in the decoder output that are segmented correctly, and recall r as the percentage of gold standard output words that are correctly segmented by the decoder. The (balanced) F-score is $2pr/(p+r)$. We also report the recall of OOV words. Note that, all idioms in our extra idiom lexicon are added into the in-vocabulary word list.

CRFsuite [Okazaki, 2007] is an implementation of Conditional Random Fields (CRFs) [Lafferty et al., 2001] for labeling sequential data. It is a speed-oriented implementation, which is written in pure C. In our experiments, we use this toolkit to learn global linear models for segmentation. We use the stochastic gradient descent algorithm to resolve the optimization problem, and set default values for other learning parameters.

4.4.2 Main Results

Table 4.1 summarizes the segmentation results on the development data with different configurations, representing a few choices between baseline, statistics-based and

[1]We would like to thank Prof. Nianwen Xue for the help with the division of the data.

Devel.	P	R	$F_{\beta=1}$	R_{oov}
Baseline	95.41	95.52	95.46	77.68
+MI	95.50	95.48	95.49	77.98
+AV(2)	95.85	96.04	95.94	79.31
+AV(2,3)	95.95	96.19	96.07	80.61
+AV(2,3,4)	96.14	95.99	96.07	81.83
+PU(2)	95.86	96.07	95.97	79.70
+PU(2,3)	95.98	96.25	96.11	80.42
+PU(2,3,4)	96.00	96.19	96.10	80.53
+MI+AV(2,3,4)+PU(2,3,4)	96.17	96.22	96.19	80.42
+DOC	95.69	95.64	95.66	79.89
+MI+AV(2,3,4)+PU(2,3,4)+DOC	96.21	96.23	96.22	81.75

Table 4.1: Segmentation performance with different feature sets on the development data. Abbreviations: MI=mutual information; AV=accessor variety; PU=punctuation variety; DOC=document features. The numbers in each bracket pair are the lengths of strings. For example, PU(2,3) means punctuation variety features of character bigrams and trigrams are added.

document-based feature sets. In this table, the symbol "+" means features of current configuration contains both the baseline features and new features for semi-supervised or transductive learning. From this table, we can clearly see the impact of features derived from the large-scale unlabeled data and the current document. Comparison between the performance of the baseline and "+MI" shows that the widely used mutual information is not helpful. Both good segmentation techniques and valuable labeled corpora have been developed, and pure supervised systems can provide strong performance. It is not a trial to design new features to enhance supervised models.

There are significant increases when accessor variety features and punctuation variety features are separately added. Extending the length of neighboring string helps a little from 2 to 3. Although the OOV recall increases when the length is extended from 3 to 4, there is no improvement of the overall balanced F-score. The line "+MI+AV(2,3,4)+PU(2,3,4)" shows the performance when all statistics-based features are added. The combination of the "AV" and "PU" features gives further helps. This system can be seen as a pure semi-supervised system. The line "+DOC" is the result when document-based features are added. In spite of its simplicity, the document-based features can help the task. However, when we combine statistics-based features with document-based features, we cannot get further improvement in terms of F-score.

Table 5.14 shows the segmentation performance on the test data set. The final results of our system are achieved with the "+MI+AV(2,3,4)+PU(2,3,4)+DOC" fea-

Test	P	R	$F_{\beta=1}$	R_{oov}
Baseline	95.21	94.90	95.06	75.52
Final	95.86	95.62	95.74	79.28

Table 4.2: Segmentation performance on the test data.

ture configuration. The new features result in relative error reductions of 13.8% and 15.4% in terms of the balanced F-score and the recall of OOV words respectively.

4.4.3 Learning Curves

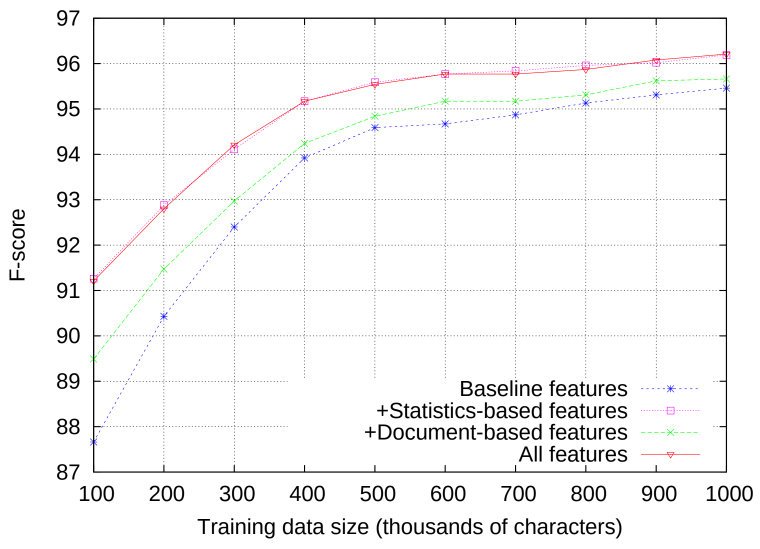


Figure 4.1: The learning curves (F-score) of different models.

We performed additional experiments to evaluate the effect of the derived features as the amount of labeled training data is varied. Figure 4.1 and 4.2 display the F-score and the OOV recall of systems with different feature sets when trained on smaller portions of the labeled data. We can clearly see that the derived features obtain consistent gains regardless of the size of the labeled training set. Both statistics-based features and document-based features can help improve the overall performance. Especially, they can help to recognize more unknown words, which is important for many applications. The F-score of semi-supervised models, i.e. models trained with statistics-based features, does not achieve further improvement when

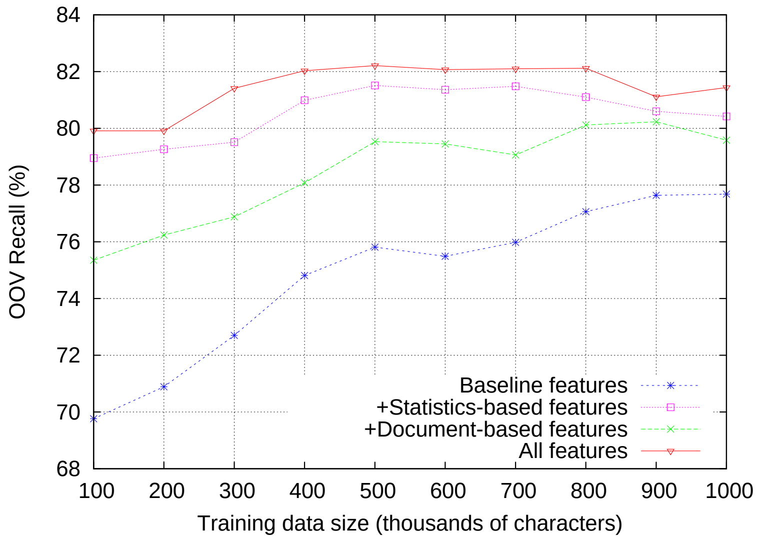


Figure 4.2: The learning curves (Recall of OOV) of different models.

document-based features are added. Nonetheless, the OOV recall obtains slightly improvements.

It is interesting to consider the amount by which derived features reduce the need for supervised data, given a desired level of accuracy. The change of the F-score in Figure 4.1 suggests that derived features reduce the need for supervised data by roughly a factor of 2. For example, the performance of the model with extra features trained on 500k characters is slightly higher than the performance of the model with only baseline features trained on the whole labeled data.

4.4.4 Feature Analysis

We discussed the choice of using binary or numeric features in Section 4.3.2.4. In our experiment, when the accessor variety and punctuation variety information are integrated as numeric features, they do not contribute. To show the non-linear way that these features contribute to the prediction problem, we present the scatter plots of the score of each feature (i.e. the weight multiply the feature value) against the value of the feature. Figure 4.3 shows the relation between the score and the value of the punctuation variety features. For example, the weight of the binary feature "$L_{pu}^{2}(c_{[i:i+1]}) = 26$ combined with the label "B" learned by the final model

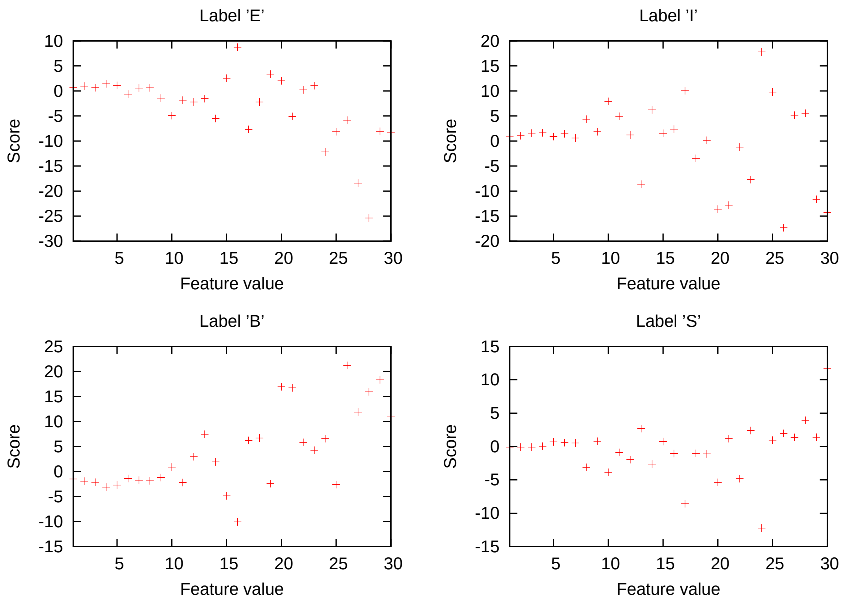


Figure 4.3: Scatter plot of feature ($L^2_{pv}(c_{[i:i+1]})$ score against feature value.

is 0.815141, so the score of this combination is $0.815141 \times 26 = 21.193666$ and a point $(26, 21.193666)$ is drawn. These plots indicate the punctuation variety features contribute to the final model in a very complicated way. It is impossible to use one weight to capture it. The accessor variety features affect the model in the same way, so we do not give detailed discussions. We only show the same scatter plot of the $L^2_{av}(c_{[i:i+1]})$ feature template in Figure 4.4.

4.5 Related Work

Xu et al. [2008] presented a Bayesian semi-supervised approach to derive task-oriented word segmentation for machine translation (MT). This method learns new word types and word distributions on unlabeled data by considering segmentation as a hidden variable in MT. Different from their concern, our focus is general word segmentation.

The "feature-engineering" semi-supervised approach has been successfully applied to named entity recognition [Miller et al., 2004] and dependency parsing [Koo et al., 2008]. These two papers demonstrated the effectiveness of using word clusters as features in discriminative learning. Moreover, Turian et al. [2010] compared different

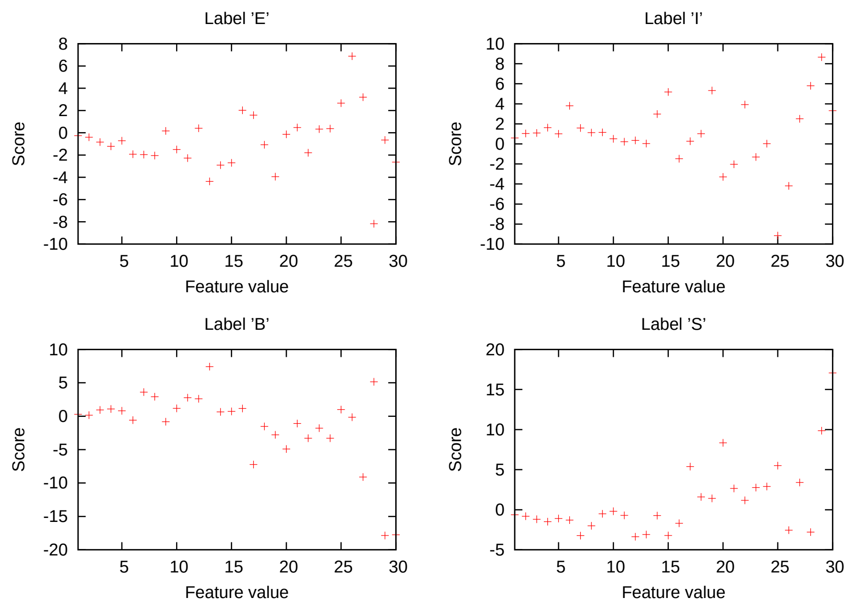


Figure 4.4: Scatter plot of feature score against feature value for $L_{av}^{2}(c_{[i:i+1]})$.

word clustering algorithms and evaluated their effect on both named entity recognition and text chunking.

As mentioned earlier, the feature design is inspired by some previous research on word segmentation. The accessor variety criterion is proposed to extract word types, i.e. the list of possible words, in [Feng et al., 2004]. Different from their work, our method resolves the segmentation problem of running texts, in which this criterion is used to define features correlated with the character position labels. Li and Sun [2009] observed that punctuations are perfect delimiters which provide useful information for segmentation. Their method can be viewed as a self-training procedure, in which extra punctuation information is incorporated to filter out automatically predicted samples. We use the punctuation information in a different way. In our method, the counts of the preceding and succeeding strings of punctuations are incorporated directly as features into a supervised model.

In machine learning, transductive learning is a learning framework that typically makes use of unlabeled data. The goal of transductive learning is to only infer labels for the unlabeled data points in the test set rather than to learn a general classification function that can be applied to any future data sets. This means that the test data is known as a priori knowledge and can be used to construct better hypotheses.

Although the idea to explore the document-level information in our work is similar to transductive learning, we do not use state-of-the-art transductive learning algorithms which involve learning when they meet the test data. For real-world applications, our approach is efficient by avoiding re-training.

4.6 Discussion: Unsupervised Language Acquisition for Supervised Language Processing

Both supervised language processing and unsupervised language acquisition systems achieve quite good performance for some applications. There are a lot of good examples of the first one, which is trained fairly well on labeled data, such as English Penn Treebank style parsing. There are also some successful unsupervised NLP algorithms, which sometimes achieve equivalent performance to supervised ones. Two of early good examples are word sense disambiguation [Yarowsky, 1995] and automatic retrieval of similar words [Lin, 1998].

Feature induction based semi-supervised learning to some extent bridges the gap between supervised language processing and unsupervised language acquisition. At its most abstract, language acquisition is simply a mapping from some input to some linguistic knowledge be used in the generation and interpretation of new utterances. Discriminative learning allows us to easily use rich linguistic knowledge derived from unlabeled data with a wide range of unsupervised language acquisition algorithms.

Two types of unsupervised language acquisition are very popular: (1) grammar induction concerning syntactic structures and (2) lexical acquisition concerning the word knowledge. Previous study, as well as our work, mainly focus on the latter. However, we think grammar induction could also provide valuable information for syntactic processing or higher-level, semantic processing, even considering the quality of automatically acquired grammars is not very good.

Bilingual (or multilingual) data has been shown useful to improve monolingual language processing, such as [Das and Petrov, 2011]. Considering unsupervised language acquisition, bilingual or multilingual data could also be useful resources. For example, the word clusters learned from bilingual data are more helpful for machine translation than the clusters independently learned from each side of language data [Och, 1999]. It is obvious that this idea can be extended to derive linguistic knowledge from bilingual or even multilingual data.

Although we show that string knowledge can help Chinese word segmentation.

There are still some methodological questions unsolved. Among them we raise two interesting topics. The first topic is how to do principled feature engineering. In our case, the baseline feature set are only derived from the surface strings. To extend these *simple* feature set is relatively simple. However, for some other complex tasks, such as semantic role labeling, the baseline feature set are very complex (as we will show in Chapter 8). In this situation, the non-trivial feature engineering need more research efforts and especially principled methods. How to design and select good new features will significantly affects the impact of the additional information. The second topic is how to motivate application-oriented language acquisition. Most existing unsupervised language acquisition research focus on general linguistic principles. For different tasks, different task-specific knowledge is needed. It is natural idea that careful designs of application-oriented language acquisition will help more.

Part II

Syntactic Parsing

Chapter 5

Comparing and Integrating Heterogeneous Parsers

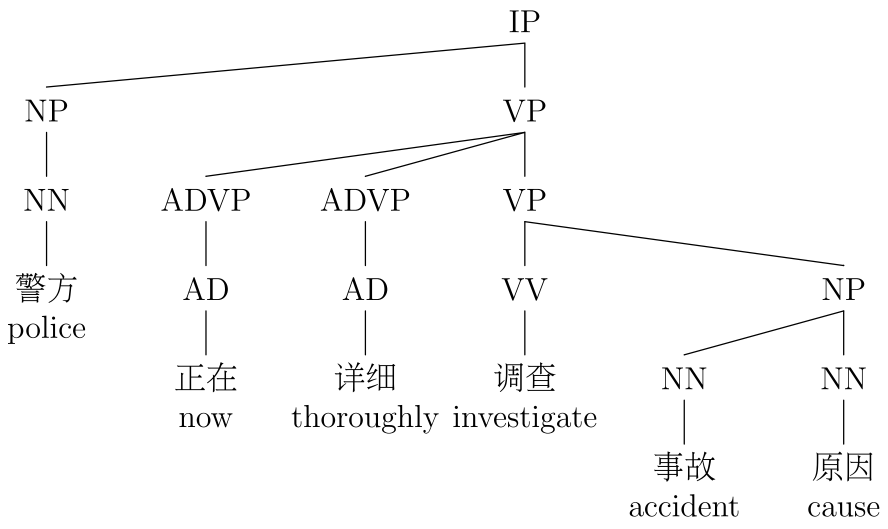
We study heterogeneous syntactic analyzing methods in this chapter. We first present a comparative analysis of state-of-the-art methods for POS tagging, constituency and dependency parsing. We show that due to their theoretical properties, heterogeneous models behave very differently and produce different error distributions and have complementary predictive powers. The analysis motivates us to investigate ensemble methods to improve processing accuracy by integrating different types of analyzers. To enhance POS tagging, we propose a Bagging model to combine the complementary strengths of syntax-free and syntax-based taggers. To enhance dependency parsing, we evaluate a previously introduced stacking method and propose a more effective Bagging model to integrate grammar-free and grammar-based parsers. Experiments on the Penn Chinese Treebank demonstrate the effectiveness of our methods.

The comparison and combination of syntax-free and syntax-based methods for POS tagging is originally introduced in [Sun and Uszkoreit, 2012].

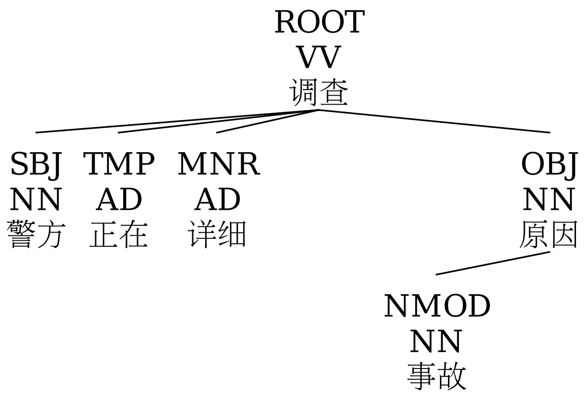
5.1 Background

5.1.1 The Problem

In a broad sense, parsing means taking an input sentence and producing some sort of linguistic analysis for it, including many kinds of structures that might be produced: morphological, syntactic, semantic, discourse. As one of the core issues of NLP, syntactic parsing is the task to assign grammatical structures to sentences, for instance, which groups of words go together as phrases and which words are the subject or



(1) The constituency parse.



(2) The dependency parse.

Figure 5.1: An example sentence with constituency and dependency structures: *The police are thoroughly investigating the cause of the accident.*

object of a particular predicate.

In grammar, a POS is a linguistic category of words, which is generally defined by the syntactic or morphological behavior of the word in question. The significance of POS's for language processing is the large amount of information they give about a word and its neighbors. Constituency grammar arranges sentences into a hierarchy of nested phrases. A phrase structure is normally represented as a constituent tree. At the lowest level of a constituency tree, each word is treated as a one-word phrase that is labeled by its POS tag. At higher levels, successively larger phrases are created by concatenating smaller phrases, culminating in a phrase covering the entire sentence. Dependency grammar formalizes syntactic structure as a directed tree of bilexical dependencies, which determines relations between head words and their dependents. Dependency grammar is less complex than lexicalized phrase-structure grammar, since head-modifier interactions are modeled directly without introducing the scaffold of phrase-structure grammar.

Figure 5.1 depicts constituency and dependency parses of a simple sentence. In the constituency tree, each non-terminal node represents a constituent, and its children represent its intermediate components. For example, the phrase "事故原因/the cause of the accident" is the composition of "事故/accident" and "原因/cause." In the dependency tree, each edge represent a dependency where the upper word is the head and the lower word is the dependent. For example, the edge between "事故" and "原因" indicates that "事故" is dependent on and modifies "原因."

It is generally accepted that finding syntactic structures is useful in determining the meaning of a sentence. For example, constituency parse trees serve as an important intermediate stage of representation for predicate-argument structure analyzing (as we will show in Chapter 8). Therefore most NLP applications (such as information extraction, machine translation, or speech recognition) would almost certainly benefit from high-accuracy parsing.

5.1.2 Previous Work

Many successful tagging and parsing algorithms designed for English have been applied to many other languages as well. In some cases, the methods work well without large modifications, such as German POS tagging. But a number of augmentations and changes became necessary when dealing with highly inflected or agglutinative languages, as well as analytic languages, of which Chinese is the focus of this thesis.

Both discriminative and generative models are explored for accurate Chinese POS tagging [Huang et al., 2009, 2007; Tseng et al., 2005b]. Tseng et al. [2005a] introduced a maximum entropy based model, which includes morphological features for unknown word recognition. Huang et al. [2007] and Huang et al. [2009] mainly focused on the generative HMM models. To enhance a HMM model, Huang et al. [2007] proposed a re-ranking procedure to include extra morphological and syntactic features, while Huang et al. [2009] proposed a latent variable inducing model.

There have been several attempts to develop high quality parsers for Chinese in both constituency and dependency formalisms [Bikel and Chiang, 2000; Huang and Sagae, 2010; Levy and Manning, 2003; Li et al., 2011; Petrov and Klein, 2007; Zhang and Clark, 2008b], but the state-of-the-art performance on Penn Chinese Treebank (CTB), achieved by the Berkeley parser and the higher order graph-based dependency parser falls far short when compared to English. Previous work mainly focuses on how to implement methods which are successful on English, which reach early success. As pointed out in [Levy and Manning, 2003], there are many linguistic differences between Chinese and English, as well as structural differences between their corresponding treebanks, and some of these make it a harder task to parse Chinese. Although some language-specific properties are preliminarily discussed, it is still very unclear what are the main difficulties for the phrase-strucutre analyzing and whether good algorithms for English processing are suited for the Chinese problems?

5.2 State-of-the-Art

In this section, we give a brief introduction to state-of-the-art syntactic analyzing methods for Chinese language processing.

5.2.1 A Discriminative Sequential Model for POS Tagging

Many algorithms have been applied to computationally assigning POS labels to English words, including hand-written rules, HMM tagging and discriminative sequence labeling. While state-of-the-art tagging systems have achieved accuracies above 97% on English, Chinese POS tagging has proven to be more challenging [Huang et al., 2009, 2007; Li et al., 2011; Tseng et al., 2005b]. According to the *ACL Wiki*[1], all the state-of-the-art English POS taggers are based on discriminative sequence labeling models, including structure perceptron [Collins, 2002; Shen et al., 2007], maximum entropy [Toutanova et al., 2003] and SVM [Gimnez and Mrquez, 2004]. A discriminative learner is easy to be extended with arbitrary features and therefore suitable to recognize more new words. Moreover, a majority of the POS tags are locally dependent on each other, so the Markov assumption can well captures the syntactic relations among words. Discriminative learning is also an appropriate solution for Chinese POS tagging, due to its flexibility to include knowledge from multiple linguistic sources. For example, we will show that word clusters which are automatically induced from unlabeled raw text can enhance Chinese POS tagging in Chapter 7.

In Chapter 3, we studied the joint word segmentation and POS tagging problem

[1]http://newdesign.aclweb.org/aclwiki/index.php?title=POS_Tagging_(State_of_the_art).

截止	目前	保险	公司	已	为	三峡	工程	提供	保险	服务
P	NT	NN	NN	AD	P	NR	NN	VP	NN	NN

Figure 5.2: An example of Chinese POS tagging: *Until now, the insurance company has provided insurance services for the Sanxia Project.*

w_{-2}=“截止”; w_{-1}=“目前”; w=“保险”; w_{+1}=“公司”; w_{+2}=“已”;
w_{-2}_w_{-1}=“截止_目前”; w_{-1}_w=“目前_保险”; w_w_{+1}=“保险_公司”; w_{+1}_w_{+2}=“公司_已”;
Prefix(1)=“ 3 ”; Prefix(2)=“ 3 5 ”; Prefix(3)=“ 3 5 5 ” Suffix(1)=“ 5 ”; Suffix(2)=“ 5 5 ”; Suffix(3)=“ 3 5 5 ”

Table 5.1: An example of features used for POS tagging.

and developed a fully discriminative method. However, we did not deeply analyze the problem from a linguistic view. To this end, we isolate the POS tagging problem and study how to build an accurate POS tagger on perfect word segmentation. In our experiments, we employ a simple feature set which draws upon information sources such as word forms and characters that constitute words. To conveniently illustrate, we denote a word in focus with a fixed window $w_{-2}w_{-1}ww_{+1}w_{+2}$, where w is the current token. The baseline features includes:

- Word unigram feature: w_{-2}, w_{-1}, w, w_{+1}, w_{+2};
- Word bigram feature: $w_{-2}\_w_{-1}$, w_{-1}_w, w_w_{+1}, $w_{+1}\_w_{+2}$.
- In order to better handle unknown words, we also extract morphological features: character n-gram prefixes and suffixes for n up to 3.

That means 15 features are used to represent a given word token. When different amount of data is available, the best configuration of feature template varies. Normally, larger window of context leads to improved accuracy when more labeled data is available. This setting can be tuned on the development data. In our experiments on the CTB 6.0, the window size is tuned to be set to 2. Take the word “保险” in Figure 5.2 for example, all features are listed in Table 5.1.

5.2.2 A Generative PCFG-LA Model for Constituency Parsing

Comparing with many other languages, statistical parsing for Chinese has reached early success, due to the fact that the language has relatively fixed word order and extremely poor inflectional morphology. Both facts allow the PCFG-based statistical modeling to perform well for constituency parsing. On the other hand, the much higher ambiguity between basic word categories like nouns and verbs makes Chinese parsing interestingly different from the situation of English.

For the constituency parsing, the majority of the state-of-the-art parsers are based on generative PCFG learning. For example, the well-known and successful parsing models developed by Collins [Collins, 2003] and Charniak [Charniak, 2000] implement generative lexicalized statistical models. Based on the N-best lists generated by these parsers, a discriminative reranker can help further improve the parsing quality [Charniak and Johnson, 2005; Collins and Koo, 2005; Huang, 2008]. However, pure discriminative constituency parsing models are limited to the huge search space and are not yet well developed. Apart from complex lexicalized PCFG parsing, unlexicalized parsing with latent variable grammars (PCFG-LA) can also produce comparable accuracy [Matsuzaki et al., 2005; Petrov et al., 2006; Petrov and Klein, 2007]. Latent variable grammars for parsing model an observed treebank of coarse parse trees with a model over more refined, but unobserved, derivation trees. Given sentences as input, the parse trees represent the desired output of the system, while the derivation trees represent much more complex syntactic processes. For example, the single Penn TreeBank category NP (noun phrase) may be better modeled by several *sub*-categories representing subject NPs, object NPs, and so on. Rather than attempting to manually specify these fine-grained categories, previous work shows that automatically inducing the sub-categories from data can work quite well.

Although the state-of-the-art lexicalized and unlexicalized parsing models work very differently, they are both inspired by the same strategy: Refining normal PCFG grammars. The former model refine statistical grammars with lexical information, while the latter one with the subcategory information. Compared to lexicalized parsers, the PCFG-LA parsers leverages on an automatic procedure to learn refined grammars and are therefore more robust to parse non-English languages that are not well studied. Take Chinese Penn TreeBank parsing as an example. A PCFG-LA parser achieves the state-of-the-art performance and defeat many other types of parsers such as discriminative transition-based models [Zhang and Clark, 2009]. The Berkeley parser is an open source implementation of the PCFG-LA model [Petrov et al., 2006; Petrov and Klein, 2007] and is used for experiments in this chapter.

5.2.3 A Discriminative Graph-based Model for Dependency Parsing

Dependency parsing, especially the statistical one, has recently gained a wide interest in the computational linguistics community. Data-driven approaches automatically learn to produce dependency graphs for sentences solely from an annotated treebank. The advantage of such models is that they are easily ported to any language in which labeled linguistic resources exist. Practically all statistical models that have been proposed in recent years can be described as either *graph-based* or *transition-based* [McDonald and Nivre, 2007]. For a set of languages, these two models achieve similar performance overall. In graph-based parsing, we learn a model for scoring possible dependency directed trees for a given sentence, typically by factoring the trees into their component edges, and execute parsing by searching for the highest-scoring tree. In transition-based parsing, we instead learn a model for scoring actions from one parse state to the next, conditioned on the parsing history, and execute parsing by incrementally, greedily taking the highest-scoring transition out of every parser state until we have derived a complete dependency graph. Both parsing models leverage discriminative learning to estimate parameters and flexible to include arbitrary features, and therefore easy to port to different languages.

Both graph-based and transition-based models are adopted to learn Chinese dependency structures [Huang and Sagae, 2010; Li et al., 2011; Zhang and Clark, 2008b]. In addition, as a sub-task of CoNLL 2009 [Hajič et al., 2009], various models are well evaluated. According Li et al. [2011]'s comparison of published results, graph-based and transition-based parsers achieve similar accuracies. In this paper, we choose a state-of-the-art second order graph-based dependency parser, i.e. mate parser[1] [Bohnet, 2010], for experiments.

5.3 Key Distinctions

5.3.1 Syntax-free and Syntax-based POS Tagging

Chinese POS tagging often requires more sophisticated language processing techniques that are capable of drawing inferences from more subtle linguistic knowledge. From a linguistic point of view, meaning arises from the differences between linguistic units, including words, phrases and so on, and these differences are of two

[1]https://code.google.com/p/mate-tools/

kinds: paradigmatic (concerning substitution) and syntagmatic (concerning positioning). The distinction is a key one in structuralist semiotic analysis. Both paradigmatic and syntagmatic lexical relations have a great impact on POS tagging, because the *value* of a word is determined by the two relations. Our error analysis of a state-of-the-art Chinese POS tagger shows that the lack of both paradigmatic and syntagmatic lexical knowledge accounts for a large part of tagging errors. Syntactic analysis, especially the full and deep one, reflects syntagmatic relations of words and phrases of sentences. Therefore syntactic parsing has a big contribution to lexical tagging. In this chapter, we will present a series of empirical studies of the tagging results of a *syntax-free* sequential tagger and a *syntax-based* chart parser, aiming at illuminating more precisely the possible impact of syntactic information on POS tagging. The analysis is helpful to understand the role of syntagmatic lexical relations in POS prediction.

5.3.2 Grammar-free and Grammar-based Dependency Parsing

Dependency parsing approaches can be divided into two classes, grammar-free and grammar-based. Grammar-free approaches, normally known as data-driven, make essential use of machine learning from linguistic annotations in order to parse new sentences. Such approaches, e.g. transition-based [Nivre, 2008], graph-based [McDonald, 2006] and ILP[1]-based [Martins et al., 2009], have attracted the most attention in recent years. In contrast, grammar-based approaches rely on formal grammars to shape the search space for possible syntactic analysis. The grammar may be hand-crafted or learned from linguistic annotations, which means that a grammar-based model may be data-driven as well. In particular, context-free grammar (CFG) based dependency parsing exploits a mapping between dependency structures and context-free phrase structure representations and reuses parsing algorithms developed for CFG, e.g. lexicalized PCFG models [Charniak, 2000; Collins, 2003], to produce dependency structures.

In previous work, grammar-free, discriminative parsing approaches have been widely discussed for Chinese dependency parsing. On the other hand, various PCFG-based constituency parsing methods have been applied to obtain phrase-structures as well. Since Chinese phrase-structures are adequate to be transformed to dependency structures, a constituency parser with a set of CS to DS conversion rules [Xue,

[1]Integer linear programming

2007] can be taken as a grammar-based dependency parser. Both dependency and constituency parsing technologies have improved considerably in the past few years for Chinese processing, but efforts to perform extensive comparisons of grammar-free and grammar-based views have been limited. In order to pave the way for new and better methods, a much more detailed empirical analysis is needed to understand the strengths and weaknesses of heterogeneous approaches. In this chapter, we will present a comparative analysis of two representative state-of-the-art parsers, in order to illuminate more precisely the properties of grammar-free and grammar-based parsing.

5.4 Experimental Setting

CTB is a segmented, POS tagged, and fully bracketed corpus in the constituency formalism. It is an popular data set to evaluate a number of Chinese NLP tasks, including word segmentation [Jiang et al., 2009; Sun, 2011; Sun and Xu, 2011], POS tagging [Huang et al., 2009, 2007], constituency parsing [Wang et al., 2006; Zhang and Clark, 2009], dependency parsing [Huang and Sagae, 2010; Li et al., 2011; Zhang and Clark, 2008b] and function tag labeling [Sun and Sui, 2009]. In this chapter, we use CTB 6.0 as the labeled data for the study. The corpus was collected during different time periods from different sources with a diversity of topics. In order to obtain a representative split of data sets, we define the training, development and test sets following two settings. To compare our tagger with the state-of-the-art, we conduct some experiments using the data setting of [Huang et al., 2009].[1] We divide all CTB files into blocks of 10 files in sorted order, and of each block the first file is used as development data, the second as test, and the remaining for training. Table 5.2 shows the statistics of this experimental setting.

Data	#sent.	#words	#char.
Training	24416	678811	900033
Devel.	1904	51229	83252
Test	1975	52861	86763

Table 5.2: Training, development and test data on CTB 6.0 (setting 1).

For detailed analysis of different syntactic analyzing methods, we conduct further experiments following the setting of the CoNLL 2009 shared task. The setting is provided by one of the organizer of the CTB project, and considers many annotation

[1] We would like to thank Zhongqiang Huang to help with the preparation of the data sets.

details. This setting is more robust for evaluation different Chinese language processing algorithms. In Chapter 3, we have presented some information of this setting. Please refer to Section 3.3.7 for more details. The syntactic annotation of the CTB project also includes information about empty categories. Modern statistical parsers such as Collins, Charniak and Berkeley parsers ignore this type of linguistic information. To train and evaluate a constituency parser, we apply a heuristic procedure on the treebank data to delete empty categories and its associated redundant ancestors. Since the CTB is annotated with phrase structures, an extra conversion is necessary to prepare data sets for dependency parsing. To evaluate dependency parsing, we directly use the CoNLL data.

In the following experiments, a first order linear-chain CRF model is used to resolve the POS tagging problem. We use the CRF learning toolkit *wapiti* [1] [Lavergne et al., 2010] to train global linear models. Among several parameter estimation methods provided by wapiti our experiments show that the "rprop-" method work best. We use this algorithm and let other setting default. For constituency parsing experiments, we use the Berkeley parser[2].

For the evaluation of constituency parsing, we used a graphical tool EvalC[3]. We report labeled precision (P), labeled recall (R) and f-score (which is the harmonic mean of P and R) to measure the phrase recovery accuracy. The balanced f-score (F) is defined by 2PR/(P+R). For the evaluation of dependency parsing, we use the evaluation tool[4] provided by the CoNLL 2006 shared task. Two evaluation metrics for dependency parsing are reported: (1) the unlabeled attachment score (UAS), i.e., the percentage of tokens with correct head word prediction, and (2) the labeled attachment score (LAS), i.e. the percentage of "scoring" tokens for which the system has predicted the correct head word as well as its relation type.

[1]http://wapiti.limsi.fr/

[2]http://code.google.com/p/berkeleyparser/

[3]http://staff.science.uva.nl/~fsangati/evalC_25_05_10.zip

[4]http://ilk.uvt.nl/conll/software/eval.pl

5.5 Comparing and Combining Syntax-free and Syntax-based Tagging Models

5.5.1 Overall Performance

5.5.1.1 Discriminative Learning is Competitive for POS Tagging

Table 5.3 summarizes the per token classification accuracy (Acc.) of our tagger and state-of-the-art results reported in [Huang et al., 2009]. Huang et al. [2009] introduced a bigram HMM model with latent variables (*Bigram HMM-LA* in the table) for Chinese tagging. Compared to earlier work [Huang et al., 2007], this model achieves the state-of-the-art accuracy. This model can be further enhanced by using unlabeled data via self-training (*Bigram HMM-LA+ST* in the table). Despite of simplicity, our discriminative POS tagging model achieves a state-of-the-art performance, even better.

System		Model	Acc.
[Huang et al., 2009]	Supervised	Trigram HMM	93.99%
		Bigram HMM-LA	94.53%
	Semi-supervised	Bigram HMM-LA+ST	94.78%
Our tagger	Supervised	Discriminative Tagging	94.69%

Table 5.3: Tagging accuracy on the test data (setting 1).

5.5.1.2 The Impact of POS Tagging on Parsing

Table 5.4 summarizes the word classification accuracies of our discriminative POS tagger and the Berkeley parser using the CoNLL setting. We can see that Chinese constituency parsing can reach a reasonably good result, when gold POS information is available. However, Chinese POS tagging is currently far from perfect. In our experiments, when automatic tagging information is used, the overall parsing performance drops more than 6 absolute points.

		Tagging	Parsing		
Tagger	Parser	Acc.	P	R	F
Berkeley Parser	Berkeley Parser	93.69%	82.44%	80.31%	81.36
CTB	Berkeley Parser	99.83%	88.16%	86.85%	87.50
Our tagger	Berkeley Parser	94.48%	80.55%	79.60%	80.07↓

Table 5.4: Parsing accuracy on the development data.

The overall tagging accuracy of our tagger is 94.48%, which is significantly better than the Berkeley parser (93.69%), though the full parser can use additional syntactic information to do disambiguation. Since the overall performance of our tagger is better, we may guess the parsing accuracy can be simply improved by inputting the Berkeley parser with our POS tagging results. Unfortunately, this is not true. The last line in Table 5.4 shows the new parsing result, while the first line is the performance of the pure Berkeley parser. Comparing the f-scores, we can see a very clear decline when the POS tags are substituted. This experiment suggests not only that the full parser is very sensitive to the POS tagging errors but also that the errors made by the tagger and the parser are very different.

This result is to some extent similar to the experimental results reported in [Li et al., 2011]. The motivation of that paper is to improve the dependency parsing via joint prediction of POS tags and bi-lexical dependencies. However, although the dependency prediction is improved by using a much more complex joint model, the performance of POS tagging goes down. They concludes that their joint learning method for tagging and parsing hurts the tagging accuracy, and that the more syntactic features the joint method incorporates, the more the tagging accuracy drops. Note that, their method is in a discriminative learning architecture, while the Berkeley parser is in a generative one. It seems that the reason of the inverse reaction is caused by the properties of the special language or the annotation strategies rather than particular machine learning algorithms.

5.5.2 Comparison

The very interesting phenomenon that better POS tagging yields worse PCFG-LA parsing suggests that the errors made by the tagger and the parser are very different. To lead to a deep understanding of Chinese POS tagging, we present a detailed comparative analysis of the tagging results generated by our discriminative sequential tagger and the Berkeley parser.

5.5.2.1 Content Words vs. Function Words

There are 49620 words in the development data set. 46491 words are correctly tagged by the first predictor, while 46881 words are correctly tagged by the second predictor. In these words, 45241 words are correctly tagged by both. 1489 words are hard to tag, since both predictors are wrong. Table 5.5 gives a detailed comparison regarding different word types. For each type of word, we report the accuracy of both solvers and

		#Words	Parser	Tagger	Δ	Upper
♠	AD	3448	94.15	94.71	+0.56	97.47
♡	AS	446	98.54	98.44	−0.10	99.11
♡	BA	78	96.15	92.52	−3.63	98.70
♡	CC	720	93.80	90.58	−3.22	97.30
♠	CD	1619	94.66	97.52	+2.86	97.69
	CS	85	91.12	92.12	+1.00	96.51
♡	DEC	1101	85.78	81.22	−4.56	93.71
♡	DEG	1258	88.94	85.96	−2.98	95.33
♡	DER	18	80.95	77.42	−3.53	94.44
♡	DEV	68	84.89	74.78	−10.11	92.54
	DT	640	98.28	98.05	−0.23	99.14
	ETC	142	99.65	100.00	+0.35	100.00
♠	JJ	1363	81.35	84.65	+3.30	89.81
	LB	46	91.30	93.18	+1.88	98.92
	LC	767	96.29	97.08	+0.79	98.38
	M	1340	95.62	96.94	+1.32	97.68
♡	MSP	113	91.30	90.14	−1.16	97.78
♠	NN	14015	93.56	94.95	+1.39	97.23
♠	NR	3445	89.84	95.07	+5.23	97.03
♠	NT	1049	96.70	97.26	+0.56	98.33
♠	OD	145	81.06	86.36	+5.20	87.45
♡	P	1916	96.26	94.56	−1.70	98.07
	PN	653	98.10	98.15	+0.05	99.08
	PU	6593	99.96	99.98	+0.02	99.99
	SB	77	95.36	96.77	+1.41	98.68
	SP	53	61.70	68.89	+7.19	75.56
♠	VA	501	81.27	84.25	+2.98	90.49
♠	VC	501	95.91	97.67	+1.76	98.99
♠	VE	297	97.12	98.48	+1.36	98.99
	VV	7121	91.99	91.87	−0.12	95.98

Table 5.5: Tagging accuracy of words of different classes on the development data.

compare the difference. We also report the upper bound of the two solvers. Column "Δ" shows the difference of the performance of the parser and the tagger. Symbol "+" means the tagger is better than the parser, whilst symbol "−" means the tagger performs worse. The majority of the words that are better labeled by the tagger are content words, including nouns(NN, NR, NT), numbers (CD, OD), predicates (VA, VC, VE), adverbs (AD), nominal modifiers (JJ), and so on. In contrast, most of the words that are better predicted by the parser are function words, including most particles (DEC, DEG, DER, DEV, AS, MSP), prepositions (P, BA) and coordinating conjunction (CC).

5.5.2.2 Open Classes vs. Close Classes

POS can be divided into two broad supercategories: closed class types and open class types. Open classes accept the addition of new morphemes (words), through such processes as compounding, derivation, inflection, coining, and borrowing. On the other hand closed classes are those that have relatively fixed membership. For example, nouns and verbs are open classes because new nouns and verbs are continually coined or borrowed from other languages, while *DEC*/*DEG* are two closed classes because only the function word "的" is assigned to them. The discriminative model can conveniently include many features, especially features related to the word formation, which are important to predict words of open classes.

Table 5.6 summarizes the tagging accuracy relative to IV and OOV words. On the whole, the Berkeley parser processes IV words slightly better than our tagger, but processes OOV words significantly worse. The numbers in this table clearly shows the main weakness of the Berkeley parser is the the predictive power of the OOV words.

	IV	OOV
Tagger	95.22%	81.59%
Parser	95.38%	64.77%

Table 5.6: Tagging accuracy of the IV and OOV words on the development data.

Table 5.7 shows the recall of OOV words on the development data set. Note that only the word types appearing more than 10 times on the development set are reported. From this table, we can also see that new words are hard to be correctly tagged, especially by using a generative model. The Berkeley parser leverage a generative learning model, which is hard to include word formation information. Compared to the Berkeley parser, the shallow tagger performs relatively well on unknown words.

For new *nouns* (NN, NR, NT) and verbs (VV), the different predictive powers are very clearly demonstrated.

	#Words	Parser	Tagger	Δ
AD	21	19.05	33.33	$<$
CD	249	99.20	97.99	$>$
JJ	86	1.16	3.49	$<$
NN	1028	77.82	91.05	$<$
NR	863	51.91	81.69	$<$
NT	25	32.00	60.00	$<$
VA	15	0.00	33.33	$<$
VV	402	59.70	67.66	$<$

Table 5.7: Tagging recall of OOV words (frequency>10) on the development data.

5.5.2.3 Local Disambiguation vs. Global Disambiguation

Closed class words are generally function words that tend to occur frequently and often have structuring uses in grammar. These words have little lexical meaning or have ambiguous meaning, but instead serve to express grammatical relationships with other words within a sentence. They signal the structural relationships that words have to one another and are the glue that holds sentences together. Thus, they serve as important elements to the structures of sentences. The disambiguation of these words normally require more syntactic clues, which is very hard and inappropriate for a sequential tagger to capture. Based on global grammatical inference of the whole sentence, the full parser is relatively good at dealing with structure related ambiguities.

We conclude that discriminative sequential tagging model can better capture local syntactic and morphological information, while the full parser can better capture global syntactic structural information. The discriminative tagging model are limited by the Markov assumption and inadequate to correctly label structure related words. When the tagging and parsing are separated as two individual steps, the tagging errors of the grammatical words will propagated to the full parser and therefore cause many structure parsing errors.

5.5.3 Combination

5.5.3.1 Tagger Ensemble via Bagging

The diversity analysis presented in last section suggests that we may improve parsing by simply combining the tagger and the parser. In Chapter 2, we successfully adapt the general *Bagging* framework to combine the strengths of word-based and character-based word segmentation models. Here, we propose a Bagging model to integrate different POS tagging models. In the training phase, given a training set D of size n, our model generates m new training sets D_i of size $63.2\% \times n$ by sampling examples from D without replacement. Namely no example will be repeated in each D_i. Each D_i is separately used to train a tagger and a parser. Using this strategy, we can get $2m$ *weak* solvers. In the tagging phase, the $2m$ models outputs $2m$ tagging results, each word is assigned one POS label. The final tagging is the voting result of these $2m$ labels. There may be equal number of different tags. In this case, our system prefer the first label they met.

5.5.3.2 Evaluation

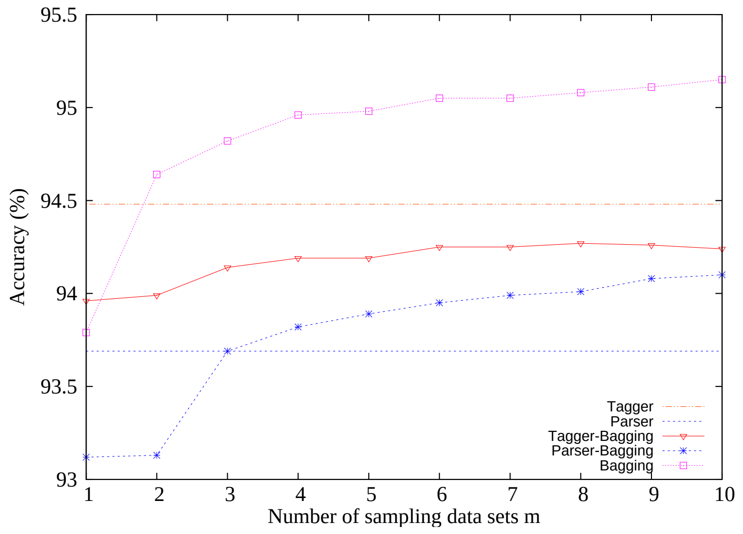


Figure 5.3: Tagging accuracy of Bagging models with different numbers of sampling data sets. *Tagger-Bagging* means that the Bagging system built on the single tagger. *Parser-Bagging* is named in the same way.

We evaluate our combination model on the same data set used above. Figure 5.3 shows the influence of m in the bagging algorithm. Because each new data set D_i in bagging algorithm is generated by a random procedure, the performance of all bagging experiments are not the same. To give a more stable evaluation, we repeat 5 experiments for each m and show the averaged accuracy. We can see that the bagging model taking both sequential tagging and chart parsing models as basic systems outperform the baseline systems and the bagging model taking either model in isolation as basic systems. An interesting phenomenon is that the bagging method can also improve the parsing model, but there is a decrease while only combining taggers.

5.5.3.3 Final Results

Table 5.8 is the final result of the bagging model on the test data set. We can see that Bagging is effective to combine POS taggers designed with different views, yielding a relative error reduction of 12.0%.

Test	System	Accuracy
Baseline	Tagger	94.33%
	Parser	93.50%
Bagging($m = 15$)	Tagger+Parser	95.01%

Table 5.8: Tagging accuracy of different models on the test data (CoNLL setting).

5.6 Comparing and Combining Grammar-free and Grammar-based Parsing Models

5.6.1 Grammar-based Dependency Parsing

Grammar-based approach is based on the finding that projective dependency grammars can be transformed from constituency grammars, such as CFGs. In such approaches, dependency parsing can be resolved by a two-step process: (1) constituent parsing and (2) rule-based extraction of dependencies from phrase structures. The advantage of regarding a dependency grammar as a constituency grammar is that all the well-studied parsing methods for such grammars can be used for dependency parsing as well. Two language-specific properties essentially make grammar-based approaches can be easily applied for Chinese dependency parsing: (1) Chinese is a projective language; (2) Chinese phrase-structures are adequate to be transformed to dependency structures.

CS to DS Conversion In a dependency representation, bi-lexical dependencies are explictly expressed and sometimes classified by functional categories that imply the role the dependent plays with regard to its head. In a constituency representation, the syntactic category of a constituent generally embodies the distributional properties of the constituent. In the absence of dependency and constituency structures for a particular treebank, treebank-guided parser developers normally apply rich linguistic rules to convert one representation formalism to another to get necessary data to train parsers. Xue [2007] examines the linguistic adequacy of dependency structure annotation automatically converted from phrase structure treebanks with rule-based approaches. A structural approach is introduced for the constituency structure (CS) to dependency structure (DS) conversion for the Chinese Treebank data, which is the basis of the CoNLL 2009 shared task data. By applying this conversion procedure on the outputs of an automatic phrase structure parser, we can build a grammar-based dependency parser.

5.6.2 Overall Performance

Table 5.9 shows the overall accuracies of the grammar-free and grammar-based parsers. Roughly speaking, currently state-of-the-art grammar-free parsing achieves slightly better precision than grammar-based parsing with regard to unlabeled dependency prediction. However, the performance of labeled dependency prediction decreases much. We can learn that the CS to DS conversion is not robust to assign functional categories to dependencies and simple linguistic rules are not capable to do fine-grained classification. Nevertheless, previous work shows that the main difficulty in dependency parsing is the prediction of dependency structures, and an extra statistical classifier can be employed to label automatically recognized dependencies with a high accuracy. Since the automatically converted relations of dependencies are not reliable, we mainly focus on the UAS metric in the following experiments.

Devel.	UAS_{dep}	LAS_{dep}	Complete
Mate parser	84.24%	80.55%	30.99%
Berkeley parser+conversion	82.86%	67.44%	27.98%

Table 5.9: Parsing accuracies on the development data.

Object functions in the grammar-free and grammar-based parser are different. The learning of a grammar-free model directly optimizes the LAS, while the grammar-based model directly maximizes a phrase bracketing score which is highly related with the LAS. With regards to the whole unlabeled dependency trees, the grammar-based model performs significantly worse than the grammar-free one. We think one way to improve the grammar-based model for dependency parsing is to change the object function and let the model directly optimize the corresponding dependency structures.

5.6.3 Comparison

5.6.3.1 Relating Parsing Accuracies

Although the overall predictive powers are similar, the two parsers make different distributions of errors. To correlate the parsing performances, we present the scatter plots of the UAS of the grammar-based model against the grammar-free model. In particular, for every sentence, a point of (x, y) is drawn, if the grammar-free model achieves a UAS of x and the grammar-based model achieves a UAS of y. Figure 5.4 shows the relation between their performance. They look like a set of randomly picked points. That means a sentence that can be well analyzed by a grammar-free parser is not necessarily well analyzed by a grammar-based parser, and vice versa.

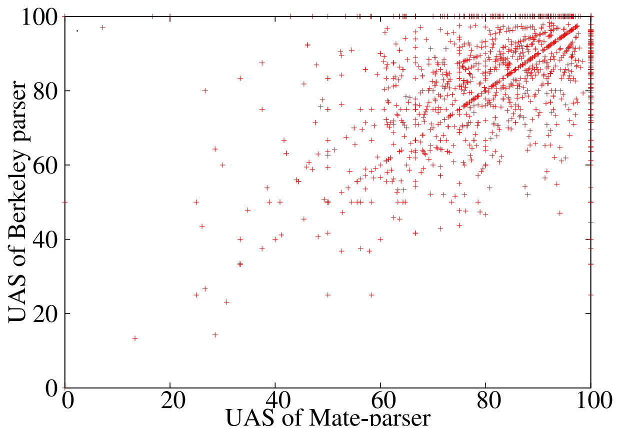


Figure 5.4: Scatter plots of UAS of the grammar-based model against the grammar-free model.

5.6.3.2 Constraints

A grammar-based model utilizes an explicitly defined formal grammar to shape the search space for possible syntactic hypotheses. Parameters of a statistical grammar-based model are related to a grammar rule, and as a result specific language constructions are constrained by each other. For example, parameters are assigned to rewrite rules for a CFG-based model. Since the grammar-based model leverages rewrite rules to locally constrain several possible dependents for one head word, it does relatively better for locally connected dependencies. The traditional evaluation metrics, i.e. UAS and LAS, only consider bi-lexical (first-order) dependencies, which are smallest pieces of a dependency structure. Here we report the prediction accuracy of sibling and grandparent dependencies, i.e. second order dependencies, in Table 5.10. Compared to Table 5.9, we can see that the grammar-based model parses relatively better for slightly larger fragments.

Devel.	F_{sib}	F_{grd}
Mate parser	69.11	81.38
Berkeley parser+conversion	69.07	81.22

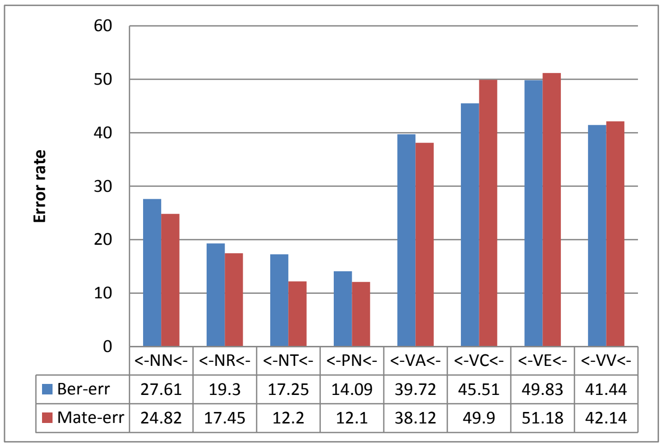
Table 5.10: Different evaluation metrics.

5.6.3.3 Endocentric and Exocentric Constructions

Arguments in exocentric constructions help complete the meaning of a predicate and are taken to be obligatory and selected by their heads; adjuncts in endocentric constructions are structurally dispensable part that provide auxiliary information and taken to be optional and not selected by their heads. An important annotation policy of the CTB is “one grammatical relation per bracket”, which means each constituent falls into one of the three primitive grammatical relations: (1) head-complementation, (2) head-adjunction and (3) coordination. Additionally, the argument is attached at a level that is “closer” to the head than the adjuncts. Due to the linguistic properties of different dependents and the annotation strategies, a grammar-based model can capture more syntactic preference properties of arguments via hard constraints, i.e. grammar rules, and are therefore more suitable to analyze exocentric constructions.

Figure 5.5 is the error rate of unlabeled dependencies considering different construction. A construction “$\leftarrow X \leftarrow$” is considered as correctly predicted if and only if all dependent words and head word of X are completely correctly found. From this figure, we can clearly see that the grammar-free parser does better for the prediction of nominal constructions (NN/NR/NT/PN/VA[1]), which relate more on optional adjuncts or modifiers; the grammar-based parser performs better for the prediction of verbal constructions (VC/VE/VV), which relate more on obligatory arguments. The evaluation of the nominal and verbal constructions roughly confirms the strength of grammar-based model to predict head-argument dependencies.

[1]For the definition and illustration of these tags, please refers to the annotation guidelines (http://www.cis.upenn.edu/~chinese/posguide.3rd.ch.pdf).



	<-NN<-	<-NR<-	<-NT<-	<-PN<-	<-VA<-	<-VC<-	<-VE<-	<-VV<-
🟦 Ber-err	27.61	19.3	17.25	14.09	39.72	45.51	49.83	41.44
🟥 Mate-err	24.82	17.45	12.2	12.1	38.12	49.9	51.18	42.14

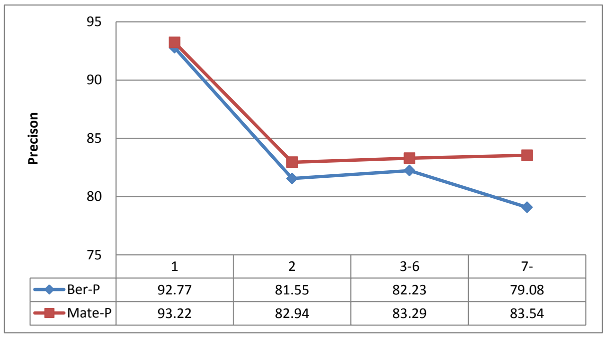
Figure 5.5: Nominal vs. verbal constructions.

5.6.3.4 Factorization

The mainstream approach to statistical natural language parsing factors a syntactic parse into sets of small parts and defines the probability of a parse as accumulation of scores of associated parts. A graph-based dependency parser factorizes a dependency graph as a set of edges (or two connected edges in a second order model). A PCFG-based constituent parser factorizes a phrase-structures as production rules. Roughly speaking, a graph-based model treats long-distance and very local dependencies equally, therefore the prediction accuracy (especially the precision) of long-distance dependencies does not decrease much, as shown in Figure 5.6. On the contrary, the weak, indirect expressive power of long-distance, non-local dependencies produce serious difficulties for the PCFG-based parser.

5.6.4 Combination

The comparative analysis highlights the fundamental diversity between grammar-free and grammar-based models and their complementary parsing strengths, which suggests that there is still space for improvement, just by combining the two existing models. The upper bound of the UAS to combine the two parsers is **91.90%**, which motivates us to address the problem of parser ensemble. Combining the outputs of several systems has been shown in the past to improve parsing performance significantly. Several ensemble models have been proposed for the parsing of syntactic



	1	2	3-6	7-
Ber-P	92.77	81.55	82.23	79.08
Mate-P	93.22	82.94	83.29	83.54

Figure 5.6: Parsing precision relative to dependency length.

constituents and dependencies, including learning-based stacking [Nivre and McDonald, 2008; Torres Martins et al., 2008], learning-free post-inference [Henderson and Brill, 1999; Sagae and Lavie, 2006b] and learning-based post-inference [Zhang et al., 2009]. For the former two approaches, Surdeanu and Manning [2010] present a systematic analysis and comparison. To enhance state-of-the-art Chinese dependency parsing models, we first implement and evaluate a previously introduced stacking model as well as a re-parsing model, and then propose a new model to better integrate heterogeneous parsers.

5.6.4.1 Parser Ensemble via Stacking

Stacking is a simple mechanism for solving parser combination in the discriminative parsing framework. All one needs to do is to extract a set of guided features from the grammar-based parser and include these features into the grammar-free model. For dependency parsing, this amounts to including features indicating whether another parser believed a certain dependency or pair of dependencies actually exist in the tree. McDonald [2006] introduces such a stacking model for English parsing. Specifically, his method adds two auxiliary features. The first is a simple binary feature indicating for each edge (or pair of edges), whether or not the auxiliary parser believes this edge to be part of the correct tree. The second feature is identical to the first, except that it is combined with the POS tags of the head and modifier in the edge. In this paper, we implement exactly the same system for Chinese parsing.

The grammar-based parser can be directly trained on the original training data.

If we directly apply this parser to extend the training data to generate samples for a stacked parser, the new training data will be very different from the data in the run time, resulting in poor performance. One way to alleviate this training/test mismatch problem is to use the stacking method, where a K-fold *cross-validation* on the original data is performed to construct the training data for the new grammar-free parser. In the following experiments, we apply a 5-fold cross validation to make the training data *dirty*.

5.6.4.2 Parser Ensemble via Re-parsing

Sagae and Lavie [2006b] present a framework for combining the output of several different accurate parsers to produce results that are superior to each of the individual parsers. Once we have obtained the two structures respectively from the grammar-free and grammar-based parsers, we can build a graph where each word in the sentence is a node. We then create weighted directed edges between the nodes corresponding to words for which dependencies are obtained from each of the initial structures. Once this graph is created, the sentence can be re-parsed by a graph-based dependency parsing algorithm such as our choice, Eisner's algorithm [Eisner, 1996].

5.6.4.3 Parser Ensemble via Bagging

The dependency parsing problem can be viewed as a word prediction problem, i.e. finding head of each word. It is convenient to transform dependency parser ensemble to a word voting problem, and the Bagging method is therefore easy to apply. In the training phase, given a training set D of size n, our model generates m new training sets D_i of size $61.8\% \times n$ by sampling examples from D without replacement. Each D_i is separately used to train the Berkeley parser and the graph-based dependency parser. Using this strategy, we can get $2m$ *weak* parsers. In the parsing phase, the $2m$ models outputesults (which are automatically converted to dependency parses) for each given sentence. For every sentence, the final parsing result is a combination of its corresponding $2m$ structures. We implement two strategies for the combination.

Word-by-word voting These $2m$ dependency trees can be combined in a simple word-by-word voting scheme, where each parser votes for the head of each word in the given sentences, and the head with most votes is assigned to each word. This very simple scheme guarantees that final set of dependencies will have as many votes as possible, but it does not guarantee that the final voted set of dependencies will be a well formed dependency tree.

Re-parsing To guarantee that the resulting dependency tree is well-formed, we employ the dynamic programming algorithm of [Eisner, 1996] for re-parsing.

5.6.4.4 Evaluation

Table 5.11 shows the parsing performance on the development data of the stacking model. Compared to the baseline results (see Table 5.9 and 5.10), we can see that the stacking model is effective to improve the parsing accuracy, with regards to both first-order and second-order dependencies.

Devel.	UAS_{dep}	F_{sib}	F_{grd}
Stacking	85.41%	71.36	82.94

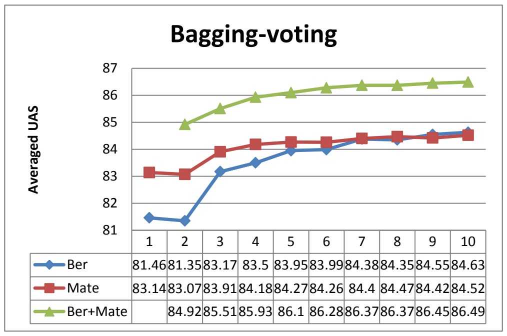
Table 5.11: Performance of the stacking model.

Table 5.12 is the re-parsing performance on the development data. When only two baseline parsers are applied to provide dependency candidates, the re-parsing method does not work well. The parsing accuracy slightly decreases, even compared to the weaker baseline performance. When the above stacking parser is also employed, the re-parsing method performs a little better than the best of the three. However, the improvement is too modest.

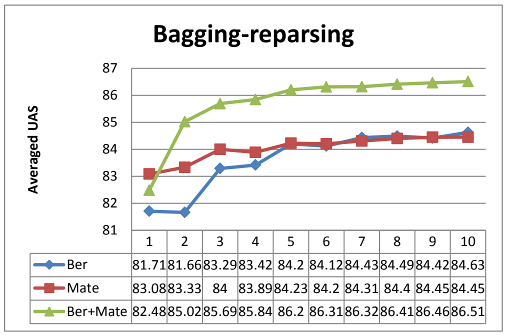
Devel.	UAS_{dep}
mate/Berkeley+conversion	83.72%
mate/Berkeley+conversion/stacking	85.60%

Table 5.12: Performance of the re-parsing model.

We evaluate our Bagging model on the same data set. In the following experiments, we use the standard discriminative POS tagger to provide inputs for the dependency parser. Because each new data set D_i in the Bagging algorithm is generated by a random procedure, the performance of all Bagging experiments are not the same. To give a more stable evaluation, we repeat 3 experiments for each m and show the averaged accuracy. Figure 5.7 shows the influence of m in the Bagging algorithm. We can see that the Bagging model taking both discriminative dependency models and generative constituency models as basic systems outperform the baseline systems and the Bagging model taking either model in isolation as basic systems. The Bagging method can also improve individual parsing models, and the grammar-based model can be enhanced more.



	1	2	3	4	5	6	7	8	9	10
Ber	81.46	81.35	83.17	83.5	83.95	83.99	84.38	84.35	84.55	84.63
Mate	83.14	83.07	83.91	84.18	84.27	84.26	84.4	84.47	84.42	84.52
Ber+Mate		84.92	85.51	85.93	86.1	86.28	86.37	86.37	86.45	86.49



	1	2	3	4	5	6	7	8	9	10
Ber	81.71	81.66	83.29	83.42	84.2	84.12	84.43	84.49	84.42	84.63
Mate	83.08	83.33	84	83.89	84.23	84.2	84.31	84.4	84.45	84.45
Ber+Mate	82.48	85.02	85.69	85.84	86.2	86.31	86.32	86.41	86.46	86.51

Figure 5.7: Dependency UAS of Bagging models with different numbers of sampling data sets.

Bagging a single-view parser Figure 5.7 indicates that (1) the Bagging method can also improve individual single-view parsers, especially for the grammar-based parser and (2) the Bagging approach to improve either grammar-free or grammar-based parser obtains equivalent overall accuracy. We analyze the outputs generated by two single-view Bagging models and present three aforementioned evaluation metrics in Table 5.13. The ensemble learning enhanced parsing models still have complementary strengths and the combination of both is therefore beneficial.

Devel.	Complete	F_{sib}	F_{grd}
Berkeley+conversion	30.08%	71.06	82.95
Mate	31.50%	69.55	80.48
Berkeley+conversion/Mate	34.39%	72.67	82.92

Table 5.13: Performance of different Bagging models. m=10, Inference=re-parsing.

5.6.4.5 Final Results

Table 5.14 summarizes the final results of different models on the test data set. We can see that parser ensemble is very important to advance the state-of-the-art of Chinese dependency parsing. Hatori et al. [2011] study several enhancement techniques, including joint learning, dynamic programming and deep feature engineering, for transition-based dependency parsing. Evaluations on an out-of-date version of CTB show that their system achieves significantly better performance than previously reported systems. We re-train their system on the CoNLL data, and report its UAS in the first line.[1] The beam width for decoding is set to 32, and the iteration

[1]We would like to thank Jun Hatori for sharing his implementation.

number (29) for model training is tuned on the development data. Though several second order graph-based parsers have been implemented and evaluated for Chinese dependency parsing, they do not focus on POS tagging much. In our experiments, the Mate parser based a much stronger POS tagger easily defeats many other systems, and obtain a state-of-the-art result.

Test	UAS
State-of-the-art [Hatori et al., 2011]	84.27%
Mate parser	84.38%
Berkeley parser+conversion	83.49%
Stacking	85.80%
Bagging($m = 20$)(voting)	86.85%
Bagging($m = 20$)(re-parsing)	86.79%

Table 5.14: Accuracies of different models on the test data.

System ensemble can significantly enhance state-of-the-art parsers for Chinese. Compared to the previously introduced stacking model, our Bagging model is more effective to integrate grammar-free and grammar-based parsers. Based on automatic POS tagging, our Bagging model achieves a UAS of 86.85%, which obtains relative error reductions of 16% and 20% respectively compared to the strong baselines. The remarkable results of parsing ensemble also demonstrate the diversity of constituency and dependency parsing.

5.7 Discussion

Chinese POS tagging has been proven much more challenging due to many language-specific properties. From a linguistic point of view, meaning arises from the differences between linguistic units, including words, phrases and so on, and these differences are of two kinds: paradigmatic and syntagmatic. Both paradigmatic and syntagmatic lexical relations have a great impact on POS tagging, because the *value* of a word is determined by the two relations. We hold a view of structuralist linguistics and study the impact of syntagmatic lexical relations on Chinese POS tagging. In particular, we comparatively analyze syntax-free and syntax-based models and employ a Bagging model to integrate a sequential tagger and a chart parser to capture syntagmatic relations that have a great impact on non-local disambiguation. In Chapter 7, we will harvest word partition information from large-scale raw texts to capture paradigmatic lexical relations to enhance a tagger.

The information encoded in a dependency representation is different from the information captured in a constituency representation. While the dependency structure represents head-dependent relations between words, the constituency structure represents the grouping of words into phrases, classified by structural categories such as noun phrase and verb phrase. These differences concern what is explicitly encoded in the respective representations, and affect grammar-free and grammar-based dependency parsing models much. Our analysis highlights the fundamental diversity between grammar-free and grammar-based models, which are either based on a particular view of syntactic processing. On one hand, each view alone can yield a reasonably good predictor, but is inadequate to interpret every linguistic phenomenon. On the other hand, some linguistic properties that are not captured by one model, can be potentially captured by other models. Many tasks can take advantages of complementary strengths of the heterogeneous views. For example, co-training style semi-supervised learning can take advantage from multiple views to explore unlabeled data; co-testing style active learning can benefit from multiple views to efficiently build hand-crafted corpora.

Our evaluation results of two parser ensemble methods confirms the importance of leveraging both constituency and dependency structures for Chinese syntactic processing. It is also worth noting that many syntactic theories make use of hybrid representations, combining elements of dependency structure with elements of phrase structure. For example, in a lexicalized tree-adjoining grammar representation, both substitution and adjunction operations build dependencies between anchor words in elementary trees, and the derived trees represent hierarchical phrase structures. Such "deep" and "rich" formalisms seems more attractive for parsing Chinese texts.

Chapter 6

Parse Reranking with Homogeneous and Heterogeneous Annotations

Discriminative parse reranking has been shown to be an effective technique to improve the generative parsing models. In this chapter, we present a series of experiments on parsing the Tsinghua Chinese Treebank (TCT) with PCFG-LA grammars and subsequent reranking with a perceptron-based discriminative model. We are also interested in exploiting heterogeneous treebanks in the discriminative reranking framework. In addition to the homogeneous annotation on TCT, we incorporate the Penn treebank based parsing result as heterogeneous annotation into the reranking feature model. The reranking model achieved 1.12% absolute improvement on F-score over the Berkeley parser on a development set.

This chapter is joint work with Yi Zhang and Rui Wang, originally published in [Sun et al., 2010].

6.1 Motivation

The data-driven approach to syntactic analysis of natural language has undergone revolutionary development in the last 15 years, ever since the first few large scale syntactically annotated corpora, i.e. treebanks, became publicly available in the mid-90s of the last century. One and a half decades later, treebanks remain to be an expensive type of language resources and only available for a small number of languages. While traditional linguistic studies typically focus on either isolated language phenomena or limited interaction among a small group of phenomena, the annotation scheme in a treebanking project requires full coverage of language use in the source media, and proper treatment with a uniform annotation format. Such high demand from the practical application of linguistic theory has given rise to a countless number of attempts and variations in the formalization frameworks. While the harsh natural selection set the bar high and many attempts failed to even reach the actual annotation phase, a handful highly competent grammar frameworks have given birth to several large scale treebanks.

The co-existence of multiple treebanks with heterogeneous annotation presents a new challenge to the consumers of such resources. The immediately relevant task is the automated syntactic analysis, or parsing. While many state-of-the-art statistical parsing systems are not bound to a specific treebank annotation (assuming the formalism is predetermined independently), almost all of them assume homogeneous annotation in the training corpus. Therefore, such treebanks can not be simply put together when training the parser. One approach would be to convert them into a uniform representation, although such conversion is usually difficult and by its nature an error-prone process. The differences in annotations constitute different generative stories: i.e., when the parsing models are viewed as mechanisms to produce structured sentences, each treebank model will associate its own structure with the surface string independently. On the other hand, if the discriminative view is adopted, it is possible to use annotations in different treebanks as indication of goodness of the tree in the original annotation.

The type of treebank annotations affects the performance of the parsing models. Taking the Penn Chinese Treebank (PCTB; Xue et al. [2005]) and Tsinghua Chinese Treebank (TCT; Zhou [2004]) as examples, PCTB is annotated with a much more detailed set of phrase categories, while TCT uses a more fine-grained POS tagset. The asymmetry in the annotation information is partially due to the difference of linguistic treatment. But more importantly, it shows that both treebanks have the potential of being refined with more detailed classification, on either phrasal or word categories. One data-driven approach to derive more fine-grained annotation is the hierarchically split-merge parsing [Petrov et al., 2006; Petrov and Klein, 2007], which induces subcategories from coarse-grained annotations through an expectation maximization procedure. In combination with the coarse-to-fine parsing strategy, efficient inference can be done with a cascade of grammars of different granularity. Such parsing models have reached (close to) state-of-the-art performance for many languages including Chinese and English.

Another effective technique to improve parsing results is discriminative reranking [Charniak and Johnson, 2005; Collins and Koo, 2005]. While the generative models compose candidate parse trees, a discriminative reranker reorders the list of candidates in favor of those trees which maximizes the properties of being a good analysis. Such extra model refines the original scores assigned by the generative model by focusing its decisions on the fine details among already "good" candidates. Due to this nature, the set of features in the reranker focus on those global (and potentially long distance) properties which are difficult to model with the generative model. Also, since it is not necessary for the reranker to generate the candidate trees, one can easily integrate additional external information to help adjust the ranking of the analysis. In the chapter, we will describe a reranking model we developed for Chinese parsing. We will also show how the heterogeneous parsing results can be integrated through the reranker to further improve the performance of the system.

6.2 Comparison of Two Chinese Treebanks

In this chapter, we focus on two popular Chinese treebanks: (1) the Penn Chinese Treebank (PCTB) and (2) the Tsinghua Chinese Treebank (TCT). They are both segmented, part-of-speech tagged, and fully bracketed corpora in the constituency formalism. However, the design of PCTB differs much from TCT. Whereas PCTB draws primarily on Government-Binding (GB) theory from 1980s, the TCT annotation strongly reflects early descriptive linguistics. We list several important differences between them as follows.

- The sources of PCTB are mostly newswires, while the dataset of TCT is a mixture of several genres, including newspaper texts, encyclopedic texts and novel texts.
- TCT and PCTB have different word segmentation standards.
- TCT is somehow branching-rich annotation, while PCTB annotation is category-rich. Specifically the topological tree structures are more detailed in TCT, and there are not many flat structures. However, constituents are not detailed classified, namely the number of phrasal categories is small. On the contrary, though flat structures are very common in PCTB, the categorization of phrases is fine-grained.
- PCTB contains functional information. Function tags appended to constituent

labels are used to indicate additional syntactic or semantic information. For example, the label *SBJ* is used to mark the surface subject.

- TCT contains head indices, which explicitly indicates the head components of each constituent.
- Following the GB theory, PCTB assume there are *movements*, so there are empty category annotation. Because of different theoretical foundations, there are different explanations for a series of linguistic phenomena such as the usage of the function word "的".

6.3 A Hybrid Parsing System

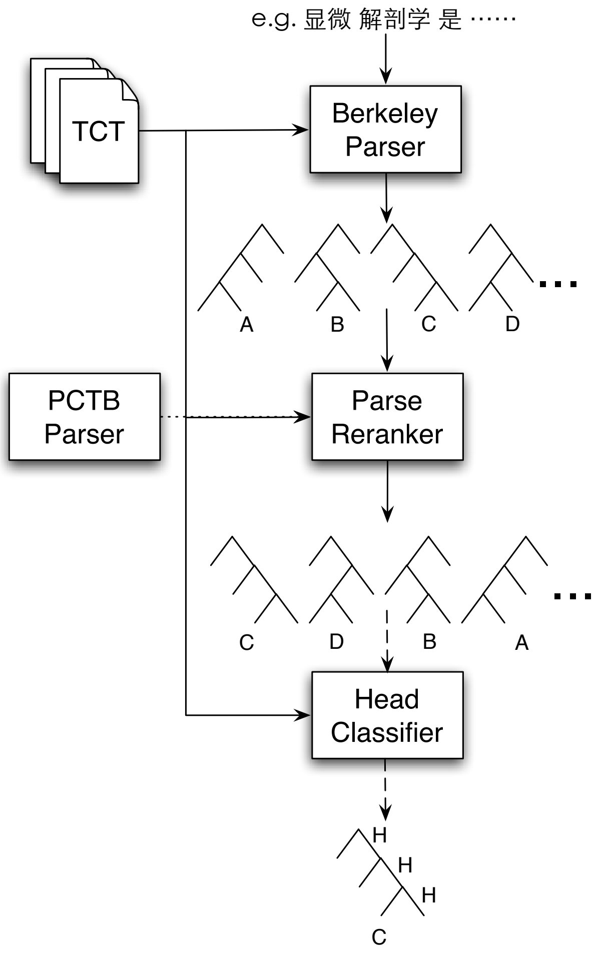


Figure 6.1: Workflow of the System

```
   input : Data $\{(\mathbf{x}_t, y_t), t = 1, 2, ..., m\}$
1  Initialize: $\mathbf{w} \leftarrow (0, ..., 0)$
2  for $i = 1, 2, ..., I$ do
3      for $t =$ SHUFFLE $(1, ..., m)$ do
4          $y_t^* = \arg\max_{y \in \mathrm{GEN}_{\mathrm{n}}^{\mathrm{best}}(\mathbf{x}_t)} \mathbf{w}^\top \Phi(\mathbf{x}_t, y)$
5          if $y_t^* \neq y_t$ then
6              $\mathbf{w} \leftarrow \mathbf{w} + (\Phi(\mathbf{x}_t, y_t) - \Phi(\mathbf{x}_t, y_t^*))$
7          end
8      end
9      $\mathbf{w}_i \leftarrow \mathbf{w}$
10 end
11 return $\mathbf{aw} = \frac{1}{I}\sum_{i=1}^{I} \mathbf{w}_i$
```

Algorithm 3: The *Perceptron* learning procedure.

6.3.1 System Architecture

In this section, we will present our approach in detail. The whole system consists of three main components, the Berkeley Parser, the Parse Reranker, and the Head Classifier. The workflow is shown in Figure 6.1. Firstly, we use the Berkeley Parser trained on the TCT to parse the input sentence and obtain a list of possible parses; then, all the parses[1] will be re-ranked by the Parse Reranker; and finally, the Head Classifer will annotate the head information for each constituent on the best parse tree. For parse reranking, we can extract features either from TCT-style parses or together with the PCTB-style parse of the same sentence. For example, we can check whether the boundary predictions given by the TCT parser are agreed by the PCTB parser. Since the PCTB parser is trained on a different treebank from TCT, our reranking model can be seen as a method to use a heterogenous resource.

6.3.2 Parse Reranking

6.3.2.1 Parameter Estimation

We follow Collins and Koo [2005]'s discriminative reranking model to score possible parse trees of each sentence given by the Berkeley Parser.

Previous research on English shows that structured perceptron [Collins, 2002] is one of the strongest machine learning algorithms for parse reranking [Collins and Duffy, 2002; Gao et al., 2007]. In our system, we use the averaged perceptron algo-

[1]In practice, we only take the top n parses. We have different n values in the experiment settings, and n is up to 50.

rithm to do parameter estimation. Algorithm 3 illustrates the learning procedure. The parameter vector $\mathbf{w}$ is initialized to $(0, ..., 0)$. The learner processes all the instances (t is from 1 to n) in each iteration (i). If current hypothesis ($\mathbf{w}$) fails to predict $\mathbf{x}_t$, the learner update $\mathbf{w}$ through calculating the difference between $\Phi(\mathbf{x}_t, y_t^*)$ and $\Phi(\mathbf{x}_t, y_t)$. At the end of each iteration, the learner save the current model as $\mathbf{w} + i$, and finally all these models will be added up to get $\mathbf{aw}$.

6.3.2.2 Features

We use an example to show the features we extract in Figure 6.2.

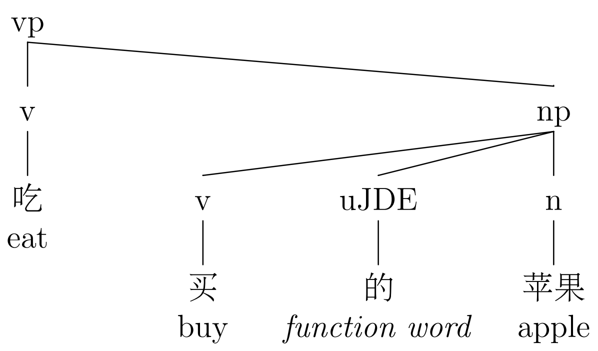


Figure 6.2: An example for interpretation of features: *To eat apples that are bought.*

- **Rules**: The context-free rule itself: np $\rightarrow$ v + uJDE + np.
- **Grandparent rules**: Same as the Rules, but also including the nonterminal above the rule: vp(np $\rightarrow$ v + uJDE + np)
- **Bigrams**: Pairs of nonterminals from the left to right of the the rule. The example rule would contribute the bigrams np(STOP, v), np(v, uJDE), np(uJDE, np) and np(np, STOP).
- **Grandparent bigrams**: Same as Bigrams, but also including the nonterminal above the bigrams. For instance, vp(np(STOP, v))
- **Lexical bigrams**: Same as Bigrams, but with the lexical heads of the two nonterminals also included. For instance, np(STOP,买).
- **Trigrams**: All trigrams within the rule. The example rule would contribute the trigrams np(STOP, STOP, v), np(STOP, v, uJDE), np(v, uJDE, np), np(uJDE, np, STOP) and np(np, STOP, STOP).

- **Combination of boundary words and rules**: The first word and the rule (i.e. 买+(np $\rightarrow$ v + uJDE + np)), the last word and the rule one word before and the rule, one word after and the rule, the first word, the last word and the rule, and the first word's POS, last word's POS and the rule.

- **Combination of boundary words and phrasal category**: Same as combination of boundary words and rules, but substitute the rule with the category of current phrases.

- **Two level rules**: Same as Rules, but also including the entire rule above the rule: vp $\rightarrow$ v + (np $\rightarrow$ v + uJDE + np)

- **Original rank**: The logarithm of the original rank of n-best candidates.

- **Affixation features**: In order to better handle unknown words, we also extract morphological features: character n-gram prefixes and suffixes for n up to 3. For example, for word/tag pair 自然环境/n, we add the following features: (prefix1,自,n), (prefix2,自然,n), (prefix3,自然环,n), (suffix1,境,n), (suffix2,环境,n), (suffix3,然环境,n).

Apart from training the reranking model using the same data set (i.e. the TCT), we can also use another treebank (e.g. the PCTB). Although they have quite different annotations as well as the data source, it would still be interesting to see whether a heterogenous resource is helpful with the parse reranking.

- **Consistent category**: If a phrase is also analyzed as one phrase by the PCTB parser, both the TCT and PCTB categories are used as two individual features. The combination of the two categories are also used.

- **Inconsistent Category**: If a phrase is not analyzed as one phrase by the PCTB parser, the TCT category is used as a feature.

- **Number of consistent and inconsistent phrases**: The two number are used as two individual featuers. We also use the ratio of the number of consistent phrases and inconsistent phrase (we add 0.1 to each number for smoothing), the ratio of the number of consistent/inconsistent phrases and the length of the current sentence.

- **POS Tags**: For each word, the combination of TCT and PCTB POS tags (with or without word content) are used.

6.3.3 Head Classifier

Following [Song and Kit, 2009], we apply a sequence tagging method to find head constituents. We suggest readers to refer to the original paper for details of the method. However, since the feature set is different, we give the description of them in this paper. To predict whether current phrase is a head phrase of its parent, we use the same example above (Figure 6.2) for convenience. If we consider np as our current phrase, the following features are extracted,

- **Rules**: The generative rule, vp $\rightarrow$ v + (np).
- **Category of the Current Phrase and its Parent**: np, vp, and (np, vp).
- **Bigrams and Trigrams**: (v, np), (np, STOP), (STOP, v, np), and (np, STOP, STOP).
- **Parent Bigrams and Trigrams**: vp(v, np), vp(np, STOP), vp(STOP, v, np), vp(np, STOP, STOP).
- **Lexical Unigram**: The first word 买, the last word 苹果, and together with the parent, (vp,买) and (vp,苹果)

6.4 Experiments

6.4.1 Setting

The data set used in the CIPS-ParsEval-2010 evaluation is converted from the Tsinghua Chinese Treebank (TCT). There are two subtasks: (1) event description subsentence analysis and (2) complete sentence parsing. On the assumption that the boundaries and relations between these event description units are determined separately, the first task aims to identify the local fine-grained syntactic structures. The goal of the second task is to evaluate the performance of the automatic parsers on complete sentences in real texts. The training data set is a mixture of several genres, including newspaper texts, encyclopedic texts and novel texts.

In order to gain a representative set of training data, we use cross-validation scheme described in [Collins, 2000]. The data set is a mixture of three genres. We equally split every genre data into 10 subsets, and collect three subset of different genres as one fold of the whole data. In this way, we can divide the whole data into 10 balanced subsets. For each fold data, a complement parser is trained using all other data to produce multiple hypotheses for each sentence. This cross-validation scheme can prevent the initial model from being unrealistically "good" on the training sentences. We use the first 9 folds as training data and the last fold as development data for the following experiments. For the final submission of the evaluation task, we re-train a reranking model using all 10 folds data. All reranking models are trained with 30 iterations.

For parsing experiments, we use the Berkeley parser[1]. All parsers are trained with 5 iterations of split, merge, smooth. To produce PCTB-style analysis, we train the Berkeley parse with PCTB 5.0 data that contains 18804 sentences and 508764 words. For the evaluation of development experiments, we used the EVALB tool[2] for evaluation, and used labeled recall (R), labeled precision (P) and F-score score (which is the harmonic mean of R and P) to measure accuracy.

For the head classification, we use SVM-HMM[3], an implementation of structural SVMs for sequence tagging. The main setting of learning parameter is C that trades off margin size and training error. In our experiments, the head classification is not sensitive to this parameter and we set it to 1 for all experiments reported. For the kernel function setting, we use the simplest linear kernel.

6.4.2 Upper Bound of Reranking

n	1	2	5	10	20	30	40	50
F-score	79.97	81.62	83.51	84.63	85.59	86.07	86.38	86.60

Table 6.1: Upper bound of f-score as a function of number n of n-best parses.

The upper bound of n-best parse reranking is shown in Table 6.1. From the 1-best result we see that the base accuracy of the parser is 79.97. 2-best and 10-best show promising oracle-rate improvements. After that things start to slow down, and we achieve an oracle rate of 86.60 at 50-best.

6.4.3 Reranking Using Homogeneous Annotations

Table 6.2 summarizes the performance of the basic reranking model. It is evaluated on short sentences (less than 40 words) from the development data of the task 2. When 40 reranking candidates are used, the model gives a 0.76% absolute improvement over the basic Berkeley parser.

[1]http://code.google.com/p/berkeleyparser/
[2]http://nlp.cs.nyu.edu/evalb/
[3]http://www.cs.cornell.edu/People/tj/svm_light/svm_hmm.html

	POS(%)	P(%)	R(%)	F
Baseline	93.59	85.60	85.36	85.48
$n = 2$	93.66	85.84	85.54	85.69
$n = 5$	93.62	86.04	85.73	85.88
$n = 10$	93.66	86.22	85.85	86.04
$n = 20$	93.70	86.19	85.87	86.03
$n = 30$	93.70	86.32	86.00	86.16
$n = 40$	93.76	86.40	86.09	86.24
$n = 50$	93.73	86.10	85.81	85.96

Table 6.2: Reranking performance with different number of parse candidates on the sentences that contain no more than 40 words in the development data.

6.4.4 Reranking Using Heterogeneous Annotations

Table 6.3 summarizes the reranking performance using PCTB data. It is also evaluated on short sentences of the task 2. When 30 reranking candidates are used, the model gives a 1.12% absolute improvement over the Berkeley parser. Comparison of Table 6.2 and 6.3 shows an improvement by using heterogeneous data.

	POS(%)	P(%)	R(%)	F
$n = 2$	93.70	85.98	85.67	85.82
$n = 5$	93.75	86.52	86.19	86.35
$n = 10$	93.77	86.64	86.29	86.47
$n = 20$	93.79	86.71	86.34	86.53
$n = 30$	93.80	86.72	86.48	86.60
$n = 40$	93.80	86.54	86.22	86.38
$n = 50$	93.89	86.73	86.41	86.57

Table 6.3: Reranking performance with different number of parse candidates on the sentences that contain no more than 40 words in the development data.

6.4.5 Head Classification

The head classification performance is evaluated using gold-standard syntactic trees. For each constituent in a gold parse tree, a structured classifier is trained to predict whether it is a head constituent of its parent. Table 6.4 shows the overall performance of head classification. We can see that the head classification can achieve a high performance.

P(%)	R(%)	$F_{\beta=1}$
98.59%	98.20%	98.39

Table 6.4: Head classification performance with gold trees on the development data.

6.5 Conclusion

In this chapter, we introduced a hybrid system for Chinese parsing. The generative coarse-to-fine parsing model is integrated with a discriminative parse reranking model, as well as a head classifier based on sequence labeling. We use the perceptron algorithm to train the reranking models and the experimental results showed improvements over the baseline. In addition, we exploit a heterogeneous treebank to improve parsing. In particular, features extracted from heterogeneous structures are incorporated into the parse reranker. Our method for annotation adaptation can be viewed as a generalized *stacking* method which relies on the ability of discriminative learning to explore informative heterogeneous annotation features. Experimental results show that the heterogeneous annotations can further enhance reranking models.

Chapter 7

Enriching Lexical Representation for Syntactic Parsing

This chapter investigates improving supervised parsing with unsupervised language acquisition. In particular, we focus on the problem of lexical representation in POS tagging, text chunking and dependency parsing, introducing new features that incorporate word clusters derived from a large-scale unlabeled corpus. We demonstrate the importance of rich lexical information in a series of parsing experiments on the Penn Chinese Treebank and Chinese Gigaword, and we show that the cluster-based features yield substantial gains in performance across a wide range of conditions.

Parts of this chapter are originally published in [Sun and Uszkoreit, 2012].

7.1 Motivation

Meaning arises from the differences between linguistic units, including words, phrases and so on, and these differences are of two kinds: syntagmatic (concerning positioning) and paradigmatic (concerning substitution). The distinction is a key one in structuralist semiotic analysis. Whilst syntagmatic relations are possibilities of combination, paradigmatic relations are functional contrasts - they involve differentiation. Generally speaking, syntagmatic relations refer intratextually to other linguistic units co-occurring within the text, while paradigmatic relations refer intertextually to linguistic units which are absent from the text.

The syntactic structures of given sentences represent the syntagmatic relations of words contained in these sentences. In Chapter 5 and 6, we introduced supervised parsing methods to directly capture the syntagmatic relations among words of a given sentences. Conventionally, a majority of parsing systems (as we have shown in Chapter 5) leverage words themselves as important features for disambiguation. There are two main problems for this naive word representation: (1) Word form is not sufficient to represent a word; (2) Word form suffers from data sparsity.

The *value* of a word is determined by both its paradigmatic and its syntagmatic relations. For example, the CTB-style POS tags capture both paradigmatic and syntagmatic relations among words, since its annotation criterion is the syntactic distribution of words. In this chapter, we are concerned with capturing paradigmatic relations among words to enhance syntactic processing. In particular, we are interested in incorporating rich lexical information into supervised parsers. The common way to build wide-coverage lexical resources is to perform unsupervised algorithms to acquire rich word representations, such as word clustering, word similarity calculating. In this chapter, we leverage unsupervised word clustering to explore useful paradigmatic relations encoded in large-scale unlabeled data. Similar to our study for word segmentation presented in Chapter 4, the work introduced in this chapter is another successful example to leverage feature induction to bridge the gap between supervised language processing and unsupervised language acquisition.

7.2 Word Clustering

Word clustering is a technique for partitioning sets of words into subsets of syntactically or semantically similar words. It is a very useful technique to capture paradigmatic or substitutional similarity among words. For example, word classes are often used in language modeling to solve the problem of sparse data. Various clustering techniques have been proposed, some of which, for example, perform automatic word clustering optimizing a maximum-likelihood criterion with iterative clustering algorithms. The main problem is that we cannot expect these independently optimized classes to be correspondent with syntactic structures. In the feature induction framework, this problem is partially resolved by exploring the ability of discriminative learning to automatically identify the correspondence between the two types of "word classes".

Distributional word clustering is based on the assumption that words that appear in similar contexts tend to have similar meanings. In the literature, contexts have been defined as subjective and objective relations involving the word, as the documents containing the word, or as search engine snippets for the word as a query. In this section, we derive new features for POS tagging by applying two distributional clustering methods, which both take into account surrounding words as contexts. They have been successfully applied to many NLP problems, such as machine translation [Och, 1999].

7.2.1 Brown Clustering

Our first choice is the bottom-up agglomerative word clustering algorithm of [Brown et al., 1992] which derives a hierarchical clustering of words from unlabeled data. This algorithm generates a hard clustering – each word belongs to exactly one cluster. The input to the algorithm is sequences of words $w_1, ..., w_n$. Initially, the algorithm starts with each word in its own cluster. As long as there are at least two clusters left, the algorithm merges the two clusters that maximizes the quality of the resulting clustering. The quality is defined based on a class-based bigram language model as follows.

$$P(w_i|w_1, ...w_{i-1}) \approx p(C(w_i)|C(w_{i-1}))p(w_i|C(w_i))$$

where the function C maps a word w to its class $C(w)$. We use a publicly available package[1] [Liang et al., 2005] to train this model.

7.2.2 MKCLS Clustering

We also do experiments by using another popular clustering method based on the exchange algorithm [Kneser and Ney, 1993]. The objective function is maximizing the likelihood $\prod_{i=1}^{n} P(w_i|w_1, ..., w_{i-1})$ of the training data given a partially class-based bigram model of the form

$$P(w_i|w_1, ...w_{i-1}) \approx p(C(w_i)|w_{i-1})p(w_i|C(w_i))$$

We use the publicly available implementation MKCLS[2] [Och, 1999] to train this model.

One downside of both Brown and MKCLS clustering is that it is based solely on bigram statistics, and does not consider word usage in a wider context. We choose to work with these two algorithms due to their prior success in other NLP applications [Koo et al., 2008; Miller et al., 2004]. However, we expect that our approach can function with other clustering algorithms.

[1] http://cs.stanford.edu/~pliang/software/brown-cluster-1.2.zip
[2] http://code.google.com/p/giza-pp/

7.3 Experiments in POS Tagging

7.3.1 Cluster-based Features

In this part, we consider using unlabeled data to improve our supervised Chinese POS tagger introduced in Chapter 5. In the spirit of [Miller et al., 2004], our basic strategy for taking advantage of unlabeled data is to derive information from unlabeled data and use it in a supervised model. Our approach is detailed as follows: In a preprocessing step, we use automatically segmented text to cluster words. The output of this step is then used as features in a discriminative learning model. We are relying on the ability of discriminative learning method to explore informative features, which play central role to boost the tagging performance.

Key to the success of our approach is the use of word clusters to assist the POS tagger. Word clusters are used as substitutes for word forms. Following the denotation in Chapter 5.2, we denote a word in focus with a fixed window $w_{-1}ww_{+1}$, where w is current token. The clustering-based features includes:

- Unigram cluster feature: w_{-1}, w, w_{+1};
- Bigram cluster feature: w_{-1}_w, w_w_{+1}.

That means 5 new features are added.

7.3.2 Experiments and Analysis

7.3.2.1 Setting

We conduct experiments using CTB 6.0 and define the training, development and test sets according to the Chinese sub-task of the CoNLL 2009 shared task. The large-scale unlabeled data for the POS tagging experiments comes from the Mandarin news text of the Chinese Gigaword. Word segmentation is a necessary pre-processing for lexical acquisition. Here we use our semi-supervised character-based segmenter introduced in Chapter 4. To obtain word clusters, we use the open source packages mentioned above. To solve the sequence labeling problem in POS tagging, we use *wapiti* with the "rprop-" algorithm.. The setting of wapiti is the same as used in Chapter 5.

7.3.2.2 Main Results

Table 7.1 summarizes the tagging results on the development data with different feature configurations. In this table, the symbol "+" in the *Features* column means

Features	Data	Brown	MKCLS
Supervised	CoNLL	94.48%	
+c100	+1991	94.70%	94.72%
+c500	+1991	94.73%	94.76%
+c1000	+1991	94.68%	94.73%
+c100	+1991-1995	94.90%	94.97%
+c500	+1991-1995	94.94%	94.88%
+c1000	+1991-1995	94.89%	94.94%
+c100	+1991-2000	94.82%	94.93%
+c500	+1991-2000	94.92%	94.99%
+c1000	+1991-2000	94.90%	95.00%
+c100	+1991-2004	- -	94.87%
+c500	+1991-2004	- -	**95.02%**
+c1000	+1991-2004	- -	94.97%

Table 7.1: Tagging accuracies with different feature configurations on the development data.

features of current configuration contains both the baseline features and new features for semi-supervised learning; the number is the total number of the clusters; the symbol "+" in the *Data* column means which portion of the Gigaword data is used to cluster words. For example, "+1991-2000" means the Xinhua News Agency from 1991 to 2000 are used for clustering. From this table, we can clearly see the impact of word clustering features on POS tagging. The new features lead to substantial improvements over the strong supervised baseline. Moreover, these increases are consistent regardless of the clustering algorithms. Both clustering algorithms contributes to the overall performance equivalently. A natural strategy for extending current experiments is to include both clustering results together. However, we find no further improvement. For each clustering algorithm, there are not much differences among different sizes of the total clustering numbers. When small size of unlabeled data (one year's data) is added, the semi-supervised learning only yields minor improvements. When a comparable amount of unlabeled data (five years' data) is used, the further increase of the unlabeled data does not lead to much changes of the performance.

		Supervised			Semi-supervised		
	#words	P	R	F	P	R	F
AD(↑)	3448	94.94%	94.08%	94.51	95.71%	94.58%	95.14
CC(↑)	720	89.89%	92.64%	91.24	90.52%	94.17%	92.31
CD(↑)	1619	97.01%	98.33%	97.67	97.26%	98.70%	97.98
CS(↑)	85	92.59%	88.24%	90.36	92.68%	89.41%	91.02
DEC(↑)	1101	84.64%	75.57%	79.85	84.00%	77.75%	80.75
DEG(↑)	1258	81.46%	90.46%	85.73	82.64%	89.67%	86.01
DER(↑)	18	91.67%	61.11%	73.33	92.86%	72.22%	81.25
DEV(↑)	68	82.14%	67.65%	74.19	90.74%	72.06%	80.33
DT(↑)	640	97.82%	97.97%	97.89	97.83%	98.44%	98.13
JJ(↑)	1363	88.25%	81.58%	84.79	88.35%	82.32%	85.23
LC(↑)	767	96.16%	97.91%	97.03	96.90%	97.91%	97.41
M(↑)	1340	95.78%	98.13%	96.94	96.34%	98.28%	97.30
MSP(↑)	113	91.30%	92.92%	92.11	91.45%	94.69%	93.04
NN(↑)	14015	94.17%	95.13%	94.65	95.18%	95.42%	95.30
NR(↑)	3445	95.61%	93.00%	94.29	95.48%	95.70%	95.59
NT(↑)	1049	97.88%	96.76%	97.32	97.99%	97.52%	97.75
P(↑)	1916	94.89%	94.10%	94.50	95.51%	94.47%	94.99
PN(↑)	653	98.16%	97.86%	98.01	98.61%	98.01%	98.31
PU(↑)	6593	100.00%	99.95%	99.98	100.00%	99.98%	99.99
VA(↑)	501	85.34%	83.63%	84.48	83.59%	86.43%	84.99
VC(↑)	501	96.81%	97.01%	96.91	97.80%	97.80%	97.80
VV(↑)	7121	91.08%	91.93%	91.50	92.38%	92.84%	92.61
AS(↓)	446	98.00%	99.10%	98.55	98.21%	98.65%	98.43
OD(↓)	145	96.67%	80.00%	87.55	94.35%	80.69%	86.99
SP(↓)	53	77.08%	69.81%	73.27	76.60%	67.92%	72.00
VE(↓)	297	99.32%	98.65%	98.99	98.65%	98.65%	98.65
BA	78	97.40%	96.15%	96.77	97.40%	96.15%	96.77
ETC	142	99.30%	100.00%	99.65	99.30%	100.00%	99.65
LB	46	97.62%	89.13%	93.18	97.62%	89.13%	93.18
SB	77	96.15%	97.40%	96.77	96.15%	97.40%	96.77

Table 7.2: Detailed tagging accuracies of the baseline model and the "+c500(MKCLS)+1991-2004" model on the development data.

From the experiments on the development data, we find that the "+c500(MKCLS) +1991-2004" semi-supervised model works best. So we use this setting in the following experiments to show the final impact and also to characterize typical errors. We report detailed tagging performance of different classes of words with and without word clustering features in Table 7.2. We can see that for most types of words, including close classes, the prediction accuracy is improved. The improved performance of the close classes or function words suggests that the word clustering is useful not only for dealing with the data sparseness problem, but also for providing good clues for disambiguation.

7.3.2.3 Learning Curves

We do additional experiments to evaluate the effect of the derived features as the amount of labeled training data is varied. We also use the "+c500(Mkcls)+1991-2004" setting for these experiments. Table 7.3 summarizes the accuracies of the systems when trained on smaller portions of the labeled data. We can see that the new features obtain consistent gains regardless of the size of the training set. The error is reduced significantly on all data sets. In other words, the word cluster features can significantly reduce the amount of labeled data required by the learning algorithm. The relative reduction is greatest when smaller amounts of the labeled data are used, and the effect lessens as more labeled data is added.

Size	Baseline	+Cluster
4.5K	90.10%	91.93%
9K	92.91%	93.94%
13.5K	93.88%	94.60%
18K	94.24%	94.77%

Table 7.3: Tagging accuracy on the development data. Size=#sentences in the training corpus.

7.3.2.4 Two-fold Effect

Word clustering derives paradigmatic relational information from unlabeled data by grouping words into different sets. As a result, the contribution of word clustering to POS tagging is two-fold. On the one hand, word clustering captures and abstracts context information. This new linguistic *knowledge* is thus helpful to better correlate a word in a certain context to its POS tag. On the other hand, the clustering of the OOV words to some extent fights the sparse data problem by correlating an OOV word with IV words through their classes. To evaluate the two contributions of the word clustering, we limit entries of the clustering lexicon to only contain IV words, i.e. words appearing in the training corpus. Using this constrained lexicon, we train a new "+c500(MKCLS)+1991-2004" model and report its prediction power in Table 7.4. The gap between the baseline and *+IV clustering* models can be viewed as the contribution of the first effect, while the gap between the *+IV clustering* and *+All clustering* models can be viewed as the second contribution. This result indicates that the improved predictive power of our semi-supervised model partially comes from the new interpretation of a POS tag through a clustering, and partially comes from its memory of OOV words that appears in the unlabeled data.

Features	Acc.
Supervised	94.48%
+IV clustering	94.70%
+All clustering	95.02%

Table 7.4: Tagging performance with IV clustering on the development data.

Table 7.5 shows the recall of OOV words on the development data set. Only the word types appearing more than 10 times are reported. The recall of all OOV words are improved, especially of proper nouns (NR) and common verbs (VV). This table is also helpful to understand the impact of the clustering information on the prediction of OOV words.

	#Words	Tagger	Semi-Tagger	Δ
AD	21	33.33%	42.86%	$<$
CD	249	97.99%	98.39%	$<$
JJ	86	3.49%	26.74%	$<$
NN	1028	91.05%	91.34%	$<$
NR	863	81.69%	88.76%	$<$
NT	25	60.00%	68.00%	$<$
VA	15	33.33%	53.33%	$<$
VV	402	67.66%	72.39%	$<$

Table 7.5: The tagging recall of OOV words (frequency>10) on the development data.

7.3.2.5 Combining with the Berkeley Parser

In Chapter 5, we enhance the baseline tagger with the help of the Berkeley parser. The motivation of that work is to better capture syntagmatic relations among words for POS tagging, which is complementary to the focus here, namely to better capture

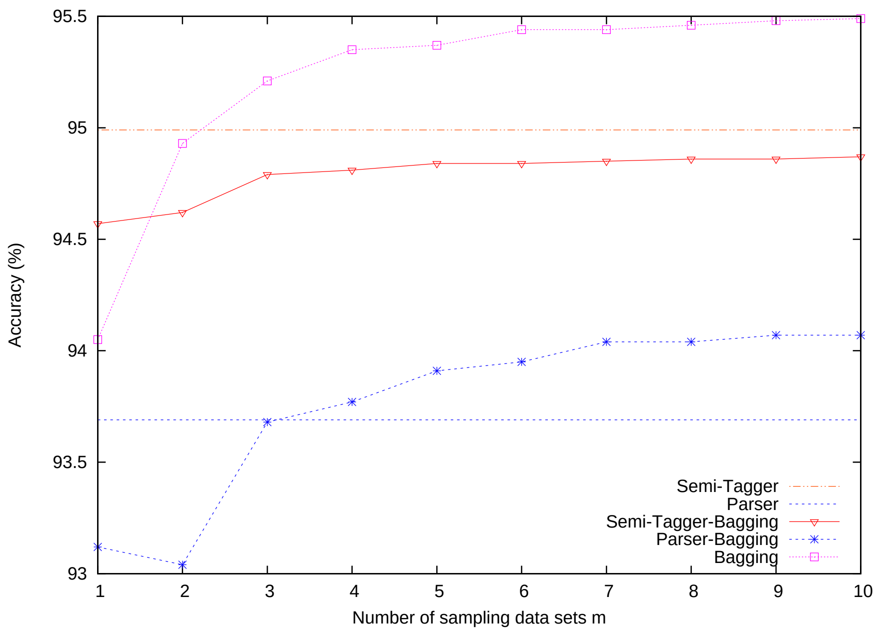


Figure 7.1: Tagging accuracy of Bagging models with different numbers of sampling data sets. *Semi-Tagger-Bagging* means that the Bagging system built on the tagger with word cluster information. *Parser-Bagging* is named in the same way.

paradigmatic relations. We therefore expect further improvement by combining both enhancements. We still use a bagging model to integrate the discriminative tagger and the Berkeley parser. The only difference between current experiment and the experiment in Chapter 5 is that the sub-tagging models are trained with help of word clustering features. For more details, please refer to Section 5.5.3.1. Table 7.1 is the final result of the bagging model on the development data set. We can see that Bagging is effective to combine POS taggers designed with different views. And another important observation is the the improvements that come from two ways, namely capturing syntagmatic and paradigmatic relations, are not much overlap and therefore the combination of both gives more improvement.

7.3.2.6 Final Results

Systems	Data	Cluster	Acc.
Baseline	CoNLL	- -	94.33%
Tagger+Parser Bagging($m = 15$)	CoNLL	- -	94.96%
Semi-Tagger(+c500)	+1991-2004	MKCLS	94.85%
Semi-Tagger(+c500)+Parser Bagging($m = 15$)	+1991-2004	MKCLS	95.34%

Table 7.6: Tagging performance on the test data.

Table 7.6 shows the performance of different systems evaluated on the test data. The final result is very promising. The word clustering features and the bagging model result in relative error reductions of 17.8% in terms of the classification accuracy. The significant improvement of the POS tagging also help successive language processing. Results in Table 7.7 indicate that the parsing accuracy of the Berkeley parser can be simply improved by inputting the Berkeley parser with the POS Bagging results.

Tagger	LP	LR	F
Berkeley	82.71%	80.57%	81.63
Bagging	82.96%	81.44%	82.19

Table 7.7: Parsing accuracies on the test data. (CoNLL)

7.4 Experiments in Text Chunking

7.4.1 Discriminative Text Chunking

Chunking identifies the non-recursive cores of various types of phrases in text, possibly as as a precursor to full parsing or information extraction. It consists of dividing a text into phrases in such a way that syntactically related words become member of the same phrase. These phrases are non-overlapping which means that one word can only be a member of one chunk. The definition of syntactic chunks is illustrated in Figure 7.2. Chunks have been represented as groups of words between square brackets. For example, "保险公司/the insurance company", consisting of two nouns, is a noun phrase.

There has been some research on Chinese text chunking, and a variety of chunk definitions have been proposed. However, most of these studies did not provide sufficient detail. In our system, we use chunk definition presented in [Chen et al., 2006], which provided a chunk extraction tool. The tool to extract chunks from CTB was developed by modifying the English tool used in CoNLL-2000 shared task, Chunklink[1], and is publicly available[2]. For more information about the chunk definition, readers may refer to the original paper.

The state-of-the-art supervised solution for English text chunking leverage on discriminative sequential labeling techniques, such as CRFs [Sha and Pereira, 2003]. With IOB2 representation [Ramshaw and Marcus, 1995], the problem of chunking

[1]http://ilk.uvt.nl/team/sabine/chunklink/chunklink_2-2-2000_for_conll.pl
[2]http://www.nlplab.cn/chenwl/chunking.html

截止	目前	保险	公司	已	为	三峡	工程	提供	保险	服务
[P]	[NT]	[NN	NN]	[AD]	[P]	[NR]	[NN]	[VP]	[NN	NN]
PP	NP	NP		ADVP	PP	NP	NP	VP	NP	

Figure 7.2: An example from of Chinese chunking: *Until now, the insurance company has provided insurance services for the Sanxia Project.*

w_{-3}=“BOS”; w_{-2}=“截止”; w_{-1}=“目前”; w=“保险”; w_{+1}=“公司”; w_{+2}=“已”; w_{+3}=“为”; w_{-3}=“BOS”; w_{-2}=“P”; w_{-1}=“NT”; w=“NN”; w_{+1}=“NN”; w_{+2}=“AD”; w_{+3}=“P”;
$w_{-3}\_w_{-2}$=“BOS_截止”; $w_{-2}\_w_{-1}$=“截止_目前”; w_{-1}_w=“目前_保险”; w_w_{+1}=“保险_公司”; $w_{+1}\_w_{+2}$=“公司_已”; $w_{+2}\_w_{+3}$=“已_为”; $w_{-3}\_w_{-2}$=“BOS_P”; $w_{-2}\_w_{-1}$=“P_NT”; w_{-1}_w=“NT_NN”; w_w_{+1}=“NN_NN”; $w_{+1}\_w_{+2}$=“NN_AD”; $w_{+2}\_w_{+3}$=“AD_P”;
$w_{-2}\_w_{-1}\_w$=“截止_目前_保险”; $w_{-1}\_w\_w_{+1}$=“目前_保险_公司”; $w\_w_{+1}\_w_{+2}$=“保险_公司_已”; $w_{-2}\_w_{-1}\_w$=“P_NT_NN”; $w_{-1}\_w\_w_{+1}$=“NT_NN_NN”; $w\_w_{+1}\_w_{+2}$=“NN_NN_AD”;

Table 7.8: An example of features used for Chunking.

can be regarded as a sequence labeling task. Given a sequence of words with their automatically annotated POS tags, a standard statistical chunker tag each word with a label indicating whether the word is outside a chunk (*O*), starts a chunk (*B-type*), or continues a chunk (*I-type*). A number of machine learning algorithms have been exploited, among which CRFs is a very effective model. In this section, we also adopt this method to resolve Chinese text chunking.

7.4.2 Features

7.4.2.1 Baseline Features

Similar to our POS tagger, we employ word *n*-gram features for word disambiguation. The features includes:

- Word/POS unigram feature: w_{-3}, w_{-2}, w_{-1}, w, w_{+1}, w_{+2}, w_{+3};
- Word/POS bigram feature: $w_{-3}\_w_{-2}$, $w_{-2}\_w_{-1}$, w_{-1}_w, w_w_{+1}, $w_{+1}\_w_{+2}$, $w_{+2}\_w_{+3}$;
- POS trigram feature: $w_{-2}\_w_{-1}\_w$, $w_{-1}\_w\_w_{+1}$, $w\_w_{+1}\_w_{+2}$;

That means 18 features are used to represent a given token. Take the word “保险” in Figure 7.2 for example, all features used for chunking are listed in Table 7.8.

7.4.2.2 Cluster-based Features

We consider using word clusters to improve chunking with the feature induction method. Our shallow parser consists of two steps: POS tagging and chunking. The cluster-based features may contribute in both steps, and we have already shown that word clustering is very helpful for Chinese POS tagging. Here we focus on the impact of new features on chunking. The new cluster-based features to assist the chunker includes:

- Unigram cluster feature: w_{-1}, w, w_{+1};
- Bigram cluster feature: w_{-1}_w, w_w_{+1}.

That means 5 new features are added.

7.4.3 Experiments and Analysis

We use the same data setting as the POS tagging experiments. To solve the sequence labeling problem in chunking, we also use *wapiti* with the "rprop-" algorithm.

7.4.3.1 Baseline Performance

		Tagging	Chunking		
Tagger	Chunker	Acc.	P	R	F
CTB	Our chunker	100.00%	92.95%	91.68%	92.31
Our tagger	Our chunker	94.48%	85.94%	84.70%	85.31
Berkeley Parser	Berkeley Parser	93.69%	85.90%	84.28%	85.09

Table 7.9: The tagging accuracy on the development data.

Table 7.9 summarizes the precision, recall, f-scores of our discriminative chunker and the Berkeley parser. To get chunking results from the Berkeley parser, we use the same chunk extraction tool. We can see that Chinese text chunking has reached an accurate performance, when gold POS information is available. However, the state-of-the-art of Chinese POS tagging is far from perfect. In our experiment, when automatic tagger is used, the overall chunking drops more than 7 absolute points. The overall performance of our chunker is 85.31, which is slightly better than the Berkeley parser (85.09). Compared to the full generative parsing, the discriminative sequence labeling technique is relatively competitive for chunking.

7.4.3.2 Comparing Chunking and PCFG-LA Parsing

Although the final accuracies of the Berkeley parser and our discriminative chunker are comparable, the underlying models are quite different and make different types of errors. Figure 7.3 shows the f-score of the two systems for different chunk types. The definition of each chunk type is detailed described in [Chen et al., 2006]. In general, the tagger has slightly better accuracy for nominal structures and related ones, while the parser does better on other categories, which are mainly verbal structures. This pattern is consistent with previous POS tagging results insofar as verbs are often involved in longer distances, while shallow nominal structures with shorter distances.

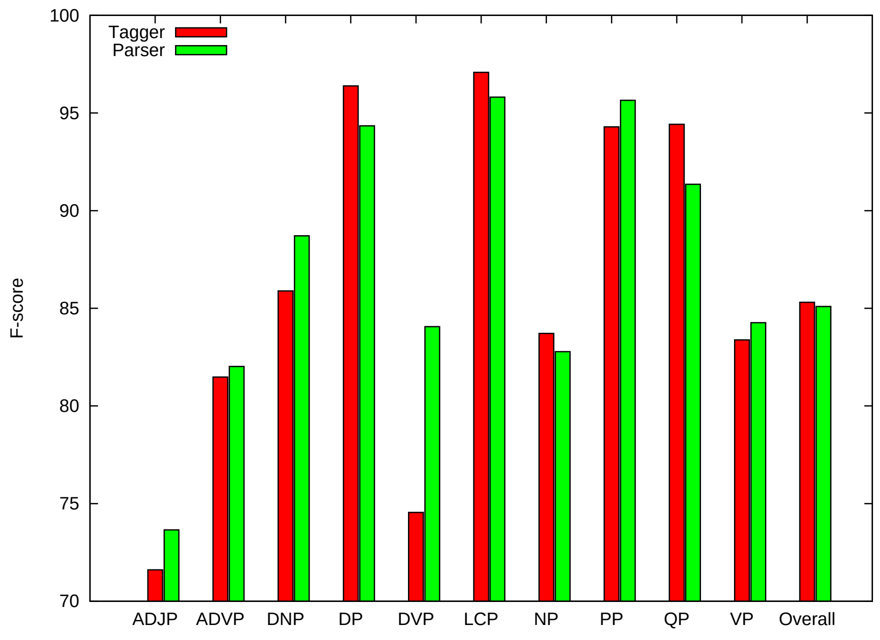


Figure 7.3: Chunking f-scores for different chunk types.

7.4.3.3 Word Clustering is Helpful

Our first set of chunking experiments are performed on the basis of a supervised POS tagger. Our second set of chunking experiments are performed on the basis of a semi-supervised POS tagger which uses the "+c500(MKCLS)+1991-2004" model. Table 7.10 summarizes the results. We can see that cluster-based features are very helpful to enhance the *bracketing and labeling* problem. Similar to the experiments of POS tagging, the clustering algorithms do not affect the final performance much. The contribution of the clustering information to a shallow parser is partially from the POS tagging stage and partially from the chunking stage.

Tagger	Features	Cluster	F
Supervised	Supervised	- -	85.31
Supervised	+c100	Brown+1991-2000	85.66
Supervised	+c500	Brown+1991-2000	**85.88**
Supervised	+c1000	Brown+1991-2000	85.69
Supervised	+c100	MKCLS+1991-2004	85.84
Supervised	+c500	MKCLS+1991-2004	85.87
Supervised	+c1000	MKCLS+1991-2004	85.81
+c500(MKCLS)+1991-2004	+c100	Brown+1991-2000	**86.47**
+c500(MKCLS)+1991-2004	+c500	Brown+1991-2000	86.25
+c500(MKCLS)+1991-2004	+c1000	Brown+1991-2000	86.26
+c500(MKCLS)+1991-2004	+c100	MKCLS+1991-2004	86.32
+c500(MKCLS)+1991-2004	+c500	MKCLS+1991-2004	86.43
+c500(MKCLS)+1991-2004	+c1000	MKCLS+1991-2004	86.29

Table 7.10: Chunking f-scores with different feature configurations on the development data.

There are two main jobs of syntactic chunking: grouping words as basic phrases and classifying their syntactic types. We report the unlabeled bracketing performance in Table 7.11. In other words, detailed phrase category is not considered. These results indicate that word clustering is very helpful to find phrase boundaries.

Tagger	Chunker	P	R	F
Supervised	Supervised	88.09%	86.81%	87.44
Supervised	+c100(Brown)+1991-2000	88.36%	87.38%	87.87
+c500(MKCLS)+1991-2004	+c100(Brown)+1991-2000	89.06%	88.03%	88.54

Table 7.11: Bracketing performance on the development data.

7.4.3.4 Final Results

Table 7.12 is the performance of different systems evaluated on the test data. The final result demonstrates the effectiveness of the application of word clustering. The cluster-based features results in a relative error reduction of 7.1% in terms of the labeled f-score.

Tagger	Chunker	P	R	F
Supervised	Supervised	86.27%	85.22%	85.75
+c500(MKCLS)+1991-2004	+c100(Brown)+1991-2000	87.05%	86.19%	86.62

Table 7.12: Chunking performance on the test data.

7.5 Experiments in Dependency Parsing

Previous experiments on the shallow parsing evaluate the impact of the word clustering on parsing in the constituency formalism. Both the *bracketing* and the *labeling* tasks can benefit from word clusters. Another important type of syntactic structure is the bilexical dependency structures. In this section we evaluate the impact of the MKLCS clusters on dependency parsing.

7.5.1 Cluster-based Features

Principled feature engineering is important for the application of word clusters to dependency parsing. In our experiments, we basically incorporate word clusters as fine-grained POS tags. We copy every real POS tag involved feature and substitute the POS tag as word clusters.

7.5.2 Experiments and Analysis

7.5.2.1 Main Results

In order to evaluate the helpfulness of cluster-based features, we conduct dependency parsing experiments using CoNLL 2009 shared task's data, i.e. the same data setting as the parsing experiments in Chapter 5. Similar to the chunking experiments, we do two sets of experiments on basis of the supervised POS tagger and the semi-supervised tagger respectively. In this chapter, we use a second order graph-based dependency parsing model [Che et al., 2009; Li et al., 2011] for experiments.[1] This parser obtains the best parsing result of the CoNLL shared task. Table 7.13 summarizes the experimental results. These results show that word clustering is very helpful to enhance dependency parsing. The size of the total number of clusters influence the quality of dependency parsing. With the increase of the total number of clusters, both the UAS and the LAS increase.

7.5.2.2 Two-fold Effect

Word clustering derives paradigmatic relational information from unlabeled data, and contribute to dependency parsing by (1) abstracting context information and (2) fighting data sparseness problem. To analyze the two-fold effect, we limit entries of the clustering lexicon to only contain IV words. Using this constrained lexicon, we

[1]We would like to thank Zhenghua Li to provide his implementation and Meishan Zhang to help with the feature configuration.

Tagger	Features	Cluster	UAS	LAS
Supervised	Supervised	- -	82.98%	78.65%
Supervised	+c100	MKCLS+1991-2004	83.60%	79.41%
Supervised	+c500	MKCLS+1991-2004	84.01%	79.85%
Supervised	+c1000	MKCLS+1991-2004	84.16%	79.99%
+c500(MKCLS)+1991-2004	+c100	MKCLS+1991-2004	79.87%	80.01%
+c500(MKCLS)+1991-2004	+c500	MKCLS+1991-2004	84.22%	80.11%
+c500(MKCLS)+1991-2004	+c1000	MKCLS+1991-2004	84.57%	80.46%
+Clustering+Bagging	+c1000	MKCLS+1991-2004	**84.80%**	**80.82%**

Table 7.13: Dependency parsing UAS/LAS with different feature configurations on the development data.

train a new "+c1000(MKCLS)+1991-2004" model and report its prediction power in Table 7.14. Note that, the POS information is provided by the supervised tagger. The gap between the baseline and *+IV clustering* models measures the first contribution, while the gap between the *+IV clustering* and *+All clustering* models measures the second one. This result indicates that the improved accuracy partially comes from the new interpretation of a word through a clustering, and partially comes from its memory of OOV words that appears in the unlabeled data.

Tagger	Features	UAS	LAS
Supervised	Supervised	82.98%	78.65%
Supervised	+IV clustering	83.45%	79.24%
Supervised	+All clustering	84.16%	79.99%

Table 7.14: Dependency performance with IV clustering on the development data.

7.5.2.3 Impact on the Prediction of OOV Words

Word clustering fights the sparse data problem by relating low-frequency words with high-frequency words through their classes. Table 7.15 shows the prediction accuracy of the different types of dependencies. We report four types of dependencies: (1) both the dependent and the head are IV words; (2) the dependent is an IV word while the head is an OOV word; (3) the dependent is an OOV word while the head is an IV word; (4) both the dependent and the head are OOV words. The semi-supervised model for evaluation is the best system available. From this table, we can see a clear gap of predictive power between IV and OOV words. There is a very interesting phenomenon that, when dependencies with OOV dependents are harder to recognize than the ones with OOV heads. We compare the improvements of the OOV and IV words and find that the error reduction of the OOV words are higher. This confirms our motivation to leverage on knowledge exploiting paradigmatic relations among words to better handle the recognition and disambiguation of the OOV words.

			Supervised			Semi-supervised		
Dependent	←	Head	P	R	F	P	R	F
IV	←	IV	84.09%	83.81%	83.95	85.42%	85.12%	85.27
IV	←	OOV	78.16%	79.65%	78.90	80.18%	81.77%	80.97
OOV	←	IV	72.74%	73.46%	73.10	74.94%	75.57%	75.26
OOV	←	OOV	69.84%	64.26%	66.94	74.92%	69.81%	72.28

Table 7.15: Dependency prediction accuracy relative to word type (OOV or IV).

7.5.2.4 Final Results

Table 7.16 is the performance of different dependency models evaluated on the test data. The first line shows the best result reported in the CoNLL 2009 shared task. The cluster-based features results in relative error reductions of 7.2% and 6.9% in terms of the UAS and LAS scores over our baseline.

Tagger	Parser	UAS	LAS
CoNLL 09	[Che et al., 2009]	- -	75.49%
Supervised	Supervised	83.27%	78.64%
+c500(MKCLS)+1991-2004	+c1000(MKCLS)+1991-2004	84.48%	80.11%

Table 7.16: Dependency parsing performance on the test data.

7.6 Conclusion and Discussion

In this chapter, we evaluate the helpfulness of unsupervised word clustering for supervised parsing. Our work is motivated by (1) the importance of rich lexical information for parsing and (2) the performance gap between supervised and unsupervised NLP methods. Our feature induction based semi-supervised approach achieves substantial improvements over competitive baseline systems for Chinese parsing. Experimental results confirm that capturing paradigmatic relations is essential to analyzing syntagmatic relations.

Despite this success, there are several ways in which our work might be improved. We demonstrate the helpfulness of word clustering for shallow chunking and dependency parsing. A natural area for future work is applying word clustering to full constituency parsing. The main difficulty to do so is that most of successful constituency parsers are based on generative models, which are hard to incorporate rich features.

Recall that the popular Brown and MKCLS clustering algorithms are based on a bigram language model. Intuitively, there is a *mismatch* between the kind of lexical information that is captured by the Brown/MKCLS clustering and the kind of lexical information that is modeled in supervised POS tagging, chunking and dependency parsing. A natural avenue for further research would be exploiting other type of lexical knowledge that reflect the syntactic behavior of words.

Part III

Semantic Role Labeling

Chapter 8

Full and Partial Parsing Based Semantic Chunking

State-of-the-art Chinese semantic role labeling (SRL) systems leverage full parsing to find arguments and classify their semantic types. To better utilize syntactic information, which is crucial to the success of SRL, we propose a semantic chunking method together with linguistically rich syntactic features. Our system achieves an F-score of 93.41, which is significantly better than the best reported performance, 92.0. We also empirically analyze the effect of full parsing in Chinese SRL. Motivated by developing a complementary method, we study an alternative lightweight solution which only makes use of partial syntactic parses. Furthermore, we present a comparative analysis of the two categories of methods. This analysis could be exploited to improve SRL accuracy by system ensemble.

The rich syntactic features used in full parsing based SRL system is introduced in [Sun, 2010a], and the partial parsing based method is introduced in [Sun et al., 2009a]. To lead to a fair comparison, we repeat experiments with slight modifications of the original papers.

8.1 Background

8.1.1 The Problem

In the last decade, there has been an increasing interest in semantic role labeling (SRL) on several languages, which consists of recognizing arguments involved by predicates in a given sentence and labeling their semantic types. Typical semantic classes include *Agent*, *Patient*, *Source*, *Goal*, and so forth, which are core arguments to a predicate, as well as *Location*, *Time*, *Manner*, *Cause*, and so on, which are adjuncts. In order to indicate exactly what semantic relations hold among a given predicate and its associated participants and properties, the role-bearing constituents must be identified and their correct semantic role labels assigned, as in:

- [警察]$_{Agent}$[正在]$_{Time}$[详细]$_{Manner}$[调查]$_{Predicate}$[事故原因]$_{Patient}$
- [The police]$_{Agent}$ are [thoroughly]$_{Manner}$ [investigating]$_{Predicate}$ [the cause of the accident]$_{Patient}$.

In the given example, the predicate in question is "调查/investigate". All arguments and adjuncts involved by "调查" have been represented as groups of words between square brackets. A tag next to the close bracket denotes the role of the argument (or adjunct). For example, the tag *Agent* indicates the doer, "警察/police," of the investigation event, since they initiates and sustains the action; the tag *Manner* indicates an adjunct of the target verb, since it notes how the process of an event is carried out.

Such sentence-level semantic analysis of text is concerned with the characterization of events and is therefore important to understand the essential meaning of the original input language sentences – *who* did *what* to *whom*, for *whom* or *what*, *how*, *where*, *when* and *why*? Different from many other information representation formalisms, this shallow semantic interpretation is independent of domains and more robust for many application purposes. SRL abstracts important semantic information away from syntactic structure and may potentially benefit many deep NLP tasks such as question answering, textual entailment, and complex information extraction.

8.1.2 The Annotation Data

Since the seminal work of [Gildea and Jurafsky, 2002], statistical and machine learning approaches have been the predominant research paradigm in SRL, like many other subfields in NLP. A pre-requisite for statistical and machine learning approaches to SRL is the availability of a significant amount of semantically interpreted corpora from which automatic systems can learn. The recent activities in SRL have in large part been driven by the availability of semantically annotated corpora such as the FrameNet [Baker et al., 1998], PropBank [Palmer et al., 2005], and Nombank [Meyers et al., 2004] projects for English; the tectogrammatical layer for Czech; and the Salsa Project [Burchardt et al., 2006] for German; the Chinese PropBank [Xue and Palmer, 2009] for Chinese.

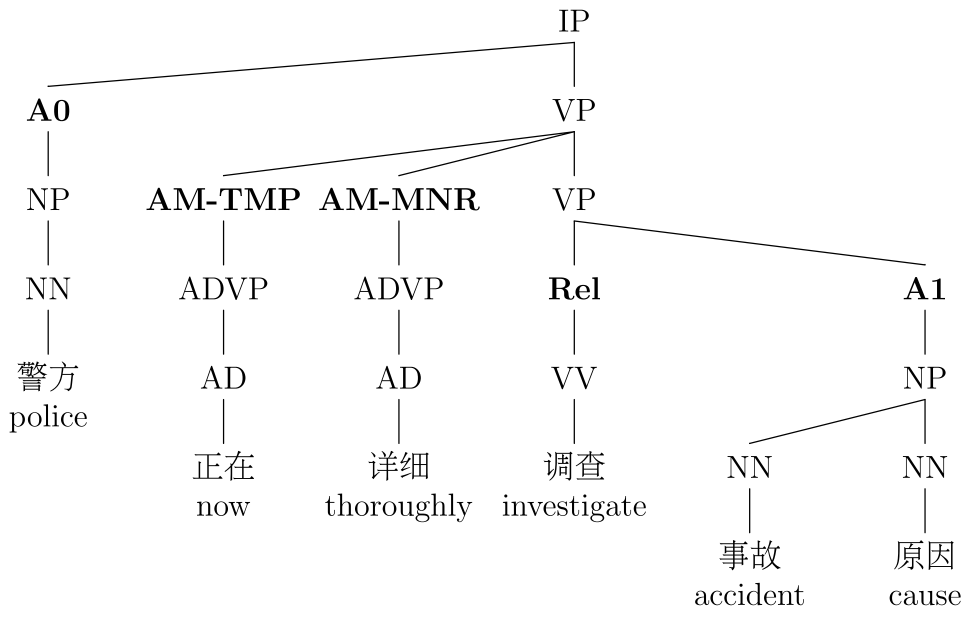


Figure 8.1: An example sentence for CTB and CPB: *The police are thoroughly investigating the cause of the accident.*

The Chinese PropBank (CPB) is a popular semantically annotated corpus for research on Chinese SRL. It adds a layer of predicate-argument structures to the Chinese TreeBank. It assigns semantic role labels to syntactic constituents (rather than to the head words in a dependency structure) in a sentence. Each verb is annotated with a fixed number of arguments and each argument plays a role with regard to the verb. The arguments of a predicate are labeled with a contiguous sequence of integers, in the form of AN (N is a natural number); the adjuncts are annotated as such with the label AM followed by a secondary tag. For the core arguments, CPB uses a set of predicate-specific semantic role labels Predicates vary on the number of core arguments they take, but generally the total number of core arguments does not exceed six. Secondary tags for the adjuncts provide semantic information such as location, manner, and time that are not specific to a particular verb or even a particular class of verbs and they are defined based on a general set of guidelines. The assignment of semantic roles is illustrated in Figure 8.1, where the predicate is the verb "调查/investigate". E.g., the NP "事故原因/the cause of the accident" is labeled as *A1*, meaning that it is the *Patient*, the PP "正在/now" is labeled as *AM-TMP*, indicating a temporal component.

With the advent of this supporting resource, Chinese SRL has become a well-defined task with a substantial body of work and comparative evaluation.

8.1.3 Successful Methods for English SRL

The work on SRL has included a broad spectrum of statistical and machine learning approaches to the task. Most SRL research takes an approach requiring training on role-annotated data. In this chapter, we only focus on supervised approaches. Given a sentence and a designated verb, the SRL task consists of identifying the boundaries of the arguments of the verb predicate and classifying them with semantic roles. The most common architecture for state-of-the-art SRL systems consists of the following steps.

- The first step in SRL is full syntactic parsing, which provide rich syntactic information for semantic processing.
- The second step typically consists of linguistically motivated pruning the set of argument candidates for a given predicate.
- The third step consists of a local classification of argument candidates. By "local," we mean that candidates are usually treated independently of each other.
- The last step is to apply joint inference in order to combine the predictions of local scorers to produce a good structure of all labeled arguments for the predicate.

A variety of research has been proposed to capture different characteristics of SRL. More recent approaches for English SRL have involved calibrating features [Gildea and Jurafsky, 2002; Xue and Palmer, 2004], analyzing the complex input – syntax trees [Liu and Sarkar, 2007; Moschitti, 2004], exploiting the complicated output – the predicate-argument structure [Punyakanok et al., 2004; Toutanova et al., 2005], system combination [Punyakanok et al., 2004; Surdeanu et al., 2007], as well as capturing paradigmatic relations between predicates [Gordon and Swanson, 2007].

8.1.4 Previous Work on Chinese SRL

Previous work on Chinese SRL mainly focus on how to implement SRL methods which are successful on English, such as [Ding and Chang, 2008; Sun and Jurafsky, 2004; Xue, 2008; Xue and Palmer, 2005; Zhuang and Zong, 2010]. Full parsing based SRL methods that are successful on English are adopted to resolve Chinese SRL. Sun and Jurafsky [2004] did the preliminary work on Chinese SRL without any large semantically annotated corpus of Chinese. They adopt English SRL methods presented in [Gildea and Jurafsky, 2002; Pradhan et al., 2003] and evaluate ten specified verbs with a small collection of Chinese sentences. This work make the first attempt on Chinese SRL and produce promising results.

After the CPB was built, [Xue and Palmer, 2005] and [Xue, 2008] produce more complete and systematic research on Chinese SRL. Their work shows that when gold parses are available, the performance of Chinese SRL is fairly encouraging, achieving an f-score of 92.0. When an automatic parser is used, the performance, however, is highly degraded, only achieving an f-score of 71.9. This indicates the importance of syntactic parsing for well-performed SRL systems. In [Ding and Chang, 2008], SRC is divided into two sub-tasks in sequence: Each argument should first be determined whether it is a core argument or an adjunct, and then be classified into fine-grained categories. However, delicately designed features are more important and our experiments suggest that by using rich features, a better SRC solver can be directly trained without using hierarchical architecture.

Dependency is another popular formalism to represent syntactic and semantic information in NLP. In the CoNLL 2008 shared task, Surdeanu et al. [2008] propose a unified dependency-based formalism to model both syntactic dependencies and semantic roles for English. The CoNLL 2009 shared task is an extension of the CoNLL 2008. It dedicates to the joint parsing of syntactic and semantic dependencies in multiple languages [Hajič et al., 2009]. As a sub-task of CoNLL 2009, Chinese semantic dependency parsing method are well evaluated.

8.2 Full Parsing Based Semantic Chunking with Rich Syntactic Features

8.2.1 Motivation

State-of-the-art Chinese SRL systems leverage rich syntactic information, which is normally provided by one (or many) parser(s). The contribution of syntactic parsing to SRL is two-fold. On one hand, SRL systems should group words as argument candidates, which are also constituents in a given sentence. Parsing can effectively supply SRL with argument candidates or at least save effort for semantic processing. On the other hand, given a constituent, SRL systems should identify whether it is an argument and further predict detailed semantic types if it is an argument. For the prediction problems, parsing can provide expressive features.

Full parsing provides boundary information of all constituents. As arguments should c-command[1] the predicate, a full parser can further prune a majority of useless constituents. In summary, parsing can effectively supply SRL with argument candidates. Unfortunately, it is very hard to correctly produce full parses for Chinese texts. We will show in the following experiments that the main challenge of Chinese SRL is to find boundaries of arguments rather than to classify them. Our first concern is to better utilize syntactic boundary information through semantic chunking, which improve system recall by *re*-bracketing some constituents into arguments with semantic clues.

Developing features that capture the right kind of information encoded in the input parses has been shown crucial to advancing the state-of-the-art. Though there has been some work on feature design in Chinese SRL, information encoded in the syntactic trees is not fully exploited and feature engineering requires more research effort. Our second concern is fine-grained feature engineering for Chinese SRL. We introduce a set of additional features, some of which are designed to better capture structural information of sub-trees in a given parse.

8.2.2 Constituent Classification System

To evaluate the impact of syntactic features, we implement a constituent classification system as a baseline. Following [Xue, 2008], our system divides SRL into three sub-tasks: 1) pruning with a heuristic rule, 2) argument identification (AI) to recognize arguments, and 3) semantic role classification (SRC) to predict semantic types. To efficiently excluded non-arguments, a pruning procedure is executed before AI. Our pruning strategy is to keep all constituents (except punctuations) that c-command current predicate in focus as argument candidates. The latter two sub-tasks, AI and SRC, are formulated as two classification problems. In other words, a binary classifier is trained to classify each argument candidate as either an argument or not. Finally, a multi-class classifier is trained to label each argument recognized in the former stage with a specific semantic role label. The main job of both AI and SRC steps is to select strong syntactic features from a given parse.

8.2.3 Constituent Chunking System

In a traditional constituent classification system, SRL is performed on the output of a syntactic parser, and only phrases in the parse tree are taken as possible candidates.

[1]The concept *c-command* comes from the X-bar theory. Assuming α and β are two nodes in a syntax tree: α c-commands β means every parent of α is an ancestor of β.

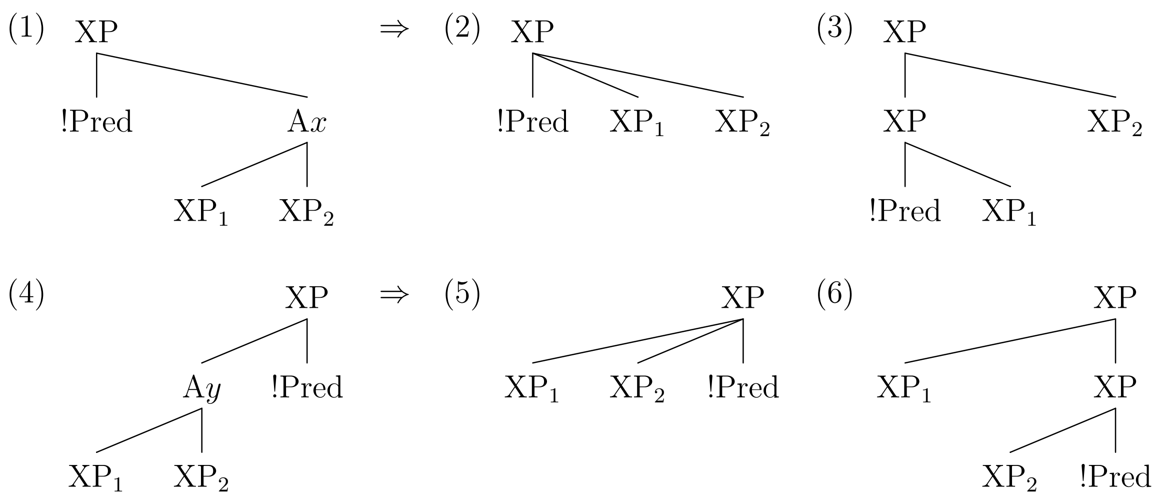


Figure 8.2: Parsing errors that can be tolerated by full parsing based constituent chunking.

If there is no phrase in the parse tree that shares the same text span with an argument in the manual annotation, the system cannot possibly get a correct prediction. In other words, the best the system can do is to correctly label all arguments that have a counterpart node in the parse tree.

We introduce an idea to perform semantic chunking over large phrases provided by a full parser: To detect unreliable constituents, break them into smaller component constituents, and re-bracket them using semantic clues. We hope to improve system recall through combining some constituent nodes. Two issues are raised to re-bracket unreliable constituents: (1) How to detect the so-called unreliable constituents, and (2) is it possible to correctly re-bracket them? We simplify the detection of unreliable constituents and exemplify the possibility by just considering c-commanders.

Our constituent chunking system first collects all c-commanders and puts them in order. Because c-commanders of a predicate are not overlapped with each other and compose the whole sentence, we can take this step as a sequentialization procedure. On basis of sequentialized constituents, we define semantic chunks which do not overlap nor embed using IOB2 representation and transfer the SRL problem as a **constituent tagging** problem. Our definition of semantic chunks is described below.

- Constituent outside an argument receive the tag *O*.
- For a sequence of constituents forming a semantic role of *Ax*, the first constituent receives the semantic chunk label *B(egin)-Ax*,
- and the remaining ones receive the label *I(nside)-Ax*.

We are interested in tolerating two types of parsing errors that are shown in Figure 8.2. Assume tree structures (1 and 4) on the left hand side are the correct syntactic analysis, while tree structures (2, 3, 5 and 6) on the right hand side are some wrong analysis. Though a constituent classification system, the arguments Ax and Ay can not be recovered since there is no node to express them. Though our constituent chunking system, however, when these errors occur, the arguments can still be found, if XP$_1$ is assigned a label *B-Ax* or *B-Ay* and XP$_2$ is assigned a label *I-Ax* or *I-Ay*.

8.2.4 Features

Part of features used in our system are a combination of features described in [Ding and Chang, 2008; Xue, 2008] as well as the c-command thread features proposed in [Sun et al., 2008]. We explain new features in details but only give a brief description of features used in previous work. For more information about the old features, readers can refer to the relevant papers.

To conveniently illustrate, we denote a candidate constituent c_k with a fixed context $w_{i-1}/p_{i-1}, [_{c_k} w_i/p_i, ..., w_h/p_h, ..., w_j/p_j], w_{j+1}/p_{j+1}$, where w_h is the head word of c_k and p_l is the associated POS tag of the word w_l, and denote predicate in focus with a context $w^v_{-2}/p^v_{-2}, w^v_{-1}/p^v_{-1}, w^v/p^v, w^v_{+1}/p^v_{+1}, w^v_{+2}/p^v_{+2}$, where w^v is the predicate in focus. We seperate all features into three sets: (1) word features that can extracted based on word sequence and its associated POS tag sequence, (2) syntactic features that can be only extracted from a full parsing tree, and (3) additional SRC feature that are only used in SRC. For example, to extract the head word features of a constituent, we must know the internal syntactic structure of that constituent. As a result, any feature based on a head word are classified as a syntactic feature, even when it is just the surface string form of a head word. All word features and syntactic features are used for the AI classification and semantic chunking, and all features are used for SRC classification.

8.2.4.1 Word Features

- *Word features*: w^v, w^v_{-1}, w^v_{+1}, w_i, w_{i-1}, w_j, w_{j+1}, w_i+w_j.
- *POS features*: p^v, p^v_{-2}, p^v_{-1}, p^v_{+1}, p^v_{+2}, p_i, p_{i-1}, p_{i-2}, p_j, p_{j+1}, p_{j+2}, , p_i+p_j.
- *Word before "LC"*: If the POS of w_j is "LC" (localizer), we use w_{j-1} (lc^w_{j-1}) and p_{j-1} (lc^p_{j-1}) as two new features.
- *Length of c_k* (plen): How many words are there in c_k.

- *Position* (posi) of c_k relative to the predicate.
- Morphological features: *First character* ($char_{+1}$), *last character* ($char_{-1}$) and *word length* (wlen) of w^v, $char_{+1}$+wlen, $char_{-1}$+wlen, $char_{+1}$+posi, $char_{-1}$+posi.
- *Verb class* (vclass) of w^v. Xue [2008] put forward a rough verb classification where verb classes are automatically derived from the frame files, which are verb lexicon for the CPB annotation. This kind of verb class information has been shown very useful for Chinese SRL. Our system also includes this feature. In our experiments, we represent a verb in two dimensions: 1) number of arguments, and 2) number of framesets. For example, a verb may belong to the class "C1C2," which means that this verb has two framesets, with the first frameset having one argument and the second having two arguments.

8.2.4.2 Syntactic Features

In SRL, the objects being modeled are syntax trees which require some mechanism to convert them into feature vectors. Taking complex syntax trees as inputs, the classifiers should characterize their structural properties. We put forward a number of new features to encode the structural information.

- *Pseudo subcategorization frame* (scf): The CFG rewrite rule expanding the parent node of w^v.
- *Parse tree path features*: The *path* feature is defined as the path from a source constituent node n_s through the parse tree to a target constituent node n_t, represented as a string of parse tree node linked by symbols indicating upward (or downward) movement through the tree. The path feature describes the syntactic relation between two constituents and has been shown very important for semantic classification. We define several different path features for Chinese SRL, and list them as follows.
  - Path from c_k to w^v ($path(c_k, w^v)$),
  - We denote the lowest ancestor of c_k and w^v as $a(c_k, w^v)$ and use as two new features the path from c_k to $a(c_k, w^v)$ ($path(c_k, a(c_k, w^v))$) and the path from w^v to $a(c_k, w^v)$ with the word content of w^v ($path(w^v, a(c_k, w^v))+w^v$).
  - We denote the root node of current parse tree as *root* and use as a new feature the path from c_k to the root ($path(w^v, root)$).

C1: VCD, VCP, VNV, VP, VPT, VRD, VSB				
C2: DNP, DP, FW, NN, NP, PN				
C3: ADVP, DVP, MSP			C4: LCP, PP	
C5: CP, FRAG, IP			C6: CLP, QP	
C7: ADJP	C8: LST	C9: PP	C10: PRN	C11: UPC
C12: Other categories				

Table 8.1: Category Clusters

- *Clustered parse tree path features*: We substitute the node labels of a standard path feature as their manually created clusters, and define several new features. The clusters of CTB categories is list in Table 8.1. Our new features include the clustered path from c_k to w^v ($cpath(c_k, w^v)$), the clustered path from c_k to $a(c_k, w^v)$ ($cpath(c_k, a(c_k, w^v))$), and the clustered path from c_k to $root$ ($cpath(c_k, root)$).

- *Phrasal category features*: Categories of c_k, c_k's parent, lift and right siblings. (c_k, c_k^p, c_k^l, c_k^r)

- *Head word features*: Head words and their associated POS tag of c_k (w_h, p_h), its parent (w_h^p, p_h^p), left and right siblings (w_h^l, p_h^l, w_h^r, p_h^r). We also combine head words and other features as conjunction features, including w_h+posi, w_h+w^v, w_h+posi+w^v, w_h+$path(c_k, a(c_k, w^v))$. To extract the syntactic head of a phrase, we use head rules described in [Sun and Jurafsky, 2004]. This set of head rules are very popular in Chinese parsing research, such as in [Duan et al., 2007; Zhang and Clark, 2008b].

- *Noun head of prepositional phrases*: Many adjunctive roles, such as temporals and locatives, occur as prepositional phrases in a sentence, and it is often the case that the head words of those phrases, which are always prepositions, are not very discriminative Therefore we include as two new features (w_h^n, p_h^n) the head word and its associated POS tag of the first noun phrase inside the prepositional phrase.

- *NT* (has_NT): If c_k is a prepositional phrase, does c_k contain a word with POS "NT" (temporal noun)?

- *CFG rewrite rule* that expands c_k and c_k's parent

- *Rewrite rule features*: The CFG rewrite rule expanding c_k and the parent of c_k (rule(c_k), rule($par(c_k)$)).

- *Lexicalized rewrite rules*: Conjunction of rewrite rule and head word of its corresponding RHS. These features of c_k (lrule(c_k)) and its parent (lrule($par(c_k)$)) are used.

- *Head Trace*: The sequential container of the head down upon the phrase. This feature is designed for function tag labeling, which is presented in the earlier work of the author [Sun and Sui, 2009]. We design two kinds of traces (htr-p, htr-w): one uses POS of the head word; the other uses the head word word itself.

- *C-commander thread of the head* C-commander thread features are designed for the prediction of maximal projections, which is presented in the earlier work of the author [Sun et al., 2008]. Hold the same principle, we define two new features (cct-c, cct-w) as sequential containers of constituents which C-command target predicate. The difference of the two features is that the *cct-c* feature uses the POS information while the *cct-w* feature uses the word content information of current predicate.

To better explain our features, we take the last noun phrase "事故原因/the cause of the accident" in Figure 8.1 for example and list all the features in Table 8.2.

8.2.4.3 Additional Features for SRC

In the SRC stage of our constituent classification system, to gather all argument position information predicted in AI step, we design a *coarse frame* feature (cframe), which is a sequential collection of arguments. So far, we do not know the detailed semantic type of each argument, so we use *XP* as each item in the frame. To distinguish the argument in focus, we use a special symbol to indicate the associated frame item. For instance, the coarse frame feature for the argument 事故原因 is *XP+XP+XP+V+!XP*, where *!XP* means that it is the argument in focus. We also respectively combine the coarse frame with the predicate, the predicate and the head word, the last character of the predicate, and the verb class of the last predicate as four new features (cframe+w^v, cframe+w^v+w_h, cframe+$char_{-1}$ and cframe+vclass).

Word features

Word features	w^v=“调查”; w^v_{-1}=“详细”; w^v_{+1}=“事故”; w_i=“事故”; w_{i-1}=“调查”; w_j=“原因”; w_{j+1}=“EOS”; $w_i + w_j$=“事故+原因”
POS features	p^v=“VV”; p^v_{-2}=“AD”; p^v_{-1}=“AD”; p^v_{+1}=“NN”; p^v_{+2}=“NN”; p_i=“NN”; p_{i-1}=“VV”; p_{i-2}=“AD”; p_j=“NN”; p_{j+1}=“EOS”; p_{j+2}=“EOS”; $p_i + p_j$=“NN+NN”
Morphology features	$char_{-1}$=“调”; $char_{+1}$=“查”; wlen=2; $char_{-1}$+wlen=“调+2”; $char_{+1}$+wlen=“查+2”; $char_{-1}$+posi=“调+after”; $char_{+1}$+posi=“查+after”;
Other features	plen=2; posi=“after”; vclass=“C2”;

Syntactic features

Path features	$path(c_k, w^v)$=“NP↑VP↓VV”; $path(c_k, a(c_k, w^v))$=“NP↑VP”; $path(w^v, a(c_k, w^v))$=“VV↑VP”; $path(c_k, root)$=“NP↑VP↑IP”; $cpath(c_k, w^v)$=“C2↑C1↓C12”; $cpath(c_k, a(c_k, w^v))$=“C2↑C1”; $cpath(c_k, root)$=“C2↑C1↑C5”;
Head features	w_h=“原因”; p_h=“NN”; w^p_h=“调查”; p^p_h=“VV”; w^l_h=“调查”; p^l_h=“VV”; w^r_h=“NULL”; p^r_h=“NULL”; w_h+posi=“原因+after”, w_h+w^v=“原因+调查”, w_h+posi+w^v=“原因+after+调查”, w_h+$path(c_k, a(c_k, w^v))$=“NP(原因)↑VP”; w^n_h=“NULL”; p^n_h=“NULL”;
Category features	c_k=“NP”, c^p_k=“VV”, c^l_k=“VV”, c^r_k=“NULL”)
CFG rule features	rule(c_k)=“NP→NN+NN”; rule($par(c_k)$=“VP→VV+NP”; lrule(c_k)=“NP(原因)→NN+NN”; rule($par(c_k)$=“VP(调查)→VV+NP”;
Other features	scf=“VP→VV+NP”; has_NT=“No”; htr-p=“NP↓NN”; htr-w=“NP↓原因”; cct-c=“VV→NP”; cct-w=“VV→调查”;

Table 8.2: An example of the features used in our SRL system.

8.3 Partial Parsing Based Semantic Chunking

8.3.1 Motivation

In English SRL research, there have been some attempts at relaxing the necessity of using syntactic information derived from full parse trees and a second strategy based on partial parsing has been proposed and well evaluated by the CoNLL 2004 shared task. Previous work on English suggests that even much better labeling performance has been achieved by full parsing based SRL systems, partial parsing based SRL systems can still enhance their performance [Surdeanu et al., 2007]. Most existing systems for automatic Chinese SRL make use of a full syntactic parse of the sentence in order to define argument boundaries and to extract relevant information for training

	保险	公司	已	为	三峡	工程	提供	保险	服务
POS	[NN	NN]	[AD]	[P]	[NR]	[NN]	[VV]	[NN	NN]
SYN	[NP]		[ADVP]	[PP	NP	NP]	[VP]	[NP]	
SEM	B-A0		B-AM-ADV	B-A2	I-A2	I-A2	B-V	B-A1	

Figure 8.3: An example of the definition of semantic chunks: *The insurance company has provided insurance services for the Sanxia Project.*

classifiers to disambiguate between role labels. Though better understanding of SRL with shallow parsing on English is achieved by CoNLL 2004 shared task [Carreras and Màrquez, 2004], little is known about how these SRL methods perform on Chinese and how different they are with the full parsing based ones.

In this chapter, we implement a lightweight semantic chunker based on our discriminative POS tagger and chunker described in Chapter 5. We evaluate this shallow semantic chunker to show how well it can resolve Chinese SRL. More interestingly, we empirically show how different it is from a full parsing based system. The diversity is important not only for understanding the difficulties of Chinese semantic processing and but also for integrating heterogeneous systems designed with different views.

8.3.2 System Architecture

On the basis of partial parsing, i.e. text chunking, we implement a lightweight systems which solve SRL as a semantic chunking problem. SRL is a complex task which has to be decomposed into a number of simpler decisions and tagging schemes in order to be addressed by learning techniques. Our strategy to perform role labeling over flat syntactic chunks by defining semantic chunks which do not overlap nor embed using IOB2 representation [Ramshaw and Marcus, 1995]. A number of words are analyzed as *non*-chunks in the text chunking stage but still make up semantic roles. To recovery all semantic roles, our semantic chunker takes these *outside* words as single word chunks and assign the syntactic chunk label *O* to them.

We define semantic chunk labels based on shallow syntactic chunks in the same way as used in the full parsing based system. The definition of semantic chunks is illustrated in Line *SEM* in Figure 8.3. For example, the noun phrase "保险公司/the insurance company," as a whole, is the *Patient* of the verb "提供/provide," as a result, its semantic tag is defined as *B-A1*.

By predicting the positional semantic role labels over syntactic chunks, the problem of SRL can be regarded as a sequence labeling task. In our experiments, we first perform POS tagging and chunking using our discriminative taggers introduced in the last chapter. A semantic chunker then extracts rich shallow features over the automatically segmented syntactic chunks and use these features as clues to predict the semantic chunk labels.

8.3.3 Features

Similar to full parsing based methods, key to the success of shallow semantic chunking is to carefully design good features. To conveniently illustrate, we denote a token chunk with a fixed context $c_{k-2}c_{k-1}c_kc_{k+1}c_{k+2}$, where each chunk $c_k = w_{k_s}...w_h/p_h...w_{k_e}$ and w_h is the head word of c_k. We denote the context of current predicate as $c_{v-2}c_{v-1}c_vc_{v+1}c_{v+2}$, where $c_v = w^v$ is the predicate in question.

- *Chunk type features*: c_{k-2}, c_{k-1}, c_k, c_{k+1}, c_{k+2}, $c_{k-2}c_{k-1}$, $c_{k-1}c_k$, c_kc_{k+1}, $c_{k+1}c_{k+2}$,
- *word features*: w_{k-2_s}, w_{k-1_s}, w_{k_s}, w_{k+1_s}, w_{k+2_s}, $w_{k-1_s}w_{k_s}$, $w_{k_s}w_{k+1_s}$, w_{k-2_e}, w_{k-1_e}, w_{k_e}, w_{k+1_e}, w_{k+2_e}, $w_{k-1_e}w_{k_e}$, $w_{k_e}w_{k+1_e}$, w^v, w^p.
- *POS chain*: sequential containers of each word's POS tag. For example, this feature for "保险服务" is "NN_NN." We include the POS tag chain of c_{k-1}, c_k and c_{k+1} as three features.
- *Length*: the number of words in a chunk.
- *Position* (posi): The position of c_k with respect to the predicate. It has three values as *before*, *after* and *here*.
- posi+w_h, posi+p_h, posi+w^v, posi+w_h+w^v, posi+p_h+w^v.
- *Path*: A flat path feature is defined as a chain of base phrases between the token and the predicate. At both ends, the chain is terminated with the POS tags of the predicate and the head word of the token. Three path features are included: $path(c_{k-1}, w^v)$, $path(c_k, w^v)$ and $path(c_{k+1}, w^v)$.
- V|De path: A sequential container of POS tags of verbal words and "的". Three V|De path features are included: $path(c_{k-1}, w^v)$, $path(c_k, w^v)$ and $path(c_{k+1}, w^v)$.
- *Distance*: we have two notions of distance. The first is the distance of the token from the predicate as a number of base phrases, and the second is the same distance as the number of VP chunks.

- Verb class: vclass, vclass+posi, vclass+posi+w_h, vclass+posi+p_h.
- *Number of predicates*: the number of predicates in the sentence.
- The context of w_v: c_{v-2}+c_{v-1}+c_v+c_{v+1}+c_{v+2}, p^v_{-1}, p^v_{+1}.

8.4 Experiments and Analysis

8.4.1 Setting

The CPB annotation is widely used as benchmark data to evaluate Chinese SRL systems. To facilitate comparison with previous work, we conduct some experiments on the CPB 1.0 and CTB 5.0, using the data setting of [Ding and Chang, 2008; Sun, 2010b; Xue, 2008; Zhuang and Zong, 2010]. Before the labeling of predicates, a SRL system should detect target predicates. The previous SRL evaluations, i.e. CoNLL 2004 and 2005 shared tasks, assume that this information is already known to isolate the role labeling problem. We also follow this setting. Table 8.3 shows the statistics of our experimental settings.

Data set	Task	CTB files	# of sent.
Training	Sem+Synt	81-899	8828
	Synt	1001-1151	8420
Devel.	Sem+Synt	301-325	561
Test	Sem+Synt	1-40, 900-931	995

Table 8.3: Training, development and test data on CTB 5.0

To resolve the classification problem, we use an open source linear classifier LIBLINEAR[1]. Note that a crucial aspect in local scoring of SRL is the representation of candidates with features, rather than the particular choice of classification algorithm. To fairly compare the performance of the classification system and the chunking system, we use the SVM-HMM[2] to train a first order sequence labeling model, since the learning algorithms we chose in LIBLINEAR and implemented in SVM-HMM are both based on the max-margin criterion. We use the basic linear model of SVM-HMM without applying any kernel function. However, SVM-HMM is a computationally expensive toolkit, which makes it not suitable for large-scale training data. We also

[1]http://www.csie.ntu.edu.tw/~cjlin/liblinear/.
[2]http://www.cs.cornell.edu/People/tj/svm_light/svm_hmm.html.

report experimental results by using a CRF toolkit, Crfsgd[1], which implements a stochastic gradient descent algorithm for fast and scalable parameter estimation.

8.4.2 Main Results

8.4.2.1 Rich Syntactic Features Are Helpful

Test	Task	Parser	P	R	F/A
Constituent classification	AI	CTB(−pruning)	98.53%	97.91%	98.22
system	SRC	CTB(−pruning)	- -	- -	94.94%
	AI+SRC	CTB(−pruning)	93.70%	93.11%	93.41
	AI	CTB(+pruning)	98.14%	97.83%	97.99
	SRC	CTB(+pruning)	- -	- -	94.93%
	AI+SRC	CTB(+pruning)	93.34%	93.04%	93.19
	AI	Berkeley	84.82%	76.42%	80.40
	SRC	Berkeley	- -	- -	93.35%
	AI+SRC	Berkeley	80.85%	72.84%	76.64
[Ding and Chang, 2008]	SRC	CTB	- -	- -	94.68%
[Xue, 2008]	AI+SRC	CTB	93.0%	91.0%	92.0
[Zhuang and Zong, 2010]	AI+SRC	Berkeley	80.75%	70.98%	75.55

Table 8.4: Performance of the full parsing based classification system on the test data.

Table 8.4 summarizes precision, recall and f-score of AI, SRC and the whole task (AI+SRC) of our full parsing based constituent classification system. The last three lines show the best published SRL performance respectively reported in [Ding and Chang, 2008; Xue, 2008; Zhuang and Zong, 2010]. Other lines show the performance of our system. The syntactic annotation of the CTB project also includes information about empty categories. Modern statistical parsers such as Collins, Charniak and Berkeley parsers ignore this type of linguistic information. When the Penn Treebank data is prepared to train a parser, a heuristic procedure is usually performed to delete empty categories and its associated redundant ancestors. Because it is unclear whether or not heuristic pruning is used in [Xue, 2008] and [Ding and Chang, 2008], we report results of our system on both setting, which is distinguished with −*pruning* and +*pruning*.

These results give a significant improvement over previous systems due to the new features. When gold parses are available, the system achieves a very accurate result, over 93, which is significantly better than the best published result of 92. When imperfect parses are used, which is the realistic situation, the accuracy drops

[1]http://leon.bottou.org/projects/sgd

dramatically, from 93+ down to 76.64. The decline of the precision and recall is not balanced. If an argument is not bracketed as a phrase by a parser, the classification system cannot recovery it since the classification system only take constituents as argument candidates. This strategy harms the recall very much.

The syntactic information affects the recognition of the arguments more than the semantic classification of these arguments. When gold syntactic information is accessible, most arguments (about 98%) can be correctly recognized. But when an automatic parser is applied, only 76.42% arguments are found. The decline of the performance of the SRC sub-task, from 94.9 to 93.35, is relatively small. That indicates the syntactic information for SRC is not as important as for AI. Ding and Chang [2008] claim that hierarchical classification is helpful for SRC. However, delicately designed features are more important and our experiments suggest that by using rich features, a better SRC classifier can be directly trained without using the hierarchical architecture.

8.4.2.2 Semantic Chunking Is Helpful

Table 8.5 summarizes performance of different full parsing based systems. These results shows the helpfulness of semantic chunking. The overall f-score goes from 76.64 up to 77.77 by using a first order Markov model. In general, both max-margin and CRF models work well and achieve a similar overall performance. The underlying learning algorithm of Crfsgd is a fast yet scalable one, which can be easily applied to deal with large scale problems. This suggests that even when more annotated data is available, semantic chunking is still feasible. We think agreement-based semi-supervised learning can take advantage of this point.

Chinese parsing has been shown a very challenging task. Bikel parser[1] [Bikel, 2004], which implements the Collins' model, is a very good system for English processing, and achieves earlier success on Chinese processing. However, it does not lead to good performance for Chinese SRL. In particular, it loses an f-score of over 5 points. Although our chunking system can tolerant some bracketing errors in the full parsing stage, it is still very much limited to the quality of full parsing because it only takes into account the c-commanders.

[1] http://www.cis.upenn.edu/~dbikel/software.html

Test	Parser	P	R	F
Constituent classification system	Berkeley	80.85%	72.84%	76.64
LIBLINEAR(Local)	Berkeley	79.97%	74.10%	76.92
SVM-HMM (1-order)	Berkeley	81.03%	74.42%	77.59
Crfsgd (1-order)	Berkeley	82.10%	73.87%	77.77
Crfsgd (1-order)	Bikel	75.26%	70.00%	72.54

Table 8.5: Performance of full parsing based SRL systems on the test data.

Test	Tagger+Chunker	P	R	F
Syntactic Chunking	Berkeley	86.97%	86.22%	86.59
	Ours	87.92%	86.62%	87.27
Semantic Chunking	Berkeley	78.43%	67.05%	72.29
	Ours	77.78%	65.87%	71.33

Table 8.6: Performance of the partial parsing based semantic chunking system on the test data.

8.4.3 Two-fold Effect of Parsing in SRL

8.4.3.1 Impact on Different Sub-tasks

The main effect of parsing in SRL is two-fold. First, grouping words into constituents, parsing helps find argument candidates. Second, parsers provide semantic classifiers plenty of syntactic information, not to only recognize arguments from all candidate constituents but also to classify their detailed semantic types. We empirically analyze each effect in turn based on our constituent classification system.

In AI, full parsing is very important for both grouping words and classification. Table 8.7 summarizes relevant experimental results on the development data. Line 2 is the AI performance when gold candidate boundaries and word features are used; Line 3 is the performance with additional syntactic features. Line 4 shows the performance by using automatic parses generated by Berkeley parser. We can see that: 1) word features only cannot train good classifiers to identify arguments; 2) it is very easy to recognize arguments with good enough syntactic parses; 3) there is a severe performance decline when automatic parses are used. The third observation is a similar conclusion in English SRL. However this problem in Chinese is much more serious due to the state-of-the-art of Chinese parsing.

Information theoretic criteria are popular criteria in variable selection [Guyon and Elisseeff, 2003]. We use the empirical mutual information between each feature

Task	Parser	Bracket	Features	P	R	F/A
AI	- -	CTB	Word	80.39%	85.68%	82.95
	CTB	CTB	Word+Synt	98.36%	97.98%	98.17
	Berkeley	Berkeley	Word+Synt	84.38%	75.59%	79.74
SRC	- -	CPB	Word	- -	- -	93.87%
	CTB	CPB	Word+Synt	- -	- -	95.71%
	Berkeley	CPB	Word+Synt	- -	- -	94.23%

Table 8.7: Classification performance on different tasks on the development data.

template and the target to roughly rank the importance of features:

$$I(X, Y) = \sum_{x \in X, y \in Y} p(x, y) \log \frac{p(x, y)}{p(x)p(y)}$$

Table 8.8 shows the ten most useful features in AI. We can see that the most important features all based on full parsing information.

Rank	Feature	Rank	Feature
1	cct-w	2	‡ w_h+posi+w^v
3	htr-w	4	htr-p
5	$path(c_k, w^v)$	6	‡ w_h+w^v
7	$cpath(c_k, w^v)$	8	cct-c
9	$path(c_k, w^v) + w^v$	10	lrule($par(c_k)$)

Table 8.8: Top 10 useful features for AI. ‡ means *word features*.

The second block in Table 8.7 summarizes the SRC performance with gold argument boundaries. Line 5 is the accuracy when word features are used; Line 6 is the accuracy when additional syntactic features are added; The last row is the accuracy when syntactic features used are extracted from automatic parses. We can see that different from AI, word features only can train reasonably good semantic classifiers. The comparison suggests that full parsing is not very important for SRC.

8.4.3.2 Why Word Features Are Effective for SRC?

Table 8.9 shows the ten most useful features in SRC. We can see that two of these ten features are word features (denoted by †). Namely, word features play a more important role in SRC than in AI. Though the other eight features are based on full parsing, four of them (denoted by ‡) use the head word which can be well approximated by word features, according to some language specific properties. The head rules described in [Sun and Jurafsky, 2004] are very popular in Chinese parsing re-

Rank	Feature	Rank	Feature
1	‡cframe+w^v+w_h	2	‡w_h+w^v+position
3	‡w_h+w^v	4	cct-w
5	lrule($par(c_k)$)	6	†w_i+w_j
7	lrule(c_k)	8	‡w_h+Posi
9	†cframe+w^v	10	htr-p

Table 8.9: Top 10 useful features for SRC.

search, such as in [Duan et al., 2007; Zhang and Clark, 2008b]. From these head rules, we can see that head words of most phrases in Chinese are located at the first or the last position. We implement these rules on Chinese Tree Bank and find that 84.12% [1] nodes realize their heads as either their first or last word. Head position suggests that boundary words are good approximation of head word features. If head words have good approximation word features, then it is not strange that the four features denoted by ‡ can be effectively represented by word features. For example, when two new approximation features, cframe+w^v+w_i and cframe+w^v+w_j, are added, the word based SRC can achieve an accuracy of 94.10%, which is very close to a full parsing armed one.

8.4.4 Predicate Frequency Factor

Each of our system, as well as most PropBank-style labeling systems, trains one model for all predicates. Even when a predicate in question never appears in the training data, the system can still try to find its arguments, though the accuracy is much lower. Figure 8.4 plots the changes of f-scores of different tasks relative to the word frequency. The influence of the word frequency on the training data to different sub-tasks, AI or SRC, is consistent. When the classifiers do not see any examples of a particular predicate, the predication is very inaccurate. The good news is that even when a predicate only appears a few times, say once or twice, the prediction will be significantly better. This observation is helpful to construct domain-specific SRL systems. For a new domain, linguists can annotate several instances for domain-specific predicates, and that will help a lot.

[1]This statistics excludes all empty categories in CTB.

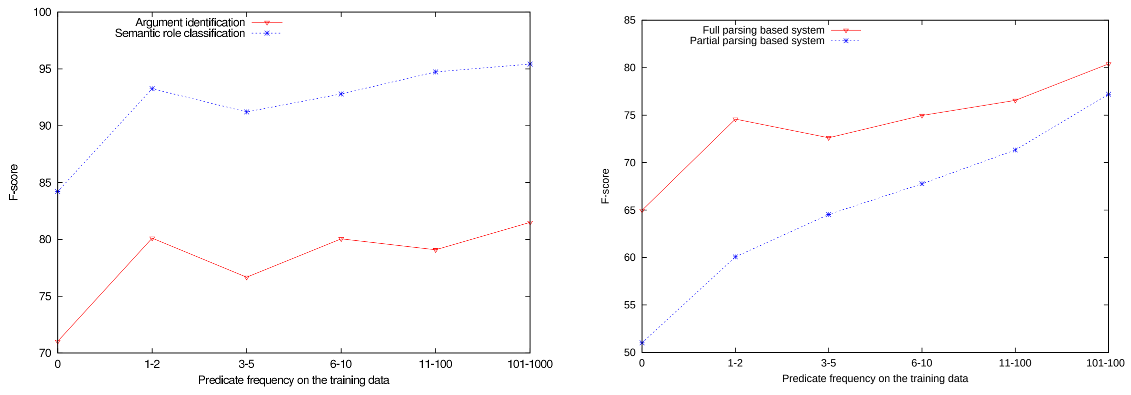


Figure 8.4: F-scores of different tasks and different systems relative to the predicate frequency on the training data.

8.5 Comparative Analysis

8.5.1 Full Parsing is Necessary

These results indicate the necessity of rich or deep syntactic analysis for Chinese SRL. When the semantic chunking is built on a discriminative tagger and chunker, the final SRL f-score is only 71.33. There is a significant gap between the full and partial parsing based systems. An interesting phenomenon is that though the overall quality of chunks provided by the Berkeley parser is lower, they are more useful for semantic chunking. This observation shows the helpfulness of the implicitly encoded structural information in the parser-style chunking results.

Table 8.10 is the overall performance of the two systems on the development data set. When we compare the systems based on state-of-the-art parsing systems, we can see a significant gap (about 5.34 points) of the balanced f-score. The gap of the prediction precision is relatively small. The main disadvantage of the partial parsing based system is its weak ability to group words as argument candidates, since little syntactic information, which only captures the boundary information of very small phrases, is used.

	Tagger+Parser	P	R	F
Full parsing based	Berkeley	81.77%	73.23%	77.27
Partial parsing based	Ours	78.09%	66.67%	71.93

Table 8.10: Performance of different semantic chunking systems on the development data.

Though the partial parsing based system is much weaker, we are still interested in whether or not the full parsing based system is better all the time. To do this, we present a comparative evaluation of our full and partial parsing based semantic chunking systems. Especially, we emphasize the complementary strengths of the partial parsing based system to the full parsing based one. This analysis will help enhance a strong full parsing based SRL system with the help of a partial parsing based shallow system.

8.5.2 A Comparison of the Recall

The predictive power of the two systems are different: There are a number of arguments that can be correctly recognized by one but not another. This point makes it possible to further improve the system recall by combining the full and partial parsing based systems. Especially, the much weaker system, i.e. the partial parsing based system, can find a comparable number of arguments are are missed by the stronger system. Figure 8.5 measures the recall for different systems relative to the length (in bins of size 5: 1-5, 6-10, etc.) of the arguments, while Figure 8.6 measures the recall relative to the distance (in bins of size 5: 1-5, 6-10, etc.) between arguments and predicates. By *distance*, we mean how many words locates between an argument and its corresponding predicate. The overall recall of the full parsing based system is significantly higher, however, there are still a number of arguments that are only recognized by the partial parsing based system. This part is respected as the blue parts in both figures.

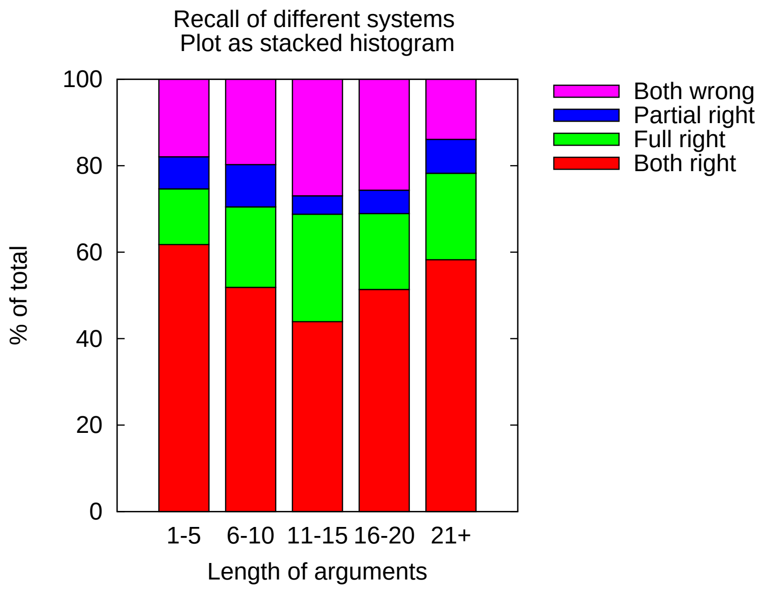


Figure 8.5: Recall of different systems relative to the length of arguments.

It is obvious that long sentences or large phrases are more hard to parsed. As a result, two types of arguments are hard to analyze: (1) *large* arguments that consists of large numbers of words, and (2) *far* arguments that are far away from their corresponding predicates. Figure 8.5 also show the impact of the length of the arguments. When the length goes up, the recall goes down. It is not surprising that arguments are more and more difficult to rightly recognize as the increase of their length. But the performance decline slows up and even stops when the length of arguments is larger than 20 for both full parsing and partial parsing based systems. In other words, some of the arguments that are composed of many words are still relatively easy to be identified. The main reason for this point is that these arguments usually have clear collocation words locating at argument boundaries. Take the phrase below for example.

- 包括/including [*A*1 ⋯⋯/...等/etc.]

The object of the verb "包括/include" has a definite collocation word "等/etc.", and this object is thus easy to be recognized as a *A1*.

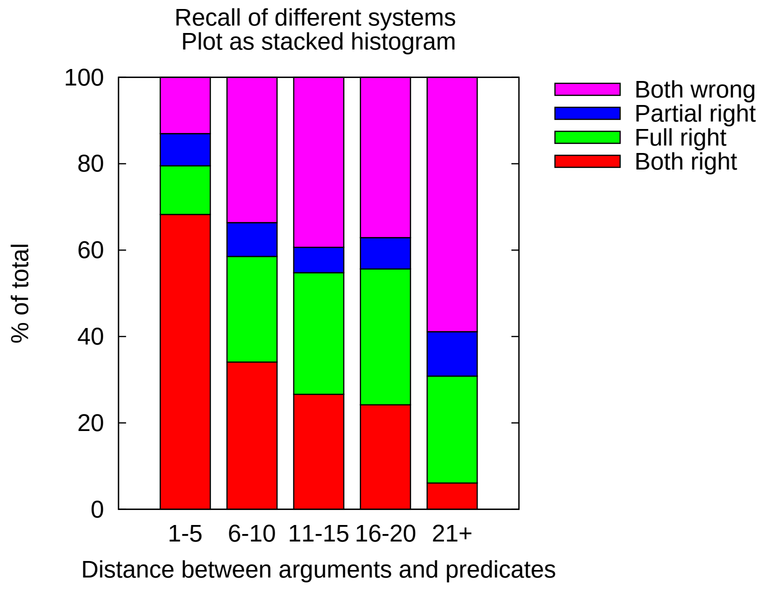


Figure 8.6: Recall of different systems relative to the distance between arguments and predicates.

The sequential tagging algorithms assume a linear chain structure to different language problems. Our partial parsing based system only leverage sequential tagging to resolve the SRL problem. One inherent problem of this system is the weak ability to deal with the so-called far arguments. On the contrary, the full parser can predict a syntactic structure with a global view. The gap between the recall of two systems grows dramatically when the arguments locate far away from the predicates, as shown in Figure 8.6. This observation is different from the observations of large arguments. The gap is more stable in that case. Another difference is that the decline does not stop when the distance is greater than 20.

8.5.3 A Comparison of the Precision

When we focus on the precision of two systems, we are sometimes interested in how many errors that are made by the first system but not by the second one. That to some extent means that the errors made by the first system can be *recovered* with the help of the second system. Denote the number of all tokens that are wrongly predicted by the first system as $\#\{\mathcal{E}_1\}$, and the number of tokens that are wrongly predicted as arguments by both systems as $\#\{\mathcal{E}_1 \cap \mathcal{E}_2\}$. We define recovery rate of the second system to the first system as $\frac{\#\{\mathcal{E}_1\} - \#\{\mathcal{E}_1 \cap \mathcal{E}_2\}}{\#\{\mathcal{E}_1\}}$. Table 8.11 show the error rate and the recovery rate relative to different argument types. This result is very encouraging for system combination, since a large rate of errors can be modified.

Type	Num.	Error rate	Recovery rate
A0	1171	23.74%	68.35%
A1	1612	15.69%	69.17%
A2	172	21.51%	72.97%
A3	11	9.09%	100.00%
A4	7	0.00%	0.00%
AM	1422	13.85%	75.63%

Table 8.11: Error rate and the recovery rate relative to the type of arguments.

8.5.4 Impact of Predicate Frequency

Figure 8.4 plots the changes of f-scores of different chunking systems relative to the word frequency. First, both systems work very badly when they never see a predicate or just see them a few times in the training data. As more and more instances of a particular predicate is available, the role labeling of this predicate is better and better. The amount of training data of a particular predicate influence the full parsing based system less. We think one main reason is that syntactic information significantly abstract the meaning from surface strings, and a semantic processor based on full parsing is thus more robust than the one based on partial parsing. The availability of labeled data of a particular predicate significantly limits the SRL performance. We thus think it is an essential topic to better capture the paradigmatic relations of predicates, e.g. through hierarchical classification of verbs.

8.6 Conclusion and Discussion

In this chapter, we first went deep into the feature engineering problem for Chinese SRL. We then introduced a new method which took either full parses or partial parse as inputs, and detected and classified semantic roles in a chunking way. Our evaluation on the benchmark data showed that the full parsing based new features and new method lead to a significant improvement over the best published individual SRL system. Furthermore, we present a series of empirical analysis to achieve better understanding of Chinese SRL. We hope our analysis is helpful to enhance existing methods and to design new solutions for Chinese SRL.

Our comparative analysis of full and partial parsing based methods emphasize on the complementary strengths of the partial parsing based system to the full parsing based one. Our analysis suggests that Chinese SRL can benefit from the combination of the full and partial parsing based methods. This direction is explored in [Zhuang and Zong, 2010], which leverage a integer linear programming based post-inference to combine the outputs from different systems. If we take different parsers as pre-processing systems, even the same SRL method can provide different labeling results. In their experiments, the combination of different full parsing based systems was helpful, but the further combination with our partial parsing based system was more remarkable. This also confirms our motivation to develop a purely discriminative shallow semantic chunker.

Chapter 9

Conclusions

This chapter provides some brief concluding remarks and discusses topics for future research.

9.1 Summary of the Thesis

This thesis is motivated by the inadequacy of single view approaches in many areas in NLP. We have studied multi-view Chinese language processing, including word segmentation, POS tagging, syntactic parsing, and semantic role labeling. We consider three situations of multiple views in statistical NLP: (1) Heterogeneous methods have been designed for a given problem; (2) Heterogeneous annotation data, which could be either different in annotation schemes or in formalisms, is available to train single systems; (3) Heterogeneous machine learning paradigms, which could be either supervised or unsupervised, are applicable. Table 9.1 lists all the problems and heterogeneous views we have investigated. Each discussed item is one evidence for the primary argument, that is, *learning language structures could benefit from multiple, heterogeneous views.*

- For word segmentation, we first present a comparative study of two state-of-the-art segmentation methods. Inspired by the diversity of the character-based and word-based views, we designed a novel stacked sub-word tagging model for joint word segmentation and POS tagging, which is robust to integrate different models, even models trained on heterogeneous annotations.
- For POS tagging, we introduced two improvements: (1) integrating chart parsing results to better capture syntagmatic relations among words and (2) inte-

	Model	Annotation		Learning paradigm
		Scheme	Formalism	
Word segmentation	√	√		√
POS tagging	√	√		√
Syntactic parsing		√	√	√
Semantic role labeling	√			

Table 9.1: The tasks and their corresponding multi-views investigated in the thesis.

grating word clusters acquired from unlabeled data to better capture paradigmatic relations among words.

- For syntactic parsing, we focused on different linguistic annotations, including both the representation formalism and the annotation scheme. We present a comparative analysis for generative PCFG-LA constituency parsing and discriminative graph-based dependency parsing. To explore the diversity of parsing in different formalisms, we introduced a Bagging model to effectively enhance dependency parsing. We also explored heterogenous treebanks to improve constituency parsing via a reranking model.

- Our work on SRL focused on improving the full parsing method with linguistically rich features and a chunking strategy. Furthermore, we developed a partial parsing based semantic chunking method, which has complementary strengths to the full parsing based method.

- Finally, we introduced a feature induction method to improve supervised a word segmenter and various syntactic processing systems via harvesting string and word knowledge from unlabeled data.

Multi-view learning can be advantageous when compared to learning with only a single view especially when learners built on different views are distinct and diverse enough. The impact of multi-views mainly stands from the diversity between learners, while it is less important whether the diversity is caused by using multiple computational models, by training on heterogeneous data, or by implementing supervised or unsupervised learning paradigms. Our work has shown that view integration benefits language processing across a wide range of conditions.

An exciting but non-obvious fact is that even in cases that one learner is much weaker than another learner, it can still enhance the stronger one if it is relevant and increases the diversity. According to our experiments, as well as some others, a slightly weaker word-based segmenter can help a character-based segmenter (Chapter

2 and 3), a weaker chart parsing based POS tagger can help a sequential tagger (Chapter 5), and a significantly weaker partial parsing based SRL system can help a strong full parsing based system [Zhuang and Zong, 2010].

Finally, supervised and unsupervised learning paradigms usually work in very different ways and there is no guarantee that outputs of unsupervised learners can be directly compared to human labeled data. Nevertheless, knowledge acquired in an unsupervised manner can still help supervised systems, if it is relevant to the task. In our experiments, the knowledge about how independently a string is used is not directly related to word boundaries but can enhance a strong supervised segmenter; word clusters that are only roughly related to the paradigmatic lexical relations can enhance syntactic parsing in different levels.

9.2 Ideas for Future Work

During the course of research, several ideas emerged that could not have been explored in this thesis. They could be fruitful to revisit some of these ideas in future work.

- Though we only considered Chinese language processing in this study, the idea to analyze and combine different views is very general in NLP. One natural idea for future work would be to apply our multi-view processing methods to other languages.

- We exemplified the advantages of multi-view learning through *system integration*. Many other topics, such as agreement/disagreement-based semi-supervised learning and active learning, could also benefit from investigation of multiple views.

- Our focus to integrate heterogeneous views for NLP is very closed to *ensemble learning*, in the sense that both employ multiple learners and combine their predictions. There are a number of other well studied ensemble learning methods, such as boosting [Schapire, 1990], error-correcting output codes [Dietterich and Bakiri, 1995] and random subspace method [Ho, 1998]. These algorithms may also benefit multi-view language structure learning.

- The key point of our post-inference-based paradigm for view integration is to re-predict (or select) based on less accurate outputs from individual systems. There are several considerable way to represent the output of a base system. The simplest way is to produce the *best* or *n-best* predictions for next level processing. A more interesting way is to compactly represent possible predictions. For example, the output of word segmentation and POS tagging can be represented as a word lattice rather that a sequence, and the output of constituency parsing can be represented as a forest rather than a tree. We think the inference over a large search space may lead to further improvements for view integration.

The success of our investigation on learning Chinese language structures supports the multi-view processing idea. Ultimately, we believe that many other tasks as well as tasks for other languages can be successfully improved by integrating multi-views. This dissertation has been an illustration of this claim.

References

Galen Andrew. 2006. A hybrid markov/semi-markov conditional random field for sequence segmentation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 465–472. Association for Computational Linguistics, Sydney, Australia. 13, 15, 29

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 86–90. Association for Computational Linguistics, Montreal, Quebec, Canada. URL `http://www.aclweb.org/anthology/P98-1013`. 134

Daniel M. Bikel. 2004. A distributional analysis of a lexicalized statistical parsing model. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 182–189. Association for Computational Linguistics, Barcelona, Spain. 149

Daniel M. Bikel and David Chiang. 2000. Two statistical parsing models applied to the Chinese treebank. In *Proceedings of the second workshop on Chinese language processing: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics - Volume 12*, CLPW '00, pages 1–6. Association for Computational Linguistics, Stroudsburg, PA, USA. URL `http://dx.doi.org/10.3115/1117769.1117771`. 78

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 89–97. Coling 2010 Organizing Committee, Beijing, China. URL `http://www.aclweb.org/anthology/C10-1011`. 82

Leo Breiman. 1996a. Bagging predictors. *Machine Learning*, 24(2):123–140. 26

Leo Breiman. 1996b. Stacked regressions. *Machine Learning*, 24:49–64. URL `http://portal.acm.org/citation.cfm?id=230972.230977`. 33

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18:467–479. URL http://portal.acm.org/citation.cfm?id=176313.176316. 116

Aljoscha Burchardt, Katrin Erk, Anette Frank, Andrea Kowalski, Sebastian Pad ó , and Manfred Pinkal. 2006. The salsa corpus: a german corpus resource for lexical semantics. In *In Proceedings of LREC 2006*. 134

Aoife Cahill, Mairead Mccarthy, Josef Van Genabith, and Andy Way. 2002. Automatic annotation of the penn treebank with lfg f-structure information. In *Proceedings of the LREC Workshop on Linguistic Knowledge Acquisition and Representation: Bootstrapping Annotated Language Data, Las Palmas, Canary Islands*, pages 8–15. 54

Xavier Carreras, Michael Collins, and Terry Koo. 2008. Tag, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 9–16. Coling 2008 Organizing Committee, Manchester, England. URL http://www.aclweb.org/anthology/W08-2102. 2

Xavier Carreras and Lluís Màrquez. 2004. Introduction to the conll-2004 shared task: Semantic role labeling. In Hwee Tou Ng and Ellen Riloff, editors, *HLT-NAACL 2004 Workshop: Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*, pages 89–97. Association for Computational Linguistics, Boston, Massachusetts, USA. 145

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the first conference on North American chapter of the Association for Computational Linguistics*. 2, 81, 83

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 173–180. Association for Computational Linguistics, Ann Arbor, Michigan. 81, 105

Wanxiang Che, Zhenghua Li, Yongqiang Li, Yuhang Guo, Bing Qin, and Ting Liu. 2009. Multilingual dependency-based syntactic and semantic parsing. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*

(CoNLL 2009): Shared Task, pages 49–54. Association for Computational Linguistics, Boulder, Colorado. URL http://www.aclweb.org/anthology/W09-1207. 128, 130

Keh-Jiann Chen and Shing-Huan Liu. 1992. Word identification for mandarin Chinese sentences. In *Proceedings of the 14th conference on Computational linguistics*, pages 101–107. Association for Computational Linguistics, Morristown, NJ, USA. 13, 14

Wenliang Chen, Yujie Zhang, and Hitoshi Isahara. 2006. An empirical study of Chinese chunking. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 97–104. Association for Computational Linguistics, Sydney, Australia. 123, 126

David Chiang. 2000. Statistical parsing with an automatically-extracted tree adjoining grammar. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ACL '00, pages 456–463. Association for Computational Linguistics, Stroudsburg, PA, USA. URL http://dx.doi.org/10.3115/1075218.1075276. 2

Stephen Clark, James R. Curran, and Miles Osborne. 2003. Bootstrapping pos taggers using unlabelled data. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 49–55. Association for Computational Linguistics, Stroudsburg, PA, USA. URL http://dx.doi.org/10.3115/1119176.1119183. 60

William W. Cohen and Vitor R. Carvalho. 2005. Stacked sequential learning. In *Proceedings of the 19th international joint conference on Artificial intelligence*, pages 671–676. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. URL http://portal.acm.org/citation.cfm?id=1642293.1642401. 33

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Computational Linguistics*, pages 175–182. Morgan Kaufmann. 110

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 1–8. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W02-1001. 79, 107

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637. 2, 81, 83

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 263–270. Association for Computational Linguistics, Philadelphia, Pennsylvania, USA. URL `http://www.aclweb.org/anthology/P02-1034`. 107

Michael Collins and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31:25–70. URL `http://dx.doi.org/10.1162/0891201053630273`. 81, 105, 107

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 100–110. 1

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *JOURNAL OF MACHINE LEARNING RESEARCH*, 7:551–585. 18

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 600–609. Association for Computational Linguistics, Portland, Oregon, USA. URL `http://www.aclweb.org/anthology/P11-1061`. 73

Thomas G. Dietterich and Ghulum Bakiri. 1995. Solving multiclass learning problems via error-correcting output codes. *J. Artif. Int. Res.*, 2:263–286. URL `http://dl.acm.org/citation.cfm?id=1622826.1622834`. 160

Weiwei Ding and Baobao Chang. 2008. Improving Chinese semantic role classification with hierarchical feature selection strategy. In *Proceedings of the EMNLP 2008*, pages 324–333. Association for Computational Linguistics, Honolulu, Hawaii. 136, 137, 140, 147, 148, 149

Xiangyu Duan, Jun Zhao, and Bo Xu. 2007. Probabilistic models for action-based Chinese dependency parsing. In *ECML '07: Proceedings of the 18th European con-*

ference on Machine Learning, pages 559–566. Springer-Verlag, Berlin, Heidelberg. 142, 152

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: an exploration. In *Proceedings of the 16th conference on Computational linguistics - Volume 1*, COLING '96, pages 340–345. Association for Computational Linguistics, Stroudsburg, PA, USA. URL `http://dx.doi.org/10.3115/992628.992688`. 2, 98, 99

Thomas Emerson. 2005. The second international Chinese word segmentation bakeoff. In *In Proceedings of the Second SIGHAN Workshop on Chinese Language Processing*, pages 123–133. 21

Haodi Feng, Kang Chen, Xiaotie Deng, and Weimin Zheng. 2004. Accessor variety criteria for Chinese word extraction. *Computational Linguistics*, 30:75–93. 64, 72

Jianfeng Gao, Galen Andrew, Mark Johnson, and Kristina Toutanova. 2007. A comparative study of parameter estimation methods for statistical natural language processing. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 824–831. Association for Computational Linguistics, Prague, Czech Republic. URL `http://www.aclweb.org/anthology/P07-1104`. 107

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28:245–288. URL `http://dx.doi.org/10.1162/089120102760275983`. 134, 136, 137

Jes ú s Gim é nez and Llu í s M à rquez. 2004. Svmtool: A general pos tagger generator based on support vector machines. In *In Proceedings of the 4th International Conference on Language Resources and Evaluation*, pages 43–46. 79

Andrew Gordon and Reid Swanson. 2007. Generalizing semantic role annotations across syntactically similar verbs. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 192–199. Association for Computational Linguistics, Prague, Czech Republic. 136

Isabelle Guyon and André Elisseeff. 2003. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182. 150

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the 13th Conference on Computational Natural Language Learning (CoNLL-2009), June 4-5*. Boulder, Colorado, USA. 82, 137

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2011. Incremental joint pos tagging and dependency parsing in chinese. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1216–1224. Asian Federation of Natural Language Processing, Chiang Mai, Thailand. URL http://www.aclweb.org/anthology/I11-1136. 100, 101

John Henderson and Eric Brill. 1999. Exploiting diversity in natural language processing: Combining parsers. In *In Proceedings of the Fourth Conference on Empirical Methods in Natural Language Processing*, pages 187–194. 97

Tin Kam Ho. 1998. The random subspace method for constructing decision forests. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20:832–844. URL http://dl.acm.org/citation.cfm?id=284980.284986. 160

Julia Hockenmaier and Mark Steedman. 2007. Ccgbank: A corpus of ccg derivations and dependency structures extracted from the penn treebank. *Computational Linguistics*, 33(3):355–396. 54

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-08: HLT*, pages 586–594. Association for Computational Linguistics, Columbus, Ohio. URL http://www.aclweb.org/anthology/P/P08/P08-1067. 81

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1077–1086. Association for Computational Linguistics, Uppsala, Sweden. URL http://www.aclweb.org/anthology/P10-1110. 78, 82, 84

Zhongqiang Huang, Vladimir Eidelman, and Mary Harper. 2009. Improving a simple bigram hmm part-of-speech tagger by latent annotation and self-training. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 213–216. Association for Computational Linguistics, Boulder, Colorado. URL http://www.aclweb.org/anthology/N/N09/N09-2054. 78, 79, 84, 86

Zhongqiang Huang, Mary Harper, and Wen Wang. 2007. Mandarin part-of-speech tagging and discriminative reranking. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1093–1102. Association for Computational Linguistics, Prague, Czech Republic. URL http://www.aclweb.org/anthology/D/D07/D07-1117. 35, 78, 79, 84, 86

Rebecca Hwa, Miles Osborne, Anoop Sarkar, and Mark Steedman. 2003. Corrected co-training for statistical parsers. In *In ICML-03 Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, pages 95–102. 1

Wenbin Jiang, Liang Huang, and Qun Liu. 2009. Automatic adaptation of annotation standards: Chinese word segmentation and pos tagging – a case study. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 522–530. Association for Computational Linguistics, Suntec, Singapore. URL http://www.aclweb.org/anthology/P/P09/P09-1059. 13, 15, 34, 51, 55, 57, 66, 84

Wenbin Jiang, Liang Huang, Qun Liu, and Yajuan Lü. 2008a. A cascaded linear model for joint Chinese word segmentation and part-of-speech tagging. In *Proceedings of ACL-08: HLT*, pages 897–904. Association for Computational Linguistics, Columbus, Ohio. URL http://www.aclweb.org/anthology/P/P08/P08-1102. 32, 41, 45

Wenbin Jiang, Haitao Mi, and Qun Liu. 2008b. Word lattice reranking for Chinese word segmentation and part-of-speech tagging. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 385–392. Coling 2008 Organizing Committee, Manchester, UK. URL http://www.aclweb.org/anthology/C08-1049. 32, 41, 45

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430. Association for Computational Linguistics, Sapporo, Japan. 2

Reinhard Kneser and Hermann Ney. 1993. Improved clustering techniques for class-based statistical language modeling. In *In Proceedings of the European Conference on Speech Communication and Technology (Eurospeech)*. 116

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL-08: HLT*, pages 595–603. Association for Computational Linguistics, Columbus, Ohio. URL http://www.aclweb.org/anthology/P/P08/P08-1068. 61, 62, 71, 116

Canasai Kruengkrai, Kiyotaka Uchimoto, Jun'ichi Kazama, Yiou Wang, Kentaro Torisawa, and Hitoshi Isahara. 2009. An error-driven word-character hybrid model for joint Chinese word segmentation and pos tagging. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 513–521. Association for Computational Linguistics, Suntec, Singapore. URL http://www.aclweb.org/anthology/P/P09/P09-1058. 33, 41, 45

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. URL http://portal.acm.org/citation.cfm?id=645530.655813. 67

Thomas Lavergne, Olivier Cappé, and François Yvon. 2010. Practical very large scale CRFs. pages 504–513. URL http://www.aclweb.org/anthology/P10-1052. 46, 85

Roger Levy and Christopher Manning. 2003. Is it harder to parse Chinese, or the Chinese treebank? In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 439–446. Association for Computational Linguistics, Stroudsburg, PA, USA. URL http://dx.doi.org/10.3115/1075096.1075152. 78, 79

Zhenghua Li, Min Zhang, Wanxiang Che, Ting Liu, Wenliang Chen, and Haizhou Li. 2011. Joint models for Chinese pos tagging and dependency parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1180–1191. Association for Computational Linguistics, Edinburgh, Scotland, UK. URL http://www.aclweb.org/anthology/D11-1109. 78, 79, 82, 84, 87, 128

Zhongguo Li and Maosong Sun. 2009. Punctuation as implicit annotations for Chinese word segmentation. *Computational Linguistics*, 35:505–512. URL http://dx.doi.org/10.1162/coli.2009.35.4.35403. 72

Percy Liang, Michael Collins, and Percy Liang. 2005. Semi-supervised learning for natural language. In *Master' s thesis, MIT*. 116

Percy Liang, Hal Daumé, III, and Dan Klein. 2008. Structure compilation: trading structure for features. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 592–599. ACM, New York, NY, USA. URL http://doi.acm.org/10.1145/1390156.1390231. 16

Shasha Liao and Ralph Grishman. 2011. Using prediction from sentential scope to build a pseudo co-testing learner for event extraction. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 714–722. Asian Federation of Natural Language Processing, Chiang Mai, Thailand. URL http://www.aclweb.org/anthology/I11-1080. 1

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 2*, pages 768–774. Association for Computational Linguistics, Montreal, Quebec, Canada. URL http://www.aclweb.org/anthology/P98-2005. 73

Dekang Lin and Xiaoyun Wu. 2009. Phrase clustering for discriminative learning. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1030–1038. Association for Computational Linguistics, Suntec, Singapore. URL http://www.aclweb.org/anthology/P/P09/P09-1116. 61

Yudong Liu and Anoop Sarkar. 2007. Experimental evaluation of LTAG-based features for semantic role labeling. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 590–599. Association for Computational Linguistics, Prague, Czech Republic. URL http://www.aclweb.org/anthology/D/D07/D07-1062. 136

Andre Martins, Noah Smith, and Eric Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of the Joint Conference of* *the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 342–350. Association for Computational Linguistics, Suntec, Singapore. URL http://www.aclweb.org/anthology/P/P09/P09-1039. 83

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic cfg with latent annotations. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 75–82. Association for Computational Linguistics, Stroudsburg, PA, USA. URL http://dx.doi.org/10.3115/1219840.1219850. 81

Ryan McDonald. 2006. *Discriminative learning and spanning tree algorithms for dependency parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, USA. AAI3225503. 83, 97

Ryan McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 122–131. Association for Computational Linguistics, Prague, Czech Republic. URL http://www.aclweb.org/anthology/D/D07/D07-1013. 82

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530. Association for Computational Linguistics, Vancouver, British Columbia, Canada. 2

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. The nombank project: An interim report. In *In Proceedings of the NAACL/HLT Workshop on Frontiers in Corpus Annotation*. 134

Scott Miller, Jethran Guinness, and Alex Zamanian. 2004. Name tagging with word clusters and discriminative training. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 337–342. Association for Computational Linguistics, Boston, Massachusetts, USA. 61, 62, 71, 116, 117

Yusuke Miyao, Takashi Ninomiya, and Jun ichi Tsujii. 2004. Corpus-oriented grammar development for acquiring a head-driven phrase structure grammar from the penn treebank. In *IJCNLP*, pages 684–693. 54

Alessandro Moschitti. 2004. A study on convolution kernels for shallow statistic parsing. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 335–342. Barcelona, Spain. URL http://www.aclweb.org/anthology/P04-1043. 136

Tetsuji Nakagawa and Kiyotaka Uchimoto. 2007. A hybrid approach to word segmentation and pos tagging. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 217–220. Association for Computational Linguistics, Prague, Czech Republic. URL http://www.aclweb.org/anthology/P07-2055. 33

Hwee Tou Ng and Jin Kiat Low. 2004. Chinese part-of-speech tagging: One-at-a-time or all-at-once? word-based or character-based? In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 277–284. Association for Computational Linguistics, Barcelona, Spain. 13, 32, 33

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Comput. Linguist.*, 34:513–553. URL http://dx.doi.org/10.1162/coli.07-056-R1-07-027. 83

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL-08: HLT*, pages 950–958. Association for Computational Linguistics, Columbus, Ohio. URL http://www.aclweb.org/anthology/P/P08/P08-1108. 30, 33, 97

Franz Josef Och. 1999. An efficient method for determining bilingual word classes. In *Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics*, EACL '99, pages 71–76. Association for Computational Linguistics, Stroudsburg, PA, USA. URL http://dx.doi.org/10.3115/977035.977046. 73, 116

Naoaki Okazaki. 2007. CRFsuite: a fast implementation of conditional random fields (CRFs). URL http://www.chokkan.org/software/crfsuite/. 67

Miles Osborne and Jason Baldridge. 2004. Ensemble-based active learning for parse selection. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 89–96. Association for Computational Linguistics, Boston, Massachusetts, USA. 1

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31:71–106. URL `http://dx.doi.org/10.1162/0891201053630264`. 134

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440. Association for Computational Linguistics, Sydney, Australia. 2, 81, 104

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411. Association for Computational Linguistics, Rochester, New York. 2, 78, 81, 104

Sameer Pradhan, Kadri Hacioglu, Wayne Ward, James H. Martin, and Daniel Jurafsky. 2003. Semantic role parsing: Adding semantic structure to unstructured text. In *Proceedings of the Third IEEE International Conference on Data Mining*, ICDM '03, pages 629–. IEEE Computer Society, Washington, DC, USA. URL `http://portal.acm.org/citation.cfm?id=951949.952080`. 137

Vasin Punyakanok, Dan Roth, Wen-tau Yih, and Dav Zimak. 2004. Semantic role labeling via integer linear programming inference. In *Proceedings of Coling 2004*, pages 1346–1352. COLING, Geneva, Switzerland. 136

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In David Yarowsky and Kenneth Church, editors, *Proceedings of the Third Workshop on Very Large Corpora*, pages 82–94. Association for Computational Linguistics, Somerset, New Jersey. 26, 51, 123, 145

Kenji Sagae and Alon Lavie. 2006a. A best-first probabilistic shift-reduce parser. In *Proceedings of the COLING/ACL on Main conference poster sessions*, COLING-ACL '06, pages 691–698. Association for Computational Linguistics, Stroudsburg, PA, USA. URL `http://dl.acm.org/citation.cfm?id=1273073.1273162`. 2

Kenji Sagae and Alon Lavie. 2006b. Parser combination by reparsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, NAACL-Short '06, pages 129–132. Association for Computational Linguistics, Stroudsburg, PA, USA. URL http://portal.acm.org/citation.cfm?id=1614049.1614082. 30, 97, 98

Sunita Sarawagi and William W. Cohen. 2004. Semi-markov conditional random fields for information extraction. In *In Advances in Neural Information Processing Systems 17*, pages 1185–1192. 18

Anoop Sarkar. 2001. Applying co-training methods to statistical parsing. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, NAACL '01, pages 1–8. Association for Computational Linguistics, Stroudsburg, PA, USA. URL http://dx.doi.org/10.3115/1073336.1073359. 1

Robert E. Schapire. 1990. The strength of weak learnability. *Mach. Learn.*, 5:197–227. URL http://dl.acm.org/citation.cfm?id=83637.83645. 160

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 134–141. Association for Computational Linguistics, Stroudsburg, PA, USA. URL http://dx.doi.org/10.3115/1073445.1073473. 123

Libin Shen and Aravind Joshi. 2005. Incremental ltag parsing. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 811–818. Association for Computational Linguistics, Vancouver, British Columbia, Canada. URL http://www.aclweb.org/anthology/H/H05/H05-1102. 2

Libin Shen, Giorgio Satta, and Aravind Joshi. 2007. Guided learning for bidirectional sequence classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 760–767. Association for Computational Linguistics, Prague, Czech Republic. URL http://www.aclweb.org/anthology/P07-1096. 79

Yan Song and Chunyu Kit. 2009. Pcfg parsing with crf tagging for head recognition. In *Proceedings of the CIPS-ParsEval-2009*. 110

Valentin I. Spitkovsky, Daniel Jurafsky, and Hiyan Alshawi. 2010. Profiting from mark-up: Hyper-text annotations for guided parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1278–1287. Association for Computational Linguistics, Uppsala, Sweden. URL http://www.aclweb.org/anthology/P10-1130. 65

Honglin Sun and Daniel Jurafsky. 2004. Shallow semantc parsing of Chinese. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 249–256. Association for Computational Linguistics, Boston, Massachusetts, USA. 136, 142, 151

Weiwei Sun. 2010a. Improving Chinese semantic role labeling with rich syntactic features. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 168–172. Association for Computational Linguistics, Uppsala, Sweden. URL http://www.aclweb.org/anthology/P10-2031. 133

Weiwei Sun. 2010b. Semantics-driven shallow parsing for Chinese semantic role labeling. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 103–108. Association for Computational Linguistics, Uppsala, Sweden. URL http://www.aclweb.org/anthology/P10-2019. 147

Weiwei Sun. 2010c. Word-based and character-based word segmentation models: Comparison and combination. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1211–1219. Coling 2010 Organizing Committee, Beijing, China. URL http://www.aclweb.org/anthology/C10-2139. 12

Weiwei Sun. 2011. A stacked sub-word model for joint Chinese word segmentation and part-of-speech tagging. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1385–1394. Association for Computational Linguistics, Portland, Oregon, USA. URL http://www.aclweb.org/anthology/P11-1139. 31, 67, 84

Weiwei Sun and Zhifang Sui. 2009. Chinese function tag labeling. In *Proceedings of the 23rd Pacific Asia Conference on Language, Information and Computation*. Hong Kong. 84, 143

Weiwei Sun, Zhifang Sui, and Haifeng Wang. 2008. Prediction of maximal projection for semantic role labeling. In *Proceedings of the 22nd International Conference* *on Computational Linguistics (Coling 2008)*, pages 833–840. Coling 2008 Organizing Committee, Manchester, UK. URL `http://www.aclweb.org/anthology/C08-1105`. 140, 143

Weiwei Sun, Zhifang Sui, Meng Wang, and Xin Wang. 2009a. Chinese semantic role labeling with shallow parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1475–1483. Association for Computational Linguistics, Singapore. URL `http://www.aclweb.org/anthology/D/D09/D09-1153`. 16, 133

Weiwei Sun and Hans Uszkoreit. 2012. Capturing paradigmatic and syntagmatic lexical relations: Towards accurate Chinese part-of-speech tagging. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics. 76, 114

Weiwei Sun and Xiaojun Wan. 2012. Reducing approximation and estimation errors for Chinese lexical processing with heterogeneous annotations. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics. 31

Weiwei Sun, Rui Wang, and Yi Zhang. 2010. Discriminative parse reranking for Chinese with homogeneous and heterogeneous annotations. In *Proceedings of Joint Conference on Chinese Language Processing (CIPS-SIGHAN)*. Beijing, China. URL `http://aclweb.org/anthology/W/W10/W10-4144.pdf`. 103

Weiwei Sun and Jia Xu. 2011. Enhancing Chinese word segmentation using unlabeled data. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 970–979. Association for Computational Linguistics, Edinburgh, Scotland, UK. URL `http://www.aclweb.org/anthology/D11-1090`. 59, 84

Xu Sun, Yaozhong Zhang, Takuya Matsuzaki, Yoshimasa Tsuruoka, and Jun'ichi Tsujii. 2009b. A discriminative latent variable Chinese segmenter with hybrid word/character information. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 56–64. Association for Computational Linguistics, Boulder, Colorado. URL `http://www.aclweb.org/anthology/N/N09/N09-1007`. 13, 15, 20, 29

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The conll-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, CoNLL '08, pages 159–177. Association for Computational Linguistics, Stroudsburg, PA, USA. URL `http://portal.acm.org/citation.cfm?id=1596324.1596352`. 137

Mihai Surdeanu and Christopher D. Manning. 2010. Ensemble models for dependency parsing: Cheap and good? In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 649–652. Association for Computational Linguistics, Los Angeles, California. URL `http://www.aclweb.org/anthology/N10-1091`. 30, 46, 97

Mihai Surdeanu, Lluís Màrquez, Xavier Carreras, and Pere R. Comas. 2007. Combination strategies for semantic role labeling. *J. Artif. Int. Res.*, 29:105–151. URL `http://portal.acm.org/citation.cfm?id=1622606.1622611`. 136, 144

André Filipe Torres Martins, Dipanjan Das, Noah A. Smith, and Eric P. Xing. 2008. Stacking dependency parsers. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 157–166. Association for Computational Linguistics, Honolulu, Hawaii. URL `http://www.aclweb.org/anthology/D08-1017`. 30, 33, 97

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2005. Joint learning improves semantic role labeling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 589–596. Association for Computational Linguistics, Stroudsburg, PA, USA. URL `http://dx.doi.org/10.3115/1219840.1219913`. 136

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 173–180. Association for Computational Linguistics, Stroudsburg, PA, USA. URL `http://dx.doi.org/10.3115/1073445.1073478`. 79

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005a. A conditional random field word segmenter. In *In Fourth SIGHAN Workshop on Chinese Language Processing*. 13, 15, 78

Huihsin Tseng, Daniel Jurafsky, and Christopher Manning. 2005b. Morphological features help pos tagging of unknown words across language varieties. In *The Fourth SIGHAN Workshop on Chinese Language Processing*. 78, 79

Yoshimasa Tsuruoka, Jun'ichi Tsujii, and Sophia Ananiadou. 2009. Fast full parsing by linear-chain conditional random fields. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 790–798. Association for Computational Linguistics, Athens, Greece. URL `http://www.aclweb.org/anthology/E09-1090`. 2

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394. Association for Computational Linguistics, Uppsala, Sweden. URL `http://www.aclweb.org/anthology/P10-1040`. 61, 71

Mengqiu Wang, Kenji Sagae, and Teruko Mitamura. 2006. A fast, accurate deterministic parser for Chinese. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 425–432. Association for Computational Linguistics, Sydney, Australia. URL `http://www.aclweb.org/anthology/P06-1054`. 84

Yiou Wang, Jun'ichi Kazama, Yoshimasa Tsuruoka, Wenliang Chen, Yujie Zhang, and Kentaro Torisawa. 2011. Improving chinese word segmentation and pos tagging with semi-supervised methods using large auto-analyzed data. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 309–317. Asian Federation of Natural Language Processing, Chiang Mai, Thailand. URL `http://www.aclweb.org/anthology/I11-1035`. 57

David H. Wolpert. 1992. Original contribution: Stacked generalization. *Neural Netw.*, 5:241–259. URL `http://dx.doi.org/10.1016/S0893-6080(05)80023-1`. 33

Dekai Wu, Grace Ngai, and Marine Carpuat. 2003. A stacked, voted, stacked model for named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 200–203. URL `http://www.aclweb.org/anthology/W03-0433.pdf`. 33

Fei Xia. 1999. Extracting tree adjoining grammars from bracketed corpora. In *Proceedings of Natural Language Processing Pacific Rim Symposium*, pages 398–403. 54

Jia Xu, Jianfeng Gao, Kristina Toutanova, and Hermann Ney. 2008. Bayesian semi-supervised Chinese word segmentation for statistical machine translation. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 1017–1024. Coling 2008 Organizing Committee, Manchester, UK. URL `http://www.aclweb.org/anthology/C08-1128`. 71

Nianwen Xue. 2003. Chinese word segmentation as character tagging. In *International Journal of Computational Linguistics and Chinese Language Processing*. 13, 33

Nianwen Xue. 2007. Tapping the implicit information for the PS to DS conversion of the Chinese treebank. In *Proceedings of the Sixth International Workshop on Treebanks and Linguistics Theories*. 83, 93

Nianwen Xue. 2008. Labeling Chinese predicates with semantic roles. *Computational Linguistics*, 34:225–255. URL `http://dx.doi.org/10.1162/coli.2008.34.2.225`. 136, 137, 138, 140, 141, 147, 148

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 88–94. Association for Computational Linguistics, Barcelona, Spain. URL `http://www.aclweb.org/anthology-new/W/W04/W04-3212`. 136

Nianwen Xue and Martha Palmer. 2005. Automatic semantic role labeling for Chinese verbs. In *Proceedings of the 19th international joint conference on Artificial intelligence*, pages 1160–1165. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. URL `http://dl.acm.org/citation.cfm?id=1642293.1642479`. 136, 137

Nianwen Xue and Martha Palmer. 2009. Adding semantic roles to the Chinese treebank. *Nat. Lang. Eng.*, 15:143–172. URL `http://portal.acm.org/citation.cfm?id=1520233.1520241`. 134

Nianwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The penn Chinese treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(2):207–238. 104

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *The 8th International Workshop of Parsing Technologies (IWPT2003)*, pages 195–206. 2

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, ACL '95, pages 189–196. Association for Computational Linguistics, Stroudsburg, PA, USA. URL http://dx.doi.org/10.3115/981658.981684. 73

Hui Zhang, Min Zhang, Chew Lim Tan, and Haizhou Li. 2009. K-best combination of syntactic parsers. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1552–1560. Association for Computational Linguistics, Singapore. URL http://www.aclweb.org/anthology/D/D09/D09-1161. 97

Ruiqiang Zhang, Genichiro Kikui, and Eiichiro Sumita. 2006. Subword-based tagging by conditional random fields for Chinese word segmentation. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 193–196. Association for Computational Linguistics, New York City, USA. URL http://www.aclweb.org/anthology/N/N06/N06-2049. 29, 36

Yue Zhang and Stephen Clark. 2007. Chinese segmentation with a word-based perceptron algorithm. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 840–847. Association for Computational Linguistics, Prague, Czech Republic. URL http://www.aclweb.org/anthology/P07-1106. 15, 18, 22, 29, 33

Yue Zhang and Stephen Clark. 2008a. Joint word segmentation and POS tagging using a single perceptron. In *Proceedings of ACL-08: HLT*, pages 888–896. Association for Computational Linguistics, Columbus, Ohio. URL http://www.aclweb.org/anthology/P/P08/P08-1101. 32, 33

Yue Zhang and Stephen Clark. 2008b. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 562–571. Association for Computational Linguistics, Honolulu, Hawaii. URL http://www.aclweb.org/anthology/D08-1059. 78, 82, 84, 142, 152

Yue Zhang and Stephen Clark. 2009. Transition-based parsing of the Chinese treebank using a global discriminative model. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, pages 162–171. Association for Computational Linguistics, Paris, France. URL `http://www.aclweb.org/anthology/W09-3825`. 81, 84

Yue Zhang and Stephen Clark. 2010. A fast decoder for joint word segmentation and POS-tagging using a single discriminative model. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 843–852. Association for Computational Linguistics, Cambridge, MA. URL `http://www.aclweb.org/anthology/D10-1082`. 32, 41, 45

Qiang Zhou. 2004. Annotation scheme for Chinese treebank (in Chinese). *Journal of Chinese Information Processing*, 18(4):1–8. 104

Tao Zhuang and Chengqing Zong. 2010. A minimum error weighting combination strategy for Chinese semantic role labeling. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1362–1370. Coling 2010 Organizing Committee, Beijing, China. URL `http://www.aclweb.org/anthology/C10-1153`. iv, vi, 136, 147, 148, 157, 160