Exploiting Vocabulary, Morphological, and Subtree Knowledge to Improve Chinese Syntactic Analysis

Kyoto University
Graduate School of Informatics
Department of Intelligence Science and Technology

Mo Shen

Doctoral Dissertation
2016

Exploiting Vocabulary, Morphological, and Subtree Knowledge to Improve Chinese Syntactic Analysis

Mo Shen

Supervisor: Daisuke Kawahara

Abstract

Syntactic analysis in the Chinese language with supervised or semi-supervised machine learning systems, including word segmentation, part-of-speech (POS) tagging, and constituency and dependency parsing, has been actively studied for around two decades since the publish of the first version of Penn Chinese Treebank (CTB). However, evaluations in major natural language processing (NLP) tasks suggest that the performance of state-of-the-art systems for Chinese is constantly lower than that for European languages. Compared to most European languages, one of the biggest disadvantages in Chinese is that, word segmentation must be performed before any further syntactic analysis can be applied. Due to the ambiguous nature of the definition of "word" as well as the lack of morphological inflection in Chinese language which we will describe in more details in later chapters, the task of word segmentation is more challenging than other Asian languages, such as Japanese; the lack of morphological inflection also makes it more difficult for a POS tagging or dependency parsing system to predict the correct tags or dependencies. Moreover, though most languages are evolving more rapidly in this internet era, Chinese language, due to its unique word formation ability which coins a large amount of new compound words that is impossible to keep track of, suffers more from the OOV problem in syntactic analysis. To address these difficulties, we investigated the strength and weakness of previous

studies and proposed new methods that improve the state-of-the-art systems, which include: a complete set of annotation guidelines for Chinese word segmentation, a tagset for part-of-speech tagging and a label set for dependency labelling to overcome the inconsistency and data sparsity problems in existing treebanks; an algorithm that extracts substrings as reliable word boundary indicators which significantly enhance the accuracy of word segmenters; the first tagset designed for the task of character-level POS tagging, based on which we manually annotate the entire CTB5; a method that performs character-level POS tagging jointly with word segmentation and word-level POS tagging; and a feature set that makes fully use of dependency grammar, can capture global information with less restriction in the structure and the size of the sub-tree context, and can be encoded efficiently.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Syntactic analysis in the Chinese language with supervised or semi-supervised machine learning systems, including word segmentation, part-of-speech (POS) tagging, and constituency and dependency parsing, has been actively studied for around two decades since the publish of the first version of Penn Chinese Treebank (CTB). However, evaluations in major natural language processing (NLP) tasks suggest that the performance of state-of-the-art systems for Chinese is constantly lower than that for European languages such as English, German, Spanish and French. We consider that the following difficulties are the major reasons which caused such a performance gap:

- Chinese sentences are written without explicit word boundaries, which leads to ambiguities.
- Chinese words are non-inflectional, which makes it hard for a parser to predict the POS tags and dependency relations in a sentence.
- Words in Chinese keep evolving: new words (out-of-vocabulary (OOV) words) are coined in a rapid pace with the boom of the internet.

To illustrate these difficulties in Chinese syntactic analysis, let's consider the following example: "中国教育界面临改革" (China's educational field is facing a reform). One of the linguistically correct segmentation of this sentence is "中国/China 教育界/educational field 面临/face 改革/reform". For a word segmenter based on a lexicon, this is not necessarily a trivial case, since "国教/national religion" and "界面/interface" are common words that are likely to exist in the lexicon. On the other

hand, “教育界/educational field”, besides being overlapping with these two words which increases the ambiguity of the segmentation, is also a compound word which consists of a stem morpheme “教育/educational” and a suffix morpheme “界/field”; compounds with the same suffix morpheme “界/field” are abundant in Chinese, and by attaching such a suffix morpheme to either an existing word, or a newly formed word (which is usually itself a compound, e.g. “物联网/ internet of things”, or an abbreviation, e.g. “电竞/e-sports”), new words can be easily formed (e.g. “物联网界/industry of internet of things”, “电竞界/industry of e-sports”). Compounds with such a structure is one of the major sources of new words that are being coined in Chinese, which can be considered as a unique word formation ability of this language, compared to European languages. These facts all suggest that it is impossible to collect all words in a lexicon, and even for word with a simple structure such as “教育界/educational field”, it is still possible for a segmenter to make mistakes in segmentation. On top of that, words in the Chinese language do not have any morphological inflections. In this example, “改革/reform” should be interpreted as a noun; however, if we change the sentence into “中国教育界即将改革” (China’s educational field will be reformed soon), it should instead be interpreted as a verb, without any change in its word form.

Compared to most European languages, one of the biggest disadvantages in Chinese is that, word segmentation must be performed before any further syntactic analysis can be applied. Due to the ambiguous nature of the definition of “word” as well as the lack of morphological inflection in the Chinese language which we will describe in more details in later chapters, the task of word segmentation is more challenging than other Asian languages, such as Japanese; the lack of morphological inflection also makes it more difficult for a POS tagging or dependency parsing system to predict the correct tags or dependencies. Moreover, though most languages are evolving more rapidly in this internet era, the Chinese language, due to its unique word formation ability which coins a large amount of new compound words

that are impossible to keep track of, suffers more from the OOV problem in syntactic analysis. It can be observed that other languages with all these three difficulties, such as Thai, often also have lower performance in the syntactic analysis. Consequently, it requires the development of both systems and resources to be specifically based on the characteristics of the Chinese language, in order to overcome these difficulties.

In dependency parsing, a parser relies on word forms and part-of-speech tags as features to make parsing decisions; with the word forms being sparse in the existing corpora and part-of-speech tags being ambiguous for Chinese words due to the difficulties we introduced above, it is also easy for a dependency parser to make incorrect decisions, and subtree-based techniques which do not solely rely on the accuracies of word segmentation and/or part-of-speech tagging should be investigated.

In the following sections, we will first describe the challenges of each major tasks in Chinese syntactic analysis and briefly introduce the existing methods and studies. We will then give an overview of our approach to tackling these challenges.

1.1 Chinese Word Segmentation and Treebank Annotation

We first raise the following two questions:

1. Is the available treebank resources such as Penn Chinese Treebank (CTB) providing consistently annotated data?
2. Is it sufficient to use *only* the annotated data in the existing treebanks to train an accurate Chinese word segmenter which can be used in different, unconventional scenarios, such as processing web documents (forum articles, reviews, blogs, tweets, etc.) or queries?

The answer to both questions is no. The definition of "word" is an open issue in Chinese linguistics. In previous studies of Chinese corpus annotation (Duan et al., 2003; Huang et al., 1997; Xia, 2000), the judgement of word-hood of a meaningful string is based on the analysis of morphology: a morpheme in Chinese is defined as the smallest combination of meaning and phonetic sound in the Chinese language. An issue with word definition using morpheme classification is that it potentially undermines the consistency of the representation of words. For example, "论" (theory) is a bound morpheme, therefore the string "进化论" (theory of evolution) is treated as a word; on the other hand the string "进化 | 理论" (theory of evolution) are treated as two words, despite the fact that the two strings have the same meaning and structure. In another example, "者" (person) is considered as a bound morpheme, and thus "反对自由贸易者" (people who are against free trade) is treated as one word, while the string without the bound morpheme, i.e. "反对 | 自由 | 贸易" (be against free trade), can only be treated as a phrase of three words.

The morphology-based word definition can also make the data sparsity problem worse in corpus annotation. As an evidence, in the Penn Chinese Treebank 5.0 (CTB5) which is an annotated corpus widely used to train Chinese morphological analysis systems, we found that one of the major sources of the OOV words is the compounds that end with a monosyllabic bound morpheme. For example, compounds "利用率" (utility rate) and "次品率" (rate of defective product) end with the bound morpheme "率" (rate); "完成度" (degree of completion) and "活跃度" (degree of activity) end with the bound morpheme "度" (degree); "持续性" (sustainability) and "挥发性" (property of volatile) end with the bound morpheme "性" (property). While these compounds are sparse in the corpus, the morphemes which they consist of can be frequently observed; this means that these OOV words can be observed and learnt by a word segmenter if we split the morphemes as individual words in the annotation.

Given a consistently annotated corpus, we can train a Chinese word segmenter with reasonable accuracy on in-domain data. However, such a segmenter would still produce low-accuracy results on out-of-domain data, such as web documents and queries. Previous studies have shown that with a comprehensive lexicon, even a simple maximum matching segmentation algorithm can yield an F-score as high as 0.99 (Sproat and Emerson 2003). This suggests that the lack of knowledge of vocabulary presents the biggest challenge in Chinese word segmentation. It is impossible, however, to collect a complete list of Chinese words. The Chinese language is continually and rapidly evolving, particularly with the rapid growth of the internet. Therefore, it is necessary to develop techniques that automatically generate vocabulary lists from large-scale web texts instead of relying on supervised CWS systems.

Recent research has attempted to exploit characteristics of frequent substrings in unlabeled data. Statistical criteria that measure the likelihood of a substring being a word have been proposed in previous studies of unsupervised segmentation, such as "accessor variety" (Feng et al. 2004), a criterion measuring the likelihood of a substring being a word by counting distinct surrounding characters. In (Jin and Tanaka-Ishii 2006) the researchers proposed "branching entropy", a similar criterion based on the assumption that the uncertainty of surrounding characters of a substring peaks at the word boundaries. In (Ye et al. 2013), the likelihood of a character n-gram being a meaningful Chinese word is measured by "overlap variety", a criterion which considers the goodness of the candidate together with the goodness of the strings overlapping the candidate. The authors have also adopted a word refinement module to filter out known words during the post-processing. The authors of (Zhao and Kit 2007) incorporated accessor variety and another type of criteria, called "co-occurrence sub-sequence", with a supervised segmentation system and conducted comprehensive experiments to investigate their impacts. There are several restrictions in using the co-occurrence sub-sequence criteria: it requires post-processing to remove overlapping instances; sub-sequences are retrievable only from different sentences; and the retrieval is performed only on training and testing data.

In (Sun and Xu 2011), the authors proposed a semi-supervised segmentation system enhanced with multiple statistical criteria; large-scale unlabeled data were used in their experiments. Jiang et al. (2010) used term contributed boundary information extracted from unlabeled text as features in semi-supervised learning to improve character-based Chinese word segmentation.

1.2 Part-of-Speech Tagging

In recent years, the focus of research on Chinese word segmentation, part-of-speech (POS) tagging and parsing has been shifting from words toward characters. Character-based methods have shown superior performance in these tasks compared to traditional word-based methods (Ng and Low, 2004; Nakagawa, 2004; Zhao et al., 2006; Kruengkrai et al., 2009; Xue, 2003; Sun, 2010). Studies investigating the morphology-level and character-level internal structures of words, which treat character as the true atom of morphological and syntactic processing, have demonstrated encouraging results (Li, 2011; Li and Zhou, 2012; Zhang et al., 2013). This line of research has provided great insight in revealing the roles of characters in word formation and syntax of the Chinese language.

However, existing methods have not yet fully utilized the potentials of Chinese characters. While Li (2011) pointed out that some characters can productively form new words by attaching to existing words, these characters consist only a portion of all Chinese characters and appear in 35% of the words in Chinese Treebank 5.0 (CTB5) (Xue et al., 2005). Zhang (2013) took one step further by investigating the character-level structures of words; however, the machine learning method of inferring these internal structures relies on the character forms, which still suffers from data sparseness.

In our view, since each Chinese character is in fact created as a word in origin with complete and independent meaning, it should be treated as the actual minimal

morphological unit in the Chinese language, and therefore should carry specific part-of-speech. For example, the character “打” (beat) is a verb and the character “破” (broken) is an adjective. A word on the other hand, is either single-character, or a compound formed by single-character words. For example, the verb “打破” (break) can be seen as a compound formed by the two single-character words with the construction “verb + adjective”.

1.3 Dependency Parsing

In dependency parsing, graph-based models (McDonald et al., 2005; McDonald and Pereira, 2006) are prevalent for their state-of-the-art accuracy and efficiency, which are gained from their ability to combine the exact inference and discriminative learning methods. The ability to perform the exact inference lies on the technique called factorization which breaks down a parse tree into smaller parts to perform an efficient dynamic programming search. This treatment, however, restricts the representation of features to a local context which can be word-pairs or adjacent edges. Such restriction prohibits the model from exploring a large and complex structure for linguistic evidence, which is the major drawback compared with other dependency parsing framework like transition-based models (Yamada and Matsumoto, 2003; Nivre et al., 2004; Attardi, 2006).

Attempts have been made in developing more complex factorization techniques and the corresponding decoding methods. Higher-order models that use grand-child, grand-sibling or tri-sibling factorization were proposed in (Koo and Collins, 2010) to explore more expressive features and have proven significant improvement on parsing accuracy. However, to utilize the power of higher-order models also suffers from expensive training cost or requires aggressive pruning in the pre-processing.

Another line of research that explores complex feature representations is parse re-ranking. In a general framework, a K-best list of parse tree candidates is first

produced from the base parser; a re-ranker is then applied to pick up the best parse for the input sentence. For constituent parsing, successful results have been reported in (Collins, 2000; Charniak and Johnson, 2005; Huang, 2008). For dependency parsing, the efficient algorithms to produce K-best list for graph-based parsers have been proposed in (Huang and Chiang, 2005) for projective parsing and in (Hall, 2007) for non-projective parsing; improvements on dependency accuracy have been achieved in (Hall, 2007; Hayashi et al., 2011). However, the feature sets in these studies explored a relatively small context, either by emulating the feature set in the constituent parse re-ranking, or by factorizing the search space. A desirable approach for the K-best list re-ranking is to encode features on sub-trees extracted from the candidate parse with arbitrary orders and structures, as long as the extraction process is tractable. It is an open question how to design this sub-tree extraction process that is able to select a set of sub-trees which provide reliable and concrete linguistic evidence. Another related challenge is to design a proper back-off strategy for any structures extracted, since large sub-tree instances are very sparse in all training data.

1.4 Overview of Our Approach

To address the issues observed in various tasks in Chinese syntactic analysis we introduced above, we investigated the strength and weakness of previous studies and proposed new methods that improve the state-of-the-art systems. An overview of our approach is illustrated in Figure 1.

Our contributions are as follows.

First, we proposed a complete set of annotation guidelines for Chinese word segmentation, a tagset for part-of-speech tagging and a label set for dependency labeling to overcome the inconsistency and data sparsity problems in existing treebanks.

Second, we proposed an algorithm that extracts substrings as reliable word boundary indicators which significantly enhance the accuracy of word segmenters. The algorithm takes linear time in the average case, which is a big advantage in processing large-scale data.

Third, we investigated the usefulness of character-level POS in the task of Chinese part-of-speech tagging. We proposed the first tagset designed for the task of character-level POS tagging, based on which we manually annotate the entire CTB5. We further proposed a method that performs character-level POS tagging jointly with word segmentation and word-level POS tagging.

Finally, we explored a feature set that makes fully use of dependency grammar, can capture global information with less restriction in the structure and the size of the sub-tree context, and can be encoded efficiently. It exhaustively explores a candidate parse tree for features from the most simple to the most expressive while it maintains the efficiency in terms of training and parsing time.

In the remainder of this thesis, we arrange the contents as follow: in Chapter 2 we present a new annotation approach to Chinese word segmentation, part-of-speech (POS) tagging and dependency labeling that aims to overcome the two major issues in traditional morphology-based annotation: inconsistency and data sparsity. In Chapter 3 we describe a simple yet effective approach for extracting a specific type of frequent substrings, called maximized substrings, which provide good estimations of unknown word boundaries; in the task of Chinese word segmentation, we incorporate extracted substrings in a semi-supervised system to improve the segmentation accuracy. In Chapter 4 we investigate the usefulness of character-level part-of-speech in the task of Chinese morphological analysis; we further propose the first tagset designed for the task of character-level POS tagging and a method that performs character-level POS tagging jointly with word segmentation and word-level POS tagging. In Chapter 5 we present a new reranking method for dependency parsing that can utilize complex subtree representations by applying efficient subtree selection methods. We conclude our work in Chapter 6.

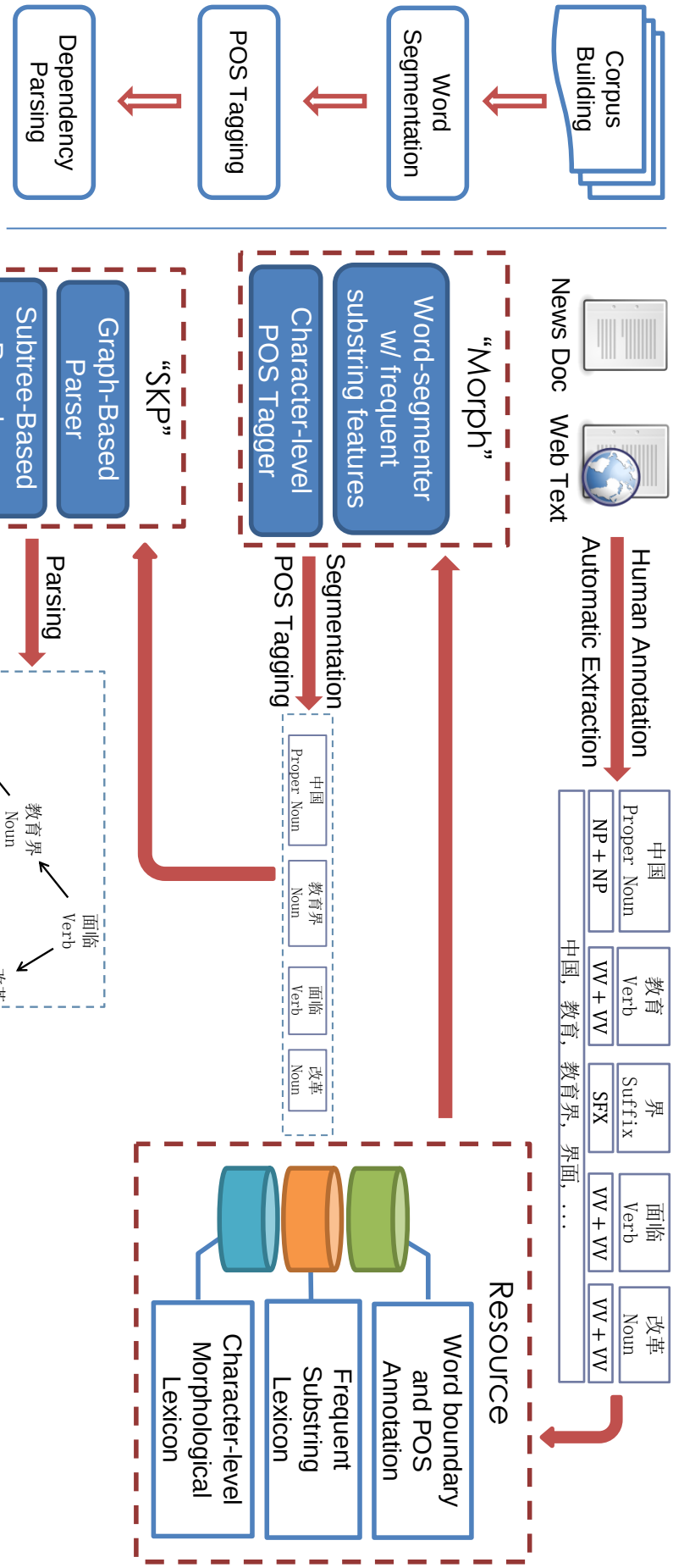


Figure 1. Overview of our approach.

Chapter 2

Consistent Chinese Word Segmentation, Part-of-Speech Tagging and Dependency Labeling Annotation

The definition of "word" is an open issue in Chinese linguistics. In previous studies of Chinese corpus annotation (Duan et al., 2003; Huang et al., 1997; Xia, 2000), the judgement of word-hood of a meaningful string is based on the analysis of morphology: a morpheme in Chinese is defined as the smallest combination of meaning and phonetic sound in the Chinese language, which can be classified into two major types:

1). **free morphemes**, which can either be words by themselves or form words with other morphemes; and

2). **bound morphemes**, which can only form words by attaching to other morphemes.

An issue with word definition using morpheme classification is that it potentially undermines the consistency of the representation of words. For example, "论" (theory) is a bound morpheme, therefore the string "进化论" (theory of evolution) is treated as a word; on the other hand the string "进化 | 理论" (theory of evolution) are treated as two words, despite the fact that the two strings have the same meaning and structure. In another example, "者" (person) is considered as a bound morpheme, therefore "反对自由贸易者" (people who are against free trade) is treated as one word, while the string without the bound morpheme, i.e. "反对 | 自由 | 贸易" (be against free trade), can only be treated as a phrase of three words.

The morphology-based word definition can also make the data sparsity problem worse in corpus annotation. As an evidence, in the Penn Chinese Treebank 5.0 (CTB5) which is an annotated corpus widely used to train Chinese morphological analysis systems, we found that one of the major sources of the out-of-vocabulary (OOV) words is the compounds that end with a monosyllabic bound morpheme. For example, compounds 利用率 (utility rate) and 次品率 (rate of defective product) end with the bound morpheme 率 (rate); 完成度 (degree of completion) and 活跃度 (degree of activity) end with the bound morpheme 度 (degree); 持续性 (sustainability) and 挥发性 (property of volatile) end with the bound morpheme 性 (property). While these compounds are sparse in the corpus, the morphemes which they consist of can be frequently observed; this means these OOV words can be observed and learnt by a word segmenter if we split the morphemes as individual words in the annotation.

In this chapter, we propose a complete set of annotation guidelines for Chinese word segmentation that overcomes the two issues: inconsistency and data sparsity, which are found in the traditional morphology-based annotation approach. We further propose a tagset for part-of-speech tagging and a label set for dependency labelling, which are consistent with our word segmentation strategy and capture more Chinese-specific syntactic structures. We re-annotate the entire CTB5 using this approach, and through word segmentation, POS tagging and machine translation experiments we demonstrate the advantages of our annotation approach compared to the original approach adopted in CTB5.

In the following sections of this chapter we will first describe our proposed annotation approach to word segmentation, then present a POS tagset and a new dependency label set which is consistent with our word segmentation strategy, and finally demonstrate the effectiveness of our approach compared to the original CTB5 through experiments.

2.1 Word Segmentation Annotation

In this section we present a complete set of annotation guidelines for Chinese word segmentation. The objective of the following guidelines is to determine the word-hood of any meaningful strings observable in raw Chinese texts. The definition of words is an unresolved issue in Chinese linguistics. In previous studies, the word-hood of a string is usually determined by applying a set of tests, many of which are subjective/statistical and therefore imprecise, such as "Does the string of characters frequently co-occur". In some cases, language resources are required in order to determine the word-hood of a string. For example, for the string "大提琴手", a list of bound morpheme is applied to determine whether the character "手" is part of the word or not. In this guideline, to guarantee the consistency of the annotated corpus, we do not apply any subjective or statistical test (or at least to do it as less as possible); to meet the needs of some practical NLP tasks and to ensure the flexibility of the annotated corpus, we do not use any existing language resource as part of the segmentation standard[1].

We categorize the words in CTB5 into three categories: common words, names, and idioms. For names and idioms, we keep them as individual words since their word boundaries are relatively easy to recognize and the consistency in manual annotation can be achieved with less efforts. We will describe the specific rules for segmenting a word in any of these categories in the following subsections.

The key in our method to define the boundaries of common words is the character-level POS pattern. Character-level POS has been introduced in previous studies (Zhang et al., 2013; Shen et al., 2014) which captures the grammatical roles of Chinese characters inside words; we further develop this idea and use it as a criterion in word definition.

[1] In most cases, in order to adapt to the style of other segmentation standards, we can recover or adjust the word boundaries of the annotated corpus in the post-processing with the aid of proper language resources.

POS Pattern	Example
pronoun + noun	我校 (this university) 他处 (other agencies)
locative + noun	前院 (front yard) 后门 (back door)
locative + verb	前述 (described above) 后略 (and so on)
noun + locative	室内 (indoor) 赛前 (pre-match)
pronoun + locative	此外 (besides) 其中 (among which)
adverb + verb	胡说 (nonsense) 猝死 (sudden death)
noun + noun	山地 (hills) 厂房 (factory plant)
noun + measure	车辆 (vehicles) 船只 (ships)
adjective + noun	佳酿 (wines) 蓝筹 (blue chips)
adjective + measure	高层 (high level) 大幅 (greatly)
verb + verb	抽取 (extract) 打扫 (clean)
verb + particle	变成 (turn out) 写完 (finish writing)
verb + adjective	打碎 (break) 看清 (see clearly)
verb + locative	如下 (as follows) 综上 (accordingly)

verb + noun	辞职 (resign) 流血 (bleed)
adjective + adjective	美好 (wonderful) 优雅 (elegant)
adverb + adjective	极难 (extremely hard) 最新 (latest)
determiner + noun	该市 (the city) 各界 (all walks of life)
determiner + temporal	翌日 (the next day) 当天 (the same day)

Table 1. Disyllabic character-level POS patterns.

We treat a meaningful disyllabic strings as a word if it falls into one of the character-level POS patterns listed in Table 1. The reason we focus on disyllabic patterns instead of other polysyllabic ones is that, based on our observation, meaningful strings with 3 or more syllables (other than names and idioms) are always compounds in Chinese, and therefore can be segmented into a sequence of monosyllabic and disyllabic tokens based on their internal structures. On the other hand, the internal structure in a disyllabic token, though still exists, is more implicit and harder to describe with syntactical relations; we believe that it would increase the difficulties for subsequent tasks, such as dependency parsing, if we further segment these disyllabic strings. Following this strategy, a polysyllabic word can be then segmented based on its structure. This is illustrated with examples in Table 2.

In the following subsections we will first describe the basic method of our segmentation strategy; we will then describe the segmentation rules in details; finally we will conclude this section with a collection of ambiguous example and their corresponding treatments according to our guidelines.

2.1.1 Syllable Count

Syllable count is one of our segmentation criteria. Normally, a syllable in Chinese corresponds to a character, with certain exceptions such as:

- multi-character Syllable: 花兒(flower) is monosyllabic in 一|朵|花兒 (a flower);
- multisyllabic Character: Some Chinese characters are multisyllabic, which are usually used to express units in the metric system. E.g. "**兙**" (qian1ke4), "**糎**" (li2mi3), "**甽**" (qian1wai3), "**圖**" (tu2shu1guan3).

2.1.2 Substitution Test

Under the same context and given the rules of substitution, if a string can be transformed while its meaning is still comprehensible, we call that the substitution is valid.

2.1.3 Sub-token Level POS Pattern

Segmentation of a meaningful string based on sub-token level POS patterns takes the following steps. Note that the POS tagging of the resulted tokens after these steps is an independent subsequent process.

(1) Disambiguate the meaning of the string based on context if necessary.

E.g. "四面" (four faces) in "四面環山" and "四面" (four) in "四面屏風"

(2) Light parsing should be done in order to limit the span. These grammatical relations should be taken into considerations.

- Predication 主 | 謂
- Complementation 動 | 賓/補
- Adjunction/Modification 狀/定 | 中
- Apposition 同位
- Coordination 並列

E.g. 胡說什麼 -> Complementation 胡說|什麼 (c.f. *Complementation 說|什麼 -> *Adjunction 胡|說什麼)

NB: as the string (胡+說) is "frozen", the adjunction that is at play is only limited to the sub-tokens.

(3) Determine the internal structure and sub-token level POS pattern of the string. The context used to determine the POS of each sub-token is limited to the span of the string itself.

E.g. To determine the sub-token level POS pattern of the string "依此類推", an example of the analysis would be:

- determine the surface pattern as "依此/AD+类推/VV";
- limit the context to "依此" and determine the pattern as "依/VV+此/PRD";
- limit the context to "類推" and determine the pattern as "類/AD+推/VV".

(4) Determine the internal word boundaries according to the rule for a specific POS pattern.

2.1.4 Principle of Semantic Compositionality

If the meaning of the string cannot be derived from the combination of the meanings of its sub-components, the string should be considered as one token. This criterion will only be used as a fallback if no other rule is applicable.

2.1.5 Specific Rules for Proper Nouns

Person's Names

- Segment into first name, last name, and title (if any).
    - 張 | 朝陽
    - 歐陽 | 海
    - 小澤 | 徵爾
    - 小 | 約翰 | 施特勞斯
    - 約翰 | 保羅 | 二世
- Note: if the name contains punctuations, segment them as well.
    - 卡爾 | · | 馬克思

Country Names

- Segment the proper noun part from the remainder.
    - 大不列顛 | 聯合 | 王國

Names of Ethnic Groups, Locations, Buildings or Natural Sites

For the categories listed below: keep transliterated names as single tokens. If the name is not transliterated, keep disyllabic names as single tokens; otherwise segment according to its internal structure.

- Name of ethnic groups
    - 維吾爾 | 族

- Name of locations
    - 北京 | 市
    - 浙江 | 省
    - 貝克 | 街
    - 下海 | 廟
    - 提籃 | 橋
    - 石家 | 莊
    - 荔枝 | 角
    - 大浪 | 凹
    - 跑馬 | 地
    - 白 | 沙咀
    - 添馬 | 艦
    - 赤洲 | 口
    - 九龍 | 半島
    - 華北 | 平原
    - 皇后 | 大道
    - 黃泥 | 涌 | 峽
    - 麒麟 | 山 | 坳
    - 北卡羅來納 | 州
- Name of buildings or natural sites
    - 黃鶴 | 樓
    - 洞庭 | 湖
    - 雅魯藏布 | 江

Names of Brands and Types

- Segment the proper noun part from the rest; segment any expression in foreign languages (including alphanumeric strings, dashes, slashes, hyphens, etc.) from Chinese expressions; for Chinese expressions, segment any numerals from the rest.
    - 松下 | 牌
    - CSS6 | 型
    - 天河 | 二 | 號

Name of Newspapers, Newswires, Books, TV shows, Movies

- First segment the proper noun part from the remainder; then segment the remainder if it has 3 syllables or more.

- 明報
- 大公 | 報
- 法新 | 社
- 新民 | 晚報
- 新華 | 通訊 | 社

Names and Units of Currency

- Segment the unit as a single token; segment the proper noun part (if any) as a single token; segment the remainder if it has 3 syllables or more.
  - 美 | 元
  - 日 | 幣
  - 新 | 台 | 幣
  - 人民 | 幣
  - 新 | 土耳其 | 里拉
  - 印度尼西亞 | 盧比

Abbreviations of Proper Nouns

- Treat one abbreviation as one token.
  - 中 | 美
  - 港 | 澳 | 台
- Unless it is a part of an acronym.
  - 美聯儲

2.1.6 Specific Rules for Common Words

Names of Relatives

- Do not segment.
  - 表哥
  - 二姨妈
  - 大姑父

JJ+NN

- Segment the string if one of the following conditions is met:
  - the string is disyllabic and the following substitution is valid:

        - rule: JJ+NN -> 很+JJ+的+NN
        - E.g. 厚|書 強|隊
    - The string is multisyllabic (3 syllables or more).
        - E.g. 新|版本 副|主席 共同|利益
- Otherwise, treat the string as one token.
    - E.g. 女人 鮮花 紅茶 好評 大洲 大海

Date and Time

- If the string contains numbers, segment the number part from the rest.
    - 九八|年
    - 5|點鐘
    - 三|點|十|分
    - 90|年代
    - 21|世紀
- If the string describes a period of time, segment the string into the number part, the measure word (if exists), and the period.
    - 3|年
    - 5|小時
    - 3|個|月

Determiner (DT)

- Treat “這”, “那”, “哪”, “這些”, “那些”, “哪些” as single tokens.
- Treat disyllabic DT+X as a single token, if X is either NN or NNB.
    - 本校
    - 當地
    - 眾人
    - 該市
    - 各界
    - 翌日
- Segment multisyllabic (3 syllables or more) DT+X strings.
    - 該|地區

Cardinal Number (CD)

- Treat as a single token.
    - Note: for the following intermediate strings in a cardinal number expression, treat them as part of the word:

        - prefix: 數, 好, 幾;
        - intermediate: 點, 分之;
        - suffix: 餘, 幾, 多, 來.
    - E.g. 三點一四, 二分之一, 三千餘, 好幾, 二十幾, 50 多, 七百來

Ordinal Number (OD)

- Combine the prefix "第" with the cardinal number.
    - 第四十五
    - 第 3
- Treat disyllabic OD+X as a single token, if X is either NN or NNB.
    - 首屆
    - 首位
    - 次席
- Otherwise treat OD as single tokens.

Measure Word (NNB)

- Separate it from cardinal numbers (CD).
    - 三|個
    - 兩|輛

NN+VV

- If string is disyllabic:
    - segment if it forms a subject-predicate relation;
        - 頭|暈
        - 花|開
    - otherwise treat as a single token.
        - 宇航
        - 草食
- If string is multisyllabic (3 syllables or more), segment the string.
        - 腿|抽筋
- Note: NR+VV is always separated.
    - 美|制

VV+NN/NNB

- Do not segment if the string is disyllabic; segment the string if it is multisyllabic (3 syllables or more).

    - 辭職
    - 流血
    - 在世
    - 到家
    - 同意
    - 打|網球
    - 手擀|面
- Note 1: if the string is VV+NR, segment the string.
    - 訪|華
    - 旅|英
- Note 2: in a VV+Coverb+NN construction, NN is always separated.
    - 回到|家

Preposition (IN)

- Treat IN as single tokens.
    - 為|何 (為=wei4)
    - 因|故
    - 始|於
    - 走|向

VV+Complement

- Do not segment if the string is disyllabic; segment the string if it is multisyllabic (3 syllables or more).
    - 打碎
    - 吃掉
    - 洗|乾淨
- Note: if the string contains "得、不、起來", see the next subsection.

VV+"得/不"+Complement

- Segment the string.
    - 寫|不|完
    - 做|得|不好
    - 高興|不|起來

A+Not+A

- If and only if A is monosyllabic, treat the string as one token.
    - 看不看
    - 好不好
    - 是否是
    - 有沒有
    - 好看|不|好看
- Note: a variation of this is A+Not+AB where A is monosyllabic, e.g. 好不好看. Segment the string in this case.
    - 好|不|好看
    - 寫|沒|寫完

VV+Coverb

- Treat disyllabic "VV+Coverb" as single tokens.
    - 做到
    - 看似
    - 變成
- Exception: segment "VV+在".
    - 放|在
    - 坐|在
- For multisyllabic strings (3 syllables or more): segment the string.
    - 認識|到
    - 翻譯|成

MSP: 所, 以, 來, 而

- Treat as a single token.

AS: 了, 著, 過, 的

- Treat as a single token.
- Exception: 本著, 為著, 藉著, 憑著, 當著, 循著, 朝著, 沿著, 趁著, 就著, 為了, 經過, 繞過, 超過

SP: 嗎，啦，啊，吧，呢，…

- Treat as a single token.

“些”

- Treat a disyllabic string of “X+些” structure that is one of or similar to the following examples as one token:
    - 一些，有些，好些，這些，哪些
- Otherwise treat “些” as one token.

BA: 把

- Treat as a single token

BEI: 被

- Treat as a single token.

PUNCT

- Treat as a single token.

FW

- Treat as a single token.
    - Exception: do not segment the string if it is a combination of foreign characters and transliterations. E.g. 卡拉 OK

Idioms[2]

- Treat as a single token.

Abbreviations

- Tokenize abbreviation of proper nouns with the following steps:
    - tokenize the string of its full name;
    - compare the short form with its full name. Abbreviated words under the same constituent can form one token. Otherwise, treat as separate token(s).
        - 發展|改革|委(full name: 中華|人民|共和|國|國家|發展|和|改革

[2] Whether a string is an idiom or not can be subjective. With no other solutions available, we will use external resources (e.g. an idiom dictionary) to help making decisions. Another note is that "Playing with word"/mis-spelling of idioms are still treated as idioms, as long as the origin can be found.

|委員|會)
    - 文史|辦(full name: 全國|政協|文史|和|學習|委員|會|辦公|室)
    - 外經貿委(full name: 對外|經濟|貿易|委員|會)
    - 職工|盟 (full name: 香港|職工|會|聯盟)
    - 駐|京|辦 (full name: 駐|北京|辦事|處)
  - For the case that the short form is reordered, treat the string as single token
    - 七中全會(full name: 中央|委員|會|第七|次|全體|會議)
- Note 1: do not extend this rule to common nouns.
  - 蛋牛|治 (meaning: 雞蛋牛肉三文治)
    - |90|後| (meaning: 90 年以後出生的人)
    - 低保|戶(meaning: 低於市低保標準的居民)
- Note 2: when the strings contains a proper noun part: this rule applies if and only if the entire string is also a proper noun.
    - 港姐 (“香港小姐競選”)
    - 港|姐 (“香港小姐”)
    - 美聯儲 (“美國聯邦儲備”)

Monosyllabic VC: 是, 為, 即, 非

- Treat monosyllabic VC as a single token[3].
  - Exception 1: 是否, 不 VC, 是不是, 是否是
  - Exception 2: disyllabic X+VC, where X is CC, CS, or conjunctive adverb.
    - E.g. 還是, 就是, 但是, 凡是, 或是, 也是, 亦即, 且為, …
  - Exception 3: X+VC is frozen or VC does not function as VC in the context.
    - E.g. 正是, 真是, 亟為, 極為, 當即, 甫即, 隨即, 無非, 莫非, ...

Interjections: 啊，哈，哦，哎呀，天啊，嗚呼，…

- Treat as a single token.

Onomatopoeia: 嘩啦，轟隆隆，…

[3] Note that these characters can also appear in other categories, in which case, please refer to the corresponding sections and make segmentation decisions accordingly: As NN: 是非; As DT: 是日; in idiom: 莫衷一是 惟利是圖 各行其是.

- Treat as a single token.

Coordinating Conjunction/Subordinating Conjunction/Conjunctive Adverbs

- Treat as a single token[4].

Duplications

- Treat as a single token. E.g.
  - AA: 看看 試試 藍藍 好好 僅僅 常常
  - AAB: 濛濛亮 分分鐘
  - ABB: 綠油油 亮晶晶
  - ABAB: 打扮打扮 雪白雪白
  - AABB: 來來往往 幹乾淨淨

Antonymous Symmetrical Structures

- Treat as a single token. E.g.
  - LC+LC[5]: 上下，前後，裡外
  - VV+VV: 開闔，張弛，伸縮
  - NN+NN: 天地, 是非
  - JJ+JJ: 好歹

Telescopic Compounds

- If the string contains no punctuation, treat it as one token.
  - E.g. 父母親 青少年 國內外 大中小學 三四十
- Otherwise, segment the string.
  - E.g. 大，中，小學: 大|，|中|，|小學
  - E.g. 一、兩萬元: 一|、|兩萬|元

Disyllabic Strings[6]

---

[4] This means if a conjunction/conjunctive adverb precedes a verb, they should be separated even if the string is disyllabic, e.g. 而|來, 就|會, 並|給, 已|付, 且|行, 尚|存, 僅|剩. The same goes for MSP (所, 以, 來, 而).

[5] LC represents the collection of localizers: "上, 下, 左, 右, 前, 後, 內, 裡, 外, 東, 南, 西, 北, etc." Although the tag LC is obsolete in the tagging guidelines, we will still use it for segmentation analysis.

- For a disyllabic string which has a pattern in the list below, treat it as a token unless otherwise specified[7]:
    - PRON+NN(代+名): 我校 我院 他處
    - LC+NN(處+名): 後門 前院 左肩
    - LC+NNB(處+量): 上個 下週 前日
    - LC+VV(處+動): 前述 下稱 後略
    - NN+LC(名+處): 室內 賽前 月下
    - PRON+LC(代+處): 此前 此外 其中
    - AD+VV(副+動)[8]: 胡說 敬獻 猝死
    - AD+NNB(副+量): 再度 再次 連年
    - Not+JJ(否定+形): 不好 不妙 不悅
    - Not+VV(否定+動)[9]: 不解 不及 不屑
    - Not+MD(否定+情態): 不願 未肯 沒能
    - NN+NN(名+名): 山地 廠房 帆布

[6] Logic behind this:

1. The definition of words is still an unsolved issue in Chinese linguistics. The fact is, many of the strings in Chinese are perceived as words by native speakers simply because of either their frequent occurrences or the long history they have been used. This implies that the definition of words in practice is inevitably statistical and subject to personal opinion. However, in an annotation task for word segmentation, it is more important to ensure the consistency of the results than to conform to people's intuition towards the language.
2. Unlike the case of compounds with 3 or more syllables, the components of a meaningful disyllabic string are two single characters, which means its internal interaction happens only in the character level. The fact that the combination of characters to form meaningful strings is much more restricted than that of multisyllabic string indicates that meaningful disyllabic strings usually are fixed patterns and have transferred meanings. Based on the principle of compositionality, we should therefore treat such strings as single tokens.
3. There are less than 20% of the word types in Chinese are monosyllabic, while more than 50% of which are disyllabic. Treating disyllabic strings as single tokens is more consistent with the lexicographic conventions.

[7] Please note that we do make the distinction between N and NR. For a string that contains a NR subcomponent, we generally segment it into multiple tokens.

[8] Treating disyllabic AD+VV as a single token should not override cross constituency constraint for ADs; the rationale behind is that ADs are relatively not frozen in an AD+VV string. This does not include conjunctive adverbs and conjunctive-like adverbs, which should still be separated. See section previous sections .

[9] Please note that constituency should be taken into consideration.

- NN+NNB(名+量): 車輛 船隻 雞隻
- JJ+NN(形+名): 藍籌 佳釀 美人
- JJ+NNB(形+量): 高層 大幅 小件
- VV+VV(動+動): 提取 打理 發送
- VV+Coverb(動+助動)[10]: 見到 變成 看似
- VV+Complement(動+補): 打碎 吃掉 擦淨
- VV+LC(動+處): 如下 綜上 帶上
- VV+Ci(動+"此"): 依此 照此 在此[11]
- VV+NN/NNB(動+名/量): 辭職 流血 在家
- VV/VC/MD+Fou(動+"否"): 有否 是否 能否
- JJ+JJ(形+形): 美好 金黃 優雅
- AD+JJ(副+形): 很好 極難 最新
- Zhi+LC("之"+處): 之前 之後 之外
- DT+NN(定+名) 本校 該市 各界
- DT+NNB(定+量) 翌日 當天 該名
- OD+NN(序+名): 次席 次等 次車
- OD+NNB(序+量): 首屆 首位 首個
- Name+Title(姓氏+稱謂): 李氏 陳總 張導
- PFX+Name(前綴+姓氏): 小李 老王 大羅

Strings of 3 Syllables

- Modifier-head structure
  - Segment as multiple tokens.
    - 1+2: e.g. 副|總統 總|指揮 前|首相
    - 2+1: e.g. 籃球|鞋 太陽|系 物理|學 研究|室 現代|化
- Coordinating structure
  - Segment as multiple tokens.
    - 上|中|下
    - 紅|黃|藍

---

[10] 在 is an exception even though it is considered as a coverb. See section 2.3.2.14.

[11] Those not explicitly listed as IN are now treated as VV in VV+X, e.g. 在家 到位 在此. In addition, the list of prepositions is used to solve the VV vs. IN ambiguity. If a preposition-like token is not in the list, it is treated the same way as VV. i.e. The following disyllabic examples are treated as single tokens: 不在(Not+VV), 在上(VV+LC), 到此(VV+Ci), 到家(VV+NN).

- Flat structure
    - Generally this category stands for proper nouns. Treat as a single token.
        - 伊拉克
        - 新加坡

Strings of 4 Syllables or More

- Unless specified in previous sections, segment the string according to its internal structure.
    - 物理|學|家
    - 反對|自由|貿易|主義|者
    - 世界|衛生|組織|總部

Modal Verbs (MD)

- Unless specified in previous sections, treat monosyllabic MD as single tokens[12].

AD (Other than Conjunctive Adverb)

- Unless specified in previous sections, treat as a single token.

Nested Structures[13]

- Segment the string according to its internal structure: combine disyllabic substrings as tokens as much as possible if the meaning of resulted disyllabic substring is exactly or in some context equal to its expanded form; otherwise, for the shared character in two spans, consider it as belonging to the left span and analyze accordingly with rules described in previous sections.

---

[12] There are exceptions that MD should not be treated as single tokens. These should serve as examples instead of a closed list, given that light parsing should be clear enough to treat X+MD or MD+X as a single token.

[13] In a nested structure, there are characters sitting in two adjacent spans due to the omission of duplications. We hence face the problem of deciding which span a shared character should belong to. According to this guideline, there are two different approaches: One is to combine disyllabic substrings whenever possible; the other is to consider shared characters as always belonging to one of the two spans, so that further analysis can be made. The former approach could sometimes change the meaning of the phrase (consider the example of "心臟病發"), while the later could introduce difficulties in the succeeding syntactic processings. We therefore adopt a hybrid strategy here.

- 大連 | 市長 (大連 = 大連市)
- 生態 | 學者(生態 = 生態學 in some context)
- 心臟 | 病 | 發 (心臟 ≠ 心臟病)

Terminology/Technical Terms

- For a terminology with the form "proper noun + common noun", first segment it into two sub-components, then segment the common noun part according to previous sections.
  - 切比雪夫 | 不等 | 式
  - 伽馬 | 射線
  - 哥德爾 | 不 | 完備 | 定理
- Scientific names of creatures: treat as a single token.
  - 丹頂鶴
  - 始祖鳥
  - 短吻鱷
- Terminologies in chemistry/biology/medicine: if a corresponding chemical formula exists, treat as a single token.
  - 三氧化二鐵 (Fe_2O_3)
  - 三硝基甲苯 ($C_7H_5N_3O_6$)
  - 硝酸甘油 ($C_3H_5N_3O_9$)
- Otherwise segment according to the word boundaries in English (don't treat a hyphen as word boundary).
  - 脱氧核糖核 | 酸 (deoxyribonucleic acid)
  - 鹼基 | 對 (base pair)
  - 核苷酸 (nucleotide)

2.1.7 Treatments of Hard Cases

We list some ambiguous cases below with their segmentation treatments. Unless otherwise specified, these treatments are according to the rules described in previous sections 2.1.5 and 2.1.6.

- Issue: complex Suffix Structures
  - Segment the string according to its internal structure.
    - E.g. 新 | 自由 | 主義 | 論 | 者

- Issue: 21 世紀
    - Segment into Cardinal Number + Time: 21 | 世紀

- Issue: 他自己
    - Segment the string: 他 | 自己

- Issue: 想想看 想一想 想了想 想不想 想了一想 想了又想 想了這麼一想
    - 想想看: Segment the string: 想想 | 看
    - 想一想: Treat as a single token.
    - 想不想: According to section rule for "A+Not+A" treat as a single token.
    - 想了想，想了一想，想了又想，想了這麼一想: segment the string.
        - 想 | 了 | 想
        - 想 | 了 | 一 | 想
        - 想 | 了 | 又 | 想
        - 想 | 了 | 這麼 | 一 | 想

- Issue: 對外 對中 對台
    - 對 is not listed as IN in Appendix, so treat it as VV.
        - 對外 (disyllabic VV+LC)
        - 對 | 中
        - 對 | 台

- Issue: 兩國 多國
    - Segment the string (CD+NN).
        - 兩 | 國
        - 多 | 國

- Issue: pseudo idioms[14]: 如前所述 如下所示 如圖 1 所示 綜上所述 由此可知 迎上前去 深以為然

[14] We define "pseudo idioms" as those patterns that have the ability to productively form new expressions by substituting some of their elements, however, the origin of these "pseudo idioms" cannot be found.

- Segment the strings according to their internal structures.
  - 如|前|所|述
  - 如|下|所|示
  - 如|圖|1|所|示
  - 綜|上|所|述
  - 由|此|可|知
  - 迎|上前|去

- Issue: VV+NN (VV+NR) as verb: the effects of word order: 獲獎學者 or 學者獲獎? 旅英學者 or 學者|旅英?
  - The order of strings is not a factor in making segmentation decisions.
    - E.g. 獲獎|學者 學者|獲獎
    - E.g. 旅|英|學者 學者|旅|英
  - Note: please notice the difference in the segmentation decisions for "獲獎" and "旅英": the former is a disyllabic VV+NN string while the latter is a disyllabic VV+NR string, therefore we treat the former as one token while segment the latter.

- Issue: 越來越
  - Treat as a single token. For those strings having similar meanings and grammatical functions, treat them as single tokens as well, e.g. 愈，愈發，愈加.

- Issue: 越戰越勇 越跑越快 愈演愈烈
  - Since these strings are productive pseudo idioms, we segment them into multiple tokens.
    - 越戰|越勇
    - 越跑|越快
    - 愈演|愈烈

- Issue: Productive Prefix: **副主席**
  - Segment the string.
    - 副|主席

- Issue: Productive Suffix: 管理員
  - Segment the string.
    - 管理|員

- Issue: 第三者, 進一步
  - Segment the string.
    - 第三|者
    - 進|一|步

- Issue: 有利於
  - According to its surface structure, we first segment it as: 有利|於; for the first part "有利", we treat it as one token.

- Issue: 並未，及其，而又，各個
  - 並|未(CC+VV)
  - 及|其(CC+PRON)
  - 而|又(CC+CC)
  - 各個(DT+NNB)

- Issue: **一員，一份子，一部分，一小部分**
  - 一員: CD+NN (as in "他是我們的一員") or CD+NNB (as in "一員大將"), in both cases we segment the string.
  - 一份子, 一部分, 一小部分: segment the strings according to the internal structures.
    - 一|份子
    - 一|部分
    - 一|小|部分

- Issue: **做不好， 做得不好，做得不太好**
  - 做|不|好
  - 做|得|不好
  - 做|得|不|太好

2.2 Part-of-Speech Tagset

To perform POS tagging re-annotation on CTB5 together with our proposed word segmentation approach, we use a POS tagset which is derived from the one used in the original CTB5 annotation. We show the tagset in Table 3 with comparison of number of occurrences of each tag in the original CTB5 and the re-annotated version, respectively. The tagset introduces several changes: first, we eliminate the use of the "LC" tag for locative words. This tag is assigned to all words that indicate locations and directions, such as 上 (up), 下(down), 左 (left), 右 (right), 内 (inside), 外 (outside) etc. We instead tag these words based on their real syntactic roles in sentences, such as "NN" (noun), "AD" (adverb) or "VV" (verb). Second, we add three new tags into the tagset for suffixes: "SFN" (nominal suffix), "SFA" (adjectival suffix), and "SFV" (verbal suffix). These tags are given to monosyllabic tokens appearing at the end of compounds, which are the bound morphemes in the traditional view. Based on our observation, these tokens have the ability to determine the syntactic role of the entire compound. For example, any compound that ends with a nominal suffix "度" (degree) always act as nouns in a sentence. It should be noted that because of this characteristic of suffixes, we can tag the children of suffixes in compounds based on their meaning but not their syntactic roles. We show some examples in Table 2 to illustrate our POS tagging strategy for compounds.

CTB5 Example	Re-annotation
副主席/NN (vice president)	副/JJ (vice) 主席/NN (president)
透明度/NN (transparency)	透明/JJ (transparent) 度/SFN (degree)
非生产性/NN (unproductiveness)	非/JJ (none) 生产/VV (produce) 性/SFN (property)
中央集权式/JJ (politically centralized)	中央/NN (center) 集权/NN (centralization) 式/SFA (type)

Table 2. Some examples of the word and POS annotation in the original CTB5 and our re-annotation.

Tag	Description	Count in CTB5	Count in proposed annotation
NN	Noun	134,321	137,816
PU	Punctuation	75,794	75,935
VV	Verb	68,789	75,033
AD	Adverb	36,122	35,922
NR	Proper Noun	29,804	30,985
P	Preposition	17,280	17,721
CD	Cardinal Number	16,030	21,493
M	Measure Word	13,668	18,091
JJ	Adjective	12,979	13,898

DEC	Complementizer	12,310	12,346
DEG	Genitive Marker	12,145	12,145
NT	Temporal Noun	9,467	4,524
LC	Locative	7,676	0
VA	Predicative Adjective	7,630	7,518
CC	Coordinating Conjunction	7,137	7,134
PN	Pronoun	6,536	6,646
DT	Determiner	5,901	5,970
VC	Copula	5,338	5,521
AS	Aspect Particle	4,027	4,033
VE	“you3” (“have”)	2,980	2,980
OD	Ordinal Number	1,661	1,661
MSP	Other Particles	1,316	1,316
ETC	“deng3” (“etc.”)	1,287	0
CS	Subordinating Conjunction	888	888
BA	Causative Auxiliary	751	756
DEV	Manner Marker	621	627
SP	Sentence-final Particle	466	466
SB	Short Passive Auxiliary	451	451
DER	Resultative Marker	258	258
LB	Long Passive Auxiliary	245	245

FW	Foreign Word	33	391
IJ	Interjection	12	17
X	Unknown	6	6
SFN*	Nominal Suffix	0	13,212
SFA*	Adjectival Suffix	0	438
SFV*	Verbal Suffix	0	129

Table 3. Tagset for part-of-speech tagging. The underlined characters in the examples correspond to the tags on the left-most column. The CTB-style word-level POS are also shown for the examples.

Label	Description	Example Phrase	Example Dependency
acomp	adjectival complement	他看上去很疲倦 (he looks very tired) 他跑得很快 (he runs very fast) 鞋是全新的 (the shoes are brand new)	**acomp**(看 looks, 疲倦 tired) **acomp**(跑 runs, 快 fast) **acomp**(是 are, 全新 brand new)
advmod	adverbial modifier	他看上去很疲倦 (he looks very tired) 大約十米左右寬 (about 10 m wide)	**advmod**(疲倦 tired, 很 very) **advmod**(米 m, 大約 about)
amod	adjectival modifier	漂亮的首飾 (cute accessory)	**amod**(首飾 accessory, 漂亮 cute)
appos	appositional modifier	總統奧巴馬 (president Obama)	**appos**(奧巴馬 Obama, 總統 president)
asp	aspect marker	他給了我一本書 (he gave me a book)	**asp**(給 gave, 了 -aspect-)
attr	attributive modifier	他是個醫生 (he is a doctor) 他是哪裡人 (where is he from)	**attr**(是 is, 醫生 doctor) **attr**(是 is, 人 person)
aux	auxiliary verb	必須解決 (must solve) 會議可能須要推遲 (meeting might need to be postponed)	**aux**(解決 solve, 必須 must) **aux**(推遲 postpone, 可能 might) **aux**(推遲 postpone, 須要 need)
auxpass	passive auxiliary	他被刺殺了 (he was assassinated)	**auxpass**(刺殺 assassinated, 被 -auxiliary-)
auxcaus	causative auxiliary	把問題解決(solve the problem)	**auxcaus**(解決 solve, 把 -auxiliary-)
cc	coordinating conjunction	聰明又可愛 (smart and cute) 他去過德國、法國和義大利 (he's been to Germany, France and Italy)	**cc**(聰明 smart, 又 and) **cc**(德國 Germany, 和 and)

ccomp	closed clausal complement	他說他喜歡游泳 (He said that he likes swimming)	**ccomp**(說 said, 喜歡 likes)
conj	conjunct	聰明又可愛 (smart and cute) 他去過德國、法國和義大利 (He's been to Germany, France and Italy)	**conj**(聰明 smart, 可愛 cute) **conj**(德國 Germany, 法國 France) **conj**(德國 Germany, 義大利 Italy)
csubj	clausal subject	能夠代表祖國參賽是他的夢想 (being able to play in the game for his country is his dream)	**csubj**(是 is, 參賽 play)
csubjpass	clausal passive subject	他在考試中作弊被老師發現了 (that he cheated during the exam is found out by the teacher)	**csubjpass**(發現 find out, 作弊 cheet)
dep	undefined dependency	添加一個日程安排時間星期二地點 3 樓 (add an event, time Tuesday, location 3rd floor)	**dep**(時間, 添加)
det	determiner	那本書 (that book)	**det**(本 -measure-, 那 that)
discourse	discoursal modifier	唉，終於到星期五了 (oh, thank God it's Friday)	**discourse**(到 is, 唉 oh) **discourse**(到 is, 了-sentence-final particle-)
dislocated	dislocated modifier	書是他買的 (book he bought) 這場火幸虧消防隊來得早. (this fire, fortunately the firefighters came in time)	**dislocated**(書 book, 買 buy) **dislocated**(火 fire, 來 come)
dobj	direct object	買了一本書(bought a book) 把問題解決了 (solved the problem)	**dobj**(買 bought, 書 book) **dobj**(解決 solved, 問題 problem)
foreign	foreign compound	職棒大聯盟(Major League Baseball)新賽季將擴軍 (Major League Baseball is	**foreign**(Major, League) **foreign**(Major, Baseball)

		going to expand in the new season)	
iobj	indirect object	他給了我一本書 (he gave me a book)	**iobj**(給 gave, 我 me)
list	list relation	添加一個日程安排時間星期二地點 3 樓 (add an event, time Tuesday, location 3rd floor)	**list**(時間 time, 地點 location)
mark	clause marker	他把信件給我之後就走了 (he left after he gave me the letter)	**mark**(給 give, 之後 after) **mark**(走 leave, 就 then)
		他整天不是吃就是玩 (he either eats or plays for the entire day)	**mark**(吃 eats, 不是 either) **mark**(玩 play, 就是 or)
mes	measure relation	一本書(a book)	**mes**(書 book, 本 -measure-)
ncomp	nominal complement	九成以上 (over 90%) 坐在椅子上 (sit on a chair) 改革中 (during the reform) 大約十米左右寬 (about 10 m wide)	**ncomp**(九成 90%, 以上 over) **ncomp**(椅子 chair, 上 -complementizer-) **ncomp**(改革 reform, 中 -complementizer-) **ncomp**(米 m, 左右 about)
neg	negation modifier	不擅長 (be not good at)	**neg**(擅長 be good at, 不 not)
nn	noun compound modifier	原油期貨價格 (oil futures price)	**nn**(價格 price, 原油 oil) **nn**(價格 price, 期貨 futures)
npadvmod	noun phrase adverbial modifier	大約十米左右寬 (about 10 m wide)	**npadvmod**(寬 wide, 米 m)
nsubj	nominal subject	他給了我一本書 (he gave me a book)	**nsubj**(給 gave, 他 he)
nsubjpass	passive nominal subject	他被刺殺了 (he was assassinated)	**nsubjpass**(刺殺 assassinated, 他 he)
num	numeric modi-	一本書 (a book)	**num**(本 -measure-, 一 a)

	fier		
p	punctuation	梨、橘子和香蕉 (pears, oranges, bananas)	**p**(梨 pears, 、)
pcomp	prepositional complement	由於路上人太多，我遲到了 (because it was so crowded, I was late)	**pcomp**(由於 because, 太多 so crowded)
pobj	prepositional object	他坐在椅子上 (he sits on a chair)	**pobj**(在 on, 椅子 chair)
ps	associative marker	這是我的家 (this is my home)	**ps**(我 me, 的 's)
poss	possessive modifier	這是我的家 (this is my home)	**poss**(家 home, 我 me)
prep	prepositional modifier	他坐在椅子上 (he sits on a chair) 這種藥提取自植物 (this medicine is extracted from plants)	**prep**(坐 sits, 在 on) **prep**(提取 extracted, 自 from)
prt	phrasal verb particle relation	他們打起來了 (they started a fight) 他看上去很疲倦 (he looks very tired) 他跑得飛快 (he runs very fast) 他飛快地跑 (he runs very fast) 這是他們所不能想像的 (this is what they can't imagine)	**prt**(打 fight, 起來 -auxiliary-) **prt**(看 looks, 上去 -auxiliary-) **prt**(跑 runs, 得 -particle-) **prt**(跑 runs, 地 -particle-) **prt**(想像 imagine, 所 particle)
		房間打掃完了 (room is all cleaned) 把數據整理成報告 (summarize the data into a report)	**prt**(打掃 clean, 完 finish) **prt**(整理 summarize, 成 become)
rcmodrel	relative clause complemen-	他回來的時候 (by the time he came back)	**rcmodrel**(回來 come back, 的 -complementizer-)

	tizer	這本是他買的書 (this is the book he bought)	**rcmodrel**(買 buy, 的 -complementizer-)
rcmod	relative clause modifier	他回來的時候 (by the time he came back)	**rcmod**(時候 time, 回來 come back)
		這本是他買的書 (this is the book he bought)	**rcmod**(書 book, 買 buy)
suff	suffix relation	科技界 (sci-tech industry)	**suff**(界 industry, 科技 sci-tech)
tmod	temporal modifier	他回來的時候天已經亮了 (it was bright outside by the time he came back)	**tmod**(亮 bright, 時候 time)
topic	topic marker	這本書是他買的 (this is the book he bought)	**topic**(書 book, 是 is)
		這是他們所不能想像的 (this is what they can't imagine)	**topic**(這 this, 是 is)
vmod	verb modifier	他打開門發現屋裏有人 (he opened the door and found out there is somebody inside)	**vmod**(發現 found out, 打開 open)
xcomp	open clausal complement	他不喜歡打網球 (he doesn't like to play tennis)	**xcomp**(喜歡 like, 打 play)

Table 4. Proposed dependency label set.

2.3 Dependency Label Set

In this section we present a label set which defines 45 dependency relations for Chinese sentences. This label set is shown in Table 4.

2.3.1 Chinese Specific Labels

dislocated The label "dislocated" is originally defined in the *universal dependencies* for languages such as Japanese to describe the syntactic relation of words in a topic–comment structure, but is not defined for Chinese. However, in Chinese it is frequent to see the topic–comment structure in a sentence, for example:

1. 這/this 本/-measure- 書/book 他/he 買/buy 的/-particle- (This book, he bought it)

In this sentence, 这本书 (this book) is the topic and 他买的 (he bought) is the comment. One common view of the syntactic structure of this sentence is that, 他 (he) is the subject of the predicate 他 (buy), and 书 (book) is the direct object. This treatment sees a topic–comment structure as having an OSV (object-subject-verb) word order, which is acceptable; it however has some problems in certain cases, for example:

2. 這/this 本/-measure- 書/book 他/he 買/buy 的/-particle- 昨天/yesterday 不見/disappear 了/-particle- (This book that he bought disappeared yesterday)

In this sentence, 书 (book) is still the direct object of 买 (buy), while it is also the subject of 不见 (disappear). Because of the nature of the dependency grammar we adopted, for such a structure we would have to choose one relation for 书 (book), either "nsubj" or "dobj", and discard the other relation which would cause a loss of the syntactic information encoded in the parse tree.

Moreover, the OSV word order cannot explain all topic-comment structure such as the following example:

3. 這/this 場/-measure- 火/fire 幸虧/fortunately 消防/firefighting 隊/team 來/come 得/-particle- 早/early (This fire, fortunately the firefighters came in time)

Unlike in the other two examples, the topic here, 這場火 (this fire), is not the direct object of the verb in the comment, 幸虧消防隊來得早 (fortunately the firefighters came in time).

To overcome these difficulties, we employ a different view which treats the topic–comment structure as having double subjects in a SSV word order. We define the first subject, 这本书 (this book) in example 2, as the head in a "dislocated" relation, and the subject-verb phrase, 他买的 (he bought) in example 2, as the modifier. The head in this dislocated relation can then form a "nsubj" (nominal subject) relation with the main predicate of the sentence, 不见 (disappear). Similarly, in example 3, the topic and the comment still form a dislocated relation even though the topic is not a direct object of the verb in the comment.

prt and ***prep*** We define the "prt" relation in two ways.

i. A relation between a verb and a particle. For example, 想像 (imagine) is the head in a "prt" relation of 所 (particle) in the sentence 這是他們所不能想像的 (this is what they can't imagine).
ii. A relation between a verb and its succeeding complement. For example, 打掃 (clean) is the head in a "prt" relation of 完 (finish) in the sentence 房間打掃完了 (the room has been cleaned).

We use the "prt" relation in the second case to capture the predicate-complement structure in Chinese. The verb 完 (finish) in the second example above functions to complement the meaning of the main verb, 打掃 (clean), and the sentence is still grammatical when the complement verb is removed: 房間打掃了 (the room is cleaned).

The complement verb sometimes also functions as a coverb in a serial verb construction, which takes its own direct object. For example:

4. 把/-auxiliary- 數據/data 整理/summarize 成/become 報告/report (summarize the data into a report)

Here the two verbs 整理 (summarize) and 成(become) form a “prt” relation, while they are the heads of 數據 (data) and 報告 (report) in the “dobj” relation.

A difficulty with labeling “prt” is that, it can be easily confused with the “prep” (prepositional modifier) relation. For example, one can argue that 成 (become) is a preposition instead of a verb and should be tagged as IN, so that the relation between 整理 (summarize) and 成 (become) would be "prep". To overcome this ambiguity, we apply a simple test: if the phrase headed by the word with a VV vs. IN ambiguity can be moved to a position before the main verb, then this word is a preposition and a prepositional modifier of the main verb; otherwise it is a verb. Here since the phrase “成 報告” (into report) cannot be moved to the position before 整理 (summarize), it should in fact be a verb phrase, not a prepositional phrase.

suff We define the suffix relation in a compound which has a “stem-suffix” structure. The suffix word with a POS tag SFN, SFA, or SFV is the root of the subtree formed by the words in the compound. It has one and only one child in this subtree, which is the head of the “stem”, and the dependency relation between them is labelled as “suff”.

The motivation of employing the “suff” label is to relieve the data sparseness problem of word forms in annotated corpora. Compounds, especially those with a “stem-suffix” structure, is a major source of new words in the Chinese language. These compounds, however, often share a set of suffix words which has a limited amount of instances. We think it is more effective for a parser to learn from features with word forms by treating the suffix words as the heads of compounds.

2.4 Evaluation

In this section we demonstrate the evaluation results of our annotation approach in word segmentation, POS tagging and machine translation experiments. We show that compared to the original CTB5, our re-annotation has the advantages of better data consistency and lower OOV word ratio, from which both a morphological analysis system and an extrinsic machine translation system can benefit.

2.4.1 Re-annotated Corpus

We re-annotated the entire CTB5 with our proposed word segmentation and POS tagging annotation strategies. The inter-annotator agreement on a set-aside dataset (first 100 sentences in files 301-325) is 99.10% for segmentation and 98.37% for POS tagging.

Table 5 shows some of the statistics of the original and the re-annotated CTB5. We split CTB5 in the same data division as in previous studies (Jiang et al., 2008a; Jiang et al., 2008b; Kruengkrai et al., 2009; Zhang and Clark, 2010; Sun, 2011). The training, development and test set have 18,089, 350 and 348 sentences, respectively. Compared to the original CTB5, the re-annotated training set has a lower percentage of unknown words and unknown word-POS pairs found in the corresponding test set. This is consistent with our observation that compounds with internal structures are one of the major sources of OOV words.

	CTB5	Re-annotated
Number of tokens	493,938	516,581
Avg. token length	1.63	1.55
Ratio of unknown words	14.67%	12.82%
Ratio of unknown word-POS pairs	15.02%	13.28%

Table 5. Statistics of the original CTB5 and our re-annotated version.

(a) Word Segmentation Results

Corpus	P	R	F
Original	97.48	98.44	97.96
Re-annotated	97.65	98.92	98.28
Re-annotated-partial	97.63	98.70	98.16

(b) Joint Segmentation and POS Tagging Results

Corpus	P	R	F
Original	93.01	93.95	93.48
Re-annotated	94.04	94.89	94.46

Table 6. Experimental results for morphological analysis on CTB5.

We further re-annotated 3,000 sentences which are randomly sampled from the training set of CTB5 using our proposed dependency label set. This re-annotated set is compared with the same sentences with the original annotation in a machines translation experiment in section 2.4.3.

2.4.2 Morphological Analysis Experiments

We compared the performance of a state-of-the-art joint word segmentation and part-of-speech tagging system (Kruengkrai et al., 2009) on the original and our re-annotated CTB5. We used the position-of-character (POC) tagset and the baseline feature set described in (Shen et al., 2014).

We trained all models using the averaged perceptron (Collins, 2002), which is an efficient and stable online learning algorithm. The models applied on all test sets are those that result in the best performance on the dev sets. To learn the characteristics of unknown words, we built the system's lexicon using only the words in the training data that appear at least 2 times.

We use precision, recall and the F-score to measure the performance of the systems. Precision (P) is defined as the percentage of output tokens that are consistent with the gold standard test data, and recall (R) is the percentage of tokens in the gold standard test data that are recognized in the output. The balanced F-score (F) is defined as $\frac{2 \cdot P \cdot R}{P+R}$.

We compared the performance of the morphological analyzer on the original and the re-annotated CTB5. The results of the word segmentation experiment and the joint experiment of segmentation and POS tagging are shown in Table 6(a) and Table 6(b), respectively. Each row in these tables shows the performance of the same system trained on the corresponding corpus.

For "Re-annotated-partial" in Table 6(a), we applied a different setting in order to directly compare the annotation consistency and data sparsity between the two corpora: we used the training set from the re-annotated corpus to train the system but the test set from the original corpus in the evaluation. To make the evaluation meaningful, we added an extra criterion when calculating the precision and the recall: if the outmost boundaries of a sequence (two or more) of output tokens are con-

sistent with a token in the test set, we consider that the output correctly identifies this token in the test set.

The results show that, the morphological analyzer can obtain higher accuracies in both word segmentation (0.32 points absolute in F-score) and joint (0.98 points absolute in F-score) experiments. Furthermore, in the word segmentation experiment "Re-annotated-partial" where we mapped the output of the system which is trained using the re-annotated training data to the original CTB5 test set, the accuracy is significantly higher than that of the "Original", which demonstrates the better consistency in our re-annotation corpus.

2.4.3 Machine Translation Experiments

To show that a morphological analysis system and a dependency parsing system can both benefit from our re-annotation, we conducted two sets of Chinese-to-Japanese machine translation experiments where a morphological analyzer and a dependency parser are used respectively.

The parallel corpus we used is the Chinese-Japanese part of the Asian Scientific Paper Excerpt Corpus (ASPEC)[15], containing 672k sentence pairs. We used 2,090 and 2,107 additional sentence pairs for tuning and testing, respectively.

In the first set of experiments, we segmented the Japanese sentences using JUMAN (Kurohashi et al., 1994), and the Chinese sentences using the same morphological analyzer described in the last subsection. For decoding, we used the state-of-the-art phrase based statistical machine translation toolkit Moses (Koehn et al., 2007) with default options. We trained the 5-gram language models on the target side of the parallel corpora using the SRILM toolkit[16] with interpolated Kneser-Ney

[15] http://lotus.kuee.kyoto-u.ac.jp/ASPEC/

[16] http://www.speech.sri.com/projects/srilm

discounting. Tuning was performed by minimum error rate training (MERT) (Och, 2003), and it was re-run for every experiment.

In the second set of experiments, we used the same morphological analyzers to segment and tag the POS of Japanese and Chinese sentences as in the first set. We further parsed the dependency structures of the Japanese sentences using KNP (Kawahara and Kurohashi, 2006), a lexicalized probabilistic dependency parser, and for the Chinese sentences we used a second-order graph-based parser proposed in (Shen et al., 2012). For decoding, we used the tree-to-tree example-based machine translation framework KyotoEBMT[17] (Richardson et al., 2015) with default options.

System	BLEU-4
Character	31.60
Original	31.46
Re-annotated	**32.08**

Table 7. Experimental results for Chinese-Japanese machine translation on ASPEC corpus using Moses system.

System	BLEU-4
Original	32.00
Re-annotated	**32.97**

Table 8. Experimental results for Chinese-Japanese machine translation on ASPEC corpus using KyotoEBMT system.

We report results on the test set using BLEU-4 score. The significance test was performed using the bootstrap resampling method proposed by Koehn (2004).

[17] http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?KyotoEBMT

In Table 7 we compare the performance of three Moses models: in "Character" we used a simple segmentation strategy for the Chinese sentences where we treated each character as a token; in "Original" and "Re-annotated" we segmented the Chinese sentences using the corresponding models described in the last subsection. The results show, with the underlying machine translation system being the same, the segmenter trained with the original CTB5 failed to support the system to outperform the simple character-based segmentation, while on the other hand the system using the segmenter trained with our re-annotated CTB5 significantly outperformed both "Character"[18] and "Original"[19].

In Table 8 we show the result of the experiment with KyotoEBMT, a tree-to-tree machine translation system which requires unlabeled dependency annotation in the model training. We show that, the model "Re-annotated" which used the training set with the proposed annotation, it significantly outperformed[20] the baseline model "Original" by 0.97 point in BLEU-4 score.

2.5 Summary

In this chapter we have proposed a new annotation approach for Chinese word segmentation, part-of-speech tagging, and dependency labeling. By re-annotating the CTB5 and conducting word segmentation, POS tagging and machine translation experiments, we have demonstrated that this approach has the advantages in achieving higher annotation consistency as well as less data sparsity, compared to the original annotation of CTB5.

---

[18] $p < 0.5$

[19] $p < 0.1$

[20] $p < 0.1$

Chapter 3

Chinese Word Segmentation and Unknown Word Extraction by Mining Maximized Substring

Chinese sentences are written without explicit word boundaries, which makes Chinese word segmentation (CWS) an initial and important step in the Chinese language processing. Recent advances in machine learning techniques have boosted the performance of CWS systems. However, previous studies have shown that with a comprehensive lexicon, even a simple maximum matching segmentation algorithm can yield an F-score as high as 0.99 (Sproat and Emerson 2003). This suggests that the lack of knowledge of vocabulary presents the biggest challenge in Chinese word segmentation. It is impossible, however, to collect a complete list of Chinese words. The Chinese language is continually and rapidly evolving, particularly with the rapid growth of the internet. Therefore, it is necessary to develop techniques that automatically develop vocabulary lists from large-scale web texts instead of relying on supervised CWS systems.

Recent research has attempted to exploit characteristics of frequent substrings in unlabeled data. Statistical criteria that measure the likelihood of a substring being a word have been proposed in previous studies of unsupervised segmentation, such as “accessor variety” (Feng et al. 2004), a criterion measuring the likelihood of a substring being a word by counting distinct surrounding characters. In (Jin and Tanaka-Ishii 2006) the researchers proposed “branching entropy”, a similar criterion based on the assumption that the uncertainty of surrounding characters of a sub-

string peaks at the word boundaries. In (Ye et al. 2013), the likelihood of an character n-gram being a meaningful Chinese word is measured by "overlap variety", a criteria which considers the goodness of the candidate together with the goodness of the strings overlapping the candidate. The authors have also adopted a word refinement module to filter out known words during the post-processing.

The authors of (Zhao and Kit 2007) incorporated accessor variety and another type of criteria, called "co-occurrence sub-sequence", with a supervised segmentation system and conducted comprehensive experiments to investigate their impacts. There are several restrictions in using the co-occurrence sub-sequence criteria: it requires post-processing to remove overlapping instances; sub-sequences are retrievable only from different sentences; and the retrieval is performed only on training and testing data. In (Sun and Xu 2011), the authors proposed a semi-supervised segmentation system enhanced with multiple statistical criteria; large-scale unlabeled data were used in their experiments. Jiang et al. (2010) used term contributed boundary information extracted from unlabeled text as features in semi-supervised learning to improve character-based Chinese word segmentation.

Substring	Count
一致	3
界限数的期望值	2
一致认定界限	2
的期望值	3
认定界限数的	2
值	4

Table 9. A particular type of substrings with multiple occurrences in the Chinese sentence: "使一致认定界限数的期望值近似于一致正确界限数的期望值，求得一致认定界限的期望值/认定界限数的值。"

Li and Sun presented a model to learn the features of word delimiters from punctuation marks in (Li and Sun 2009). Wang et al. proposed a semi-supervised word segmentation method that took advantages from auto-analyzed data (Wang et al. 2011).

Nakagawa showed the advantage of the hybrid model combining both character-level information and word-level information in Chinese and Japanese word segmentation (Nakagawa 2004). In (Nakagawa and Uchimoto 2007) and (Kruengkrai et al. 2009a; 2009b) the researchers presented word-character hybrid models for joint word segmentation and POS tagging, and achieved the state-of-the-art accuracy on Chinese and Japanese datasets.

Another related research area is unknown Chinese word extraction. Previous studies have explored statistical criteria for measuring the likelihood of a substring being a word (Feng et al. 2004), methods that combine morphological rules with statistical information (Chen and Ma 2002; Ma and Chen 2003), character-based role tagging (Zhang et al. 2002), and generating words from an initial segmentation (Zhou 2005; Goh et al. 2003; Goh et al. 2005). Recent advances have also verified the ability to process large-scale web texts (Zhang et al. 2011).

In this chapter, we propose a novel method that extracts substrings as reliable word boundary indicators. Unlike existing methods which take quadratic or quasi-linear time in the average case due to the need of collecting frequency information for all possible substrings, our method takes linear time in the average case, which is a big advantage since the size of raw text for substring extraction is usually very large.

To illustrate the idea, we first consider the following example taken from a scientific article:

“使一致认定界限数的期望值近似于一致正确界限数的期望值，求得一致认定界限的期望值/认定界限数的值。”

Without any knowledge of the Chinese language one may still notice that some substrings like “一致” and “的期望值”, occur multiple times in the sentence and are likely to be valid words or chains of words. Consider a particular type of frequent substrings (the formal definition can be found in section 2.1), in the sense that all its occurrences have surrounding characters which are distinct from each other from both left and right-hand sides (Table 9). We can observe that the boundaries of such substrings can be used as cues for word segmentation. We then segment the sentence by simply matching the substrings in Table 9 in the sentence; each time there is an occurrence of a substring, we split the sentence at the boundaries of the detected occurrence. The resulted segmentation is as follows:

“使|一致|认定|界限|数|的|期望|值|近似于|一致|正确|界限数|的期望|值|，求得|一致|认定界限|的期望|值|/|认定界限数的|值|。”

Compared with the gold-standard segmentation, this segmentation has a precision of 100% and a recall of 73% with regard to boundary estimation. This is high when we consider that the method does not use a trained segmenter or annotated data. While we have obtained this result on a selected instance, it still suggests that unlabeled data has the potential to enhance the performance of supervised segmentation systems by tracking consistency among substrings.

There are some key properties of the substrings listed in Table 9: the substrings are frequent, in the sense that they appear multiple times in a relatively small window of text; also, any substring in the list doesn’t have a superstring that has the same frequency. These properties are studied in previous research. For example, (Nagao and Mori 1994) described these kind of substrings as "longest compound word" and successfully extracted words and phrases based on their properties from large scale Japanese text data. (Lin and Yu 2001) described these substrings as “reduced N-gram” and proposed a two-stage algorithm for extracting frequent substrings from a corpus and removing internal substrings of the same frequency; the

second step is called statistical substring reduction (SSR), which can be done in linear time by recent advance in its studies (Lü et al. 2004).

We exploit these substrings in two ways. First, similar to (Jiang et al. 2010), we incorporate the substrings as features in a semi-supervised CWS system. In this case the unlabeled data can be either test data only, or a large-scale external corpus. Second, we also refine the extracted substrings into a list of unknown Chinese words as a language resource, which can be easily used by other language processing applications. We will formally define this particular type of substring, referred to as a "maximized substring", in a later section.

In the remainder of this chapter, we will define maximized substrings and proposes an algorithm for extracting these substrings from unlabeled data; we will also describe our baseline segmentation system and introduces the maximized substring features; we will then describe our post-processing techniques for reducing the redundancy and noise in unknown word extraction; finally we will present the experimental results and discuss the relation and difference between our proposed method and previous work.

We summarize the main contributions of this work as follows.

- The frequent substring mining algorithm that we propose is the first algorithm that is able to extract maximized substrings in one scan. These substrings can be reliably used to estimate word boundaries. The algorithm is efficient for processing very large scale text, and its underlying data structure makes it suitable for incorporating with caching methods to further increase the quality of the output.
- We incorporate the extracted list of maximized substrings with a baseline Chinese word segmentation system and demonstrate its performance in relation to previous studies. We show that our method outperforms other state-of-the-art systems.

- We apply post-processing strategies on the extracted list of maximized sub-strings to extract unknown words. Our evaluation shows that the accuracy of our method significantly outperforms a previous study (Feng et al. 2004).

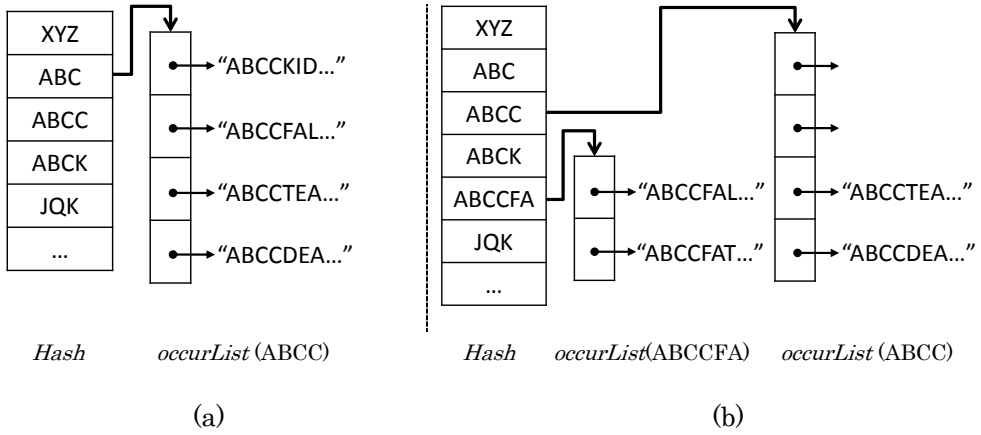


Figure 2. Data structure for maximized substring mining. *Hash* is a hash table which stores the detected maximized substrings, where each maximized substring s is a key that is associated with a list of its occurrences $occurList(s)$. $occurList(s)$ also stores the original postitions of all the occurrences of s in the document. (a) shows a certain state of the data structure, and (b) the state after a *maximized substring* "ABCCFA" is inserted with the context being "ABCCFAT…" in the document.

3.1 Maximized Substring Extraction

3.1.1 Maximized Substring: The Definition

Frequent substrings in unlabeled data can be used as clues for identifying word boundaries, as we have illustrated in the beginning of this chapter. Nevertheless, some substrings, although frequent, are not useful to the system. In the previous example, the substring "致认定界" occurs the same amount of times as the substring

"一致认定界限". However, only the latter is a valid identifier for word delimiters. The former substring is "internal": it is overlapped by other substrings and can be further extended by its surrounding characters, without reducing its frequency. What we are interested in are non-overlapping substrings, meaning that in each side of these substrings, the surrounding characters are all different from each other. We use the term *maximized substring* to describe these substrings.

Formally, we define a maximized substring as follows:

Definition 1 (Maximised substring). Given a document D that is a collection of sentences, denote a length n substring which starts with character c_t by $s_t = [c_t c_{t+1} \dots c_{t+n-1}]$, where c_t is the character at t–th position in D. s_t is called a *maximized substring* if:

1. It has a set of distinct occurrences, M, with at least two elements[21]:

   $\mathrm{M} = \{s_{t_1}, s_{t_2}, \dots, s_{t_k}\}$, $k > 1$, $t_1, t_2, \dots t_k$ are distinct from each other s.t. $s_{t_1} = s_{t_2} = \dots = s_{t_k}$; and

2. $c_{t_i-1} \neq c_{t_j-1}$ and $c_{t_i+n} \neq c_{t_j+n}$ $\forall i, j = 1,2 \dots, \mathrm{m}, i \neq j$.

The substrings listed in Table 9 are therefore maximized substrings, given that D is the example sentence. Note that these are not all maximized substrings extractable from the example sentence, but are the result of the extraction algorithm that we will describe in the next section.

[21] It should be noted that, in order to extract a substring, the size of M is not necessarily identical to its total count in the document.

3.1.2 Maximized Substring Extraction: Algorithm and Data Structure

The problem of mining frequent substrings in a document has been extensively researched. Data structures and substring mining strategies, such as suffix trees (Nelson 1996), suffix arrays (Yamamoto and Church 2001; Fischer et al. 2005), and Apriori property (Agrawal and Srikant 1994), have been used in existing algorithms. The Apriori property states that a string of length k+1 is frequent only if its substring of length k is frequent, which can significantly reduce the size of enumerable substring candidates. While suffix tree or suffix array-based algorithms can extract maximized substrings with the time complexity of $O(n \log n)$ (Yamamoto and Church 2001), we propose a novel algorithm for fast maximized substring mining to improve the efficiency in processing large scale data.

Algorithm 1: Maximized Substring Extraction

```
1  procedure ExtractMaxSub(D)
2    i ← 0
3    H ← ∅  ◁ H is a hash table of maximized substrings
4              ◁ each key in H is associated with a list of its occurrences
5    until i reaches the end of document D
6      s ← longest substring from position i which can be found in H
7      if |s| = 0  ◁ empty string
8        s_new ← [c_i] ◁ single-character string
9        occurList_new ← {i}
10       ◁ starting position of new occurrence
11       H.Add(⟨s_new, occurList_new⟩)
12       ◁ associate string s_new with its occurrence list and add to H
13       i ← i + 1
14     else
15       (H, i) ← Maximize(H, i, s)
16   return H
17
18 procedure Maximize(H, i, s)
19   t ← 0
20   for each u in H.Find(s) ◁ H.Find(s) returns the occurList associated with s
21     if c_{u+|s|} = c_{i+|s|}
22       while c_{u+|s|+t} = c_{i+|s|+t}
23         t ← t + 1
24       s_new ← [c_i c_{i+1} ... c_{i+|s|+t−1}]
25       occurList_new ← {i, u}
26       H.Add(⟨s_new, occurList_new⟩)
27       i ← i + |s| + t
28       return (H, i)
29   s.occurList.Add(i)
30   i ← i + |s|
31   return (H, i)
```

The data structure we used in this algorithm is illustrated in Figure 2. It supports fast prefix searching for storing and extracting maximized substrings, with each entry associated to a list of occurrences that refer to the original positions in the document. Fast prefix matching is a particular advantage of a trie, which is a type of prefix tree; however we choose a simple hash table structure in our implementation, which is mainly due to the frequent deletion operations required in the so-called "short-term store" technique, which we will describe in next section. A hash table data structure is efficient for the deletion operation in the sense that the average case complexity is $O(1)$, and it also has less computational overhead compared to a trie, since less data structure manipulation is needed after a deletion is made.

The hash table stores the detected maximized substrings, where each maximized substring s is a key that is associated with a list of its occurrences. The occurrence list also stores the original postitions of all the occurrences of a maximized substring in the document with the surrounding context being visible, so that new (longer) maximized substrings can be found by extension.

We sketch the process of maximized substring extraction in Algorithm 1. The data structure described above is corresponding to H in the algorithm. From the beginning of the document D, we scan each position and register maximized substrings into the data structure H. If an incoming substring already exists in H, we look up its occurrence list to check if its succeeding characters can extend the substring. As the current occurrence list is a set of maximized substrings, there will be only two possible outcomes. Either exactly one element in the occurrence list is found to have a longer common prefix with the incoming substring, in which case we create a new occurrence list consisting of the two longer substrings. Alternatively, the prefix remains the same and we add the incoming substring to the occurrence list. In both cases, the scanning skips the rest of the positions inside the newly added occurrence (line 27 and 30). This ensures that the algorithm will not extract any "internal" substrings.

The average case complexity of Algorithm 1 is $O(n)$, assuming the number of occurrences of each maximized substring is less than a constant C, such that $C \ll n$. The worst case complexity is $O(n^2)$, which happens when the document consists of a sequence of extractable occurrences of the same maximized substring s, e.g. $D = [x_1 s x_2 s x_3 s \dots x_N s]$, where $x_1, x_2, \dots x_N$ are distinct from each other. This is because of the block starts from line 20 in Algorithm 1 which requires a comparison operation for each occurrence, and in this example the number of occurrences of s is $n/2$. The "while loop" starts from line 22 however doesn't increase the complexity, since the searching always skip those characters that have already been checked in this loop and starts at the next unchecked position (line 27). However, the worst case can rarely happen when processing real world text, since the largest number of occurrences of any maximized substring is much smaller than $n/2$.

It can be demonstrated that a substring s with n characters extracted by this algorithm with more than one occurrence is a maximized substring:

Proof

1. s has more than one occurrence, which form a set M:

   $M = \{s_{t_1}, s_{t_2}, \dots, s_{t_k}\}$, $k > 1$, $t_1, t_2, \dots t_k$ are distinct from each other s.t. $s_{t_1} = s_{t_2} = \dots = s_{t_k}$.

2. Assume there exists two elements, s_a and s_b $(a < b)$, such that $c_{a+n} = c_{b+n}$ or $c_{a-1} = c_{b-1}$.

   a) If $c_{a+n} = c_{b+n}$, in line 22 of Algorithm 1 the character c_{b+n} will be appended to s_b, s.t. a string $s_{b'} = [c_b c_{b+1} \dots c_{b+n+t}]$ $(t \geq 0)$ will be extracted and added into the hash table, in which case s_b will not be extracted since it is a substring of $s_{b'}$ starting from the same position in the document.

   b) If $c_{a-1} = c_{b-1}$, let u be the largest positive integer s.t. $c_{a-i} = c_{b-i}$ $\forall i = 1,2, \dots, u$. Since $c_{a-u-1} \neq c_{b-u-1}$, Algorithm 1 visited position $b-u$ and string $s_{b''} =$

$[c_{b-u}c_{b-u+1} \dots c_{b+n+t-1}]$ $(t \geq 0)$ will be extracted in line 24, in which case s_b will not be extracted since it is a substring of $s_{b''}$.

By contradiction, there are no pair of elements s_a and s_b $(a < b)$, s.t. $c_{a+n} \neq c_{b+n}$ and $c_{a-1} \neq c_{b-1}$, therefore $c_{t_i-1} \neq c_{t_j-1}$ and $c_{t_i+n} \neq c_{t_j+n}$ $\forall i, j = 1,2 \dots, \mathrm{m}$, $i \neq j$.

By *Definition 1*, *s* is a maximized substring.

However, the algorithm does not generally guarantee to extract all maximized substrings from unlabeled data. Namely, in the example sentence we have shown in the beginning of this chapter ("使一致认定界限数的期望值近似于一致正确界限数的期望值，求得一致认定界限的期望值/认定界限数的值。"), "界限" *is* a maximized substring according to the definition (with the occurrences being "正确界限数" and "认定界限的"), but it is not extractable by Algorithm 1. This happens because after the algorithm extracted "一致认定界限", the new position for the next look-up becomes the end of this substring, and the second valid occurrence of "界限" is automatically omitted. We consider this as a necessary compromise in order to keep the efficiency of one-time scanning. To further investigate this issue, we have extracted all maximized substrings from the dev set of Chinese Treebank 7 (CTB7) which has 10,136 sentences. We used the following method in order to extract all maximized strings from a doc: first, we modified Algorithm 1 to force it to look-up from all possible positions regardless of whether a maximized string is found, which can be done by simply replacing line 30 ("i←i+|s|") with "i←i+1". Then, from the occurrence list of an extracted substring, we remove all the sets of occurrences if they can be extended from the left-hand side from their original positions in the document while still being equal to each other. Finally, we remove all the extracted substrings if their occurrence list has either zero or one instance left.

We compared the result with the maximized substrings extracted by Algorithm 1 from the same dataset. With Algorithm 1, we are able to extract 99.62% of the max-

imized substrings. This small difference suggests that the disadvantage of Algorithm 1 of incomplete extraction can be easily overcome by adding more raw sentences.

In addition, we have observed in preliminary experiments that extracting all maximized substrings is not only unnecessary, but can introduce harmful noise. In the next section, we will discuss our solution to this problem.

3.1.3 Short-Term Store

Maximized substrings can provide good estimations of word boundaries, but noise can be introduced during the extraction process in Algorithm 1.

To address this problem, we take advantage of a linguistic phenomenon. It has been observed that a word occurring in the recent past has a much higher probability to occur again soon, when compared with its overall frequency (Kuhn and Mori 1990). It follows that, for speech recognition, we can then use a window of recent history to adjust the static overall language mode.

This observation is applicable to the task of maximized substring extraction in the following way. Suppose a substring is registered into the data structure. If the substring is in fact a word, it is much more likely to reoccur in the next 50 to 100 sentences than in the remainder of the corpus (especially when it is a technical term or a named entity). Otherwise, the substring represents noise, and it should have a more sparse distribution in the corpus.

This motivated us to introduce a functionality into the process of maximized substring extraction, called “short-term store” (STS). The STS is an analogy to the cache component in speech recognition as well as the human phonological working memory in language acquisition. It restricts the length of the visible context when extracting the next occurrence of a registered substring, making it proportional to the current number of occurrences of the substring. For a registered substring, the extraction algorithm scans a certain number of sentences after the latest occurrence of the substring, where the number of sentences D(s) is determined as follows:

$$\mathrm{D}(s) = \begin{cases} \lambda \cdot count(s), & if\ count(s) < \theta, \\ \infty, & otherwise, \end{cases}$$

where $count(s)$ is the current number of occurrences of s in the data structure. The parameter λ contributes a fixed-length distance to the visible context. The parameter θ works as a threshold of reliability. If we have observed s at least θ times in a short period, we can regard s as a word, or a sequence of words, with a high level of confidence. Thus, $\mathrm{D}(s) = \infty$ implies that s is no longer subject to periodical decaying and will stay in the data structure statically.

During the scanning of the D(s) sentences, if a new occurrence of s is found, it is added into the data structure and D(s) is recalculated immediately, starting a new scanning period. If no new occurrences are found, we remove the earliest occurrence of s from the data structure and then re-calculate D(s). Note that although we have described the short-term store functionality as if each substring in the data structure is scanned separately, in practice we only need to make minor changes in Algorithm 1: each time an occurrence of a maximized substring is found, we associate a "time stamp" which is the order of the current sentence in the document with it. Then we perform the following two simple steps when the algorithm attempts to add this occurrence into the data structure.

- Step 1. If the occurrence list is empty, add the new occurrence into the list; otherwise the difference between time stamps of it and the most recent occurrence is calculated.
- Step 2. If the difference is smaller than D(s), add the new occurrence into the list; otherwise the occurrence lists pops out the most recent occurrence and go back to Step 1.

By making this modification to Algorithm 1, we can obtain the effects of STS without increasing the complexity.

Introducing STS into the extraction process results in a substantial improvement to the quality of extracted substrings. It is also important that STS greatly improves the processing efficiency for large scale unlabeled data by keeping the size of the data structure relatively small. This is because a substring entry will decay from the data structure if it has not been refreshed in a short period.

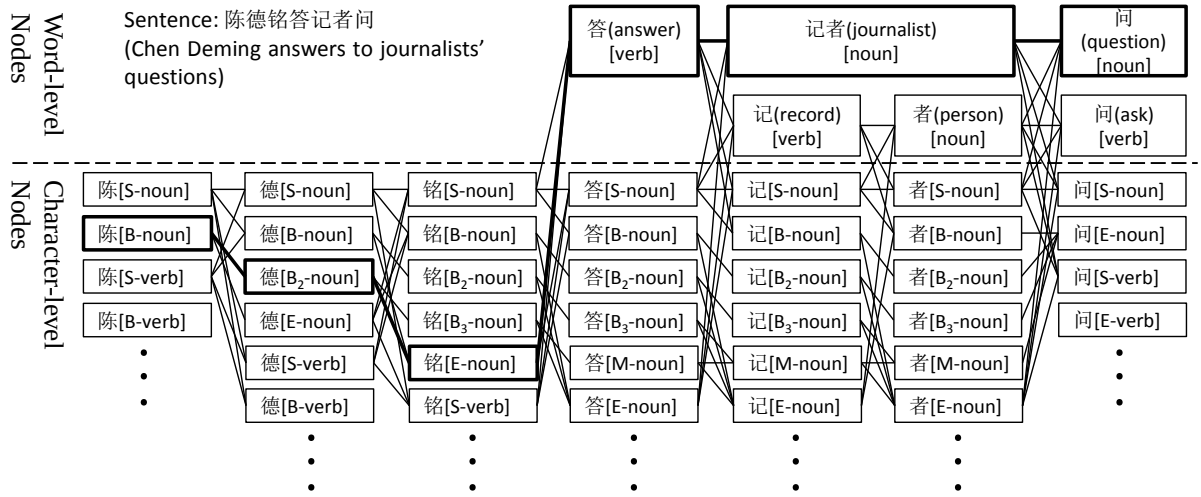


Figure 3. A Word-character hybrid lattice of a Chinese sentence. Correct path is represented by bold lines.

Word Length	1	2	3	4	5	6	7 or more
Tags	*S*	*BE*	*BB_2E*	*BB_2B_3E*	*BB_2B_3ME*	*BB_2B_3MME*	*$BB_2B_3M...ME$*

Table 10. Word representation with a 6-tag tagset: *S*, *B*, *B_2*, *B_3*, *M*, *E*.

Category	Template	Condition
Unigram	$\langle w_0 \rangle \langle p_0 \rangle \langle w_0, p_0 \rangle \langle l_0, p_0 \rangle \langle \text{begin}(w_0), p_0 \rangle \langle \text{end}(w_0), p_0 \rangle$	W_0
	$\langle \text{begin}(w_0), \text{end}(w_0), p_0 \rangle$	
	$\langle c_{-2}, p_0 \rangle \langle c_{-1}, p_0 \rangle \langle c_0, p_0 \rangle \langle c_1, p_0 \rangle \langle c_2, p_0 \rangle$	C_0
	$\langle c_{-2}, c_{-1}, p_0 \rangle \langle c_{-1}, c_0, p_0 \rangle \langle c_0, c_1, p_0 \rangle \langle c_1, c_2, p_0 \rangle \langle c_{-1}, c_1, p_0 \rangle$	
Bigram	$\langle w_{-1}, w_0 \rangle \langle p_{-1}, p_0 \rangle \langle w_{-1}, p_0 \rangle \langle p_{-1}, w_0 \rangle \langle w_{-1}, p_{-1}, w_0 \rangle \langle w_{-1}, w_0, p_0 \rangle$	$W_{-1} \times W_0$
	$\langle w_{-1}, p_{-1}, p_0 \rangle \langle p_{-1}, w_0, p_0 \rangle \langle w_{-1}, p_{-1}, w_0, p_0 \rangle \langle l_{-1}, p_{-1}, l_0 \rangle \langle l_{-1}, l_0, p_0 \rangle$	
	$\langle l_{-1}, p_{-1}, p_0 \rangle \langle p_{-1}, l_0, p_0 \rangle \langle l_{-1}, p_{-1}, l_0, p_0 \rangle \langle \text{end}(w_{-1}), p_0 \rangle$	
	$\langle p_{-1}, \text{begin}(w_0) \rangle \langle \text{end}(w_{-1}), p_{-1}, p_0 \rangle \langle p_{-1}, \text{begin}(w_0), p_0 \rangle$	
	$\langle c_{-1}, c_0 \rangle \langle p_{-1}, p_0 \rangle \langle c_{-1}, p_{-1}, c_0 \rangle \langle c_{-1}, c_0, p_0 \rangle$	$C_{-1} \times C_0$
	$\langle c_{-1}, p_{-1}, p_0 \rangle \langle p_{-1}, c_0, p_0 \rangle \langle c_{-1}, p_{-1}, c_0, p_0 \rangle$	
	$\langle p_{-1}, p_0 \rangle$	Otherwise

Table 11. Feature templates. The "Condition" column describes when to apply the templates: W_{-1} and W_0 denote the previous and the current word-level node; C_{-1} and C_0 denote the previous and the current character-level node; N_{-1} and N_0 denote the previous and the current node of any types. begin(·) and end(·) denote the first and last character in a word-level node, respectively. Word-level nodes represent known words that can be found in the system's lexicon.

3.2 Chinese Word Segmentation

3.2.1 Baseline Segmentation System

We have used a word-character hybrid model as our baseline Chinese word segmentation system (Nakagawa and Uchimoto 2007; Kruengkrai et al. 2009). This hybrid model constructs a lattice that consists of word-level and character-level nodes from a given input sentence. Word-level nodes correspond to words found in the system's lexicon, which has been compiled from training data. Character-level nodes have

special tags called position-of-character (POC) that indicate the word-internal position (Asahara 2003; Nakagawa 2004). We have adopted the 6-tag tagset, which (Zhao et al. 2006) reported to be optimal. This tagset is illustrated in Table 10.

Previous studies have shown that jointly processing word segmentation and part-of-speech tagging is preferable to separate processing, which can propagate errors (Nakagawa and Uchimoto 2007; Kruengkrai et al. 2009). If the training data was annotated by part-of-speech tags, we have combined them with both word-level and character-level nodes.

Figure 3 shows an example of a lattice for the Chinese sentence: "陈德铭答记者问" (Chen Deming answers to journalists' questions). The correct path is marked with bold lines. The upper part of the lattice (word-level nodes) represents known words from the system's lexicon, and the lower part of the lattice (character-level nodes) represents unknown words. A sequence of character-level nodes are considered as an unknown word if and only if the sequence of position-of-character tags forms one of the cases listed in Table 10. This table also illustrates the permitted transitions between adjacent character-level nodes. We have used a dynamic programming algorithm to search for the best path in the lattice.

Note that there are some character-level nodes that are not reachable. For instance, the first character of the sentence combined with a POC tag "B_2", or the last character of the sentence combined with a POC tag "M". We have omitted unreachable nodes from the lattice.

We apply the feature templates described in Table 11. Note that if the part-of-speech tags are not available, we omit those templates involving POS tags.

3.2.2 Maximized Substring Features

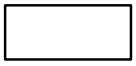
We have incorporated the list of extracted maximized substrings into the baseline system by using a technique which discriminatively learns their features. For every word-level and character-level node in the lattice, the method checks the maximized substring list for entries that satisfy the following two conditions:

(1) the node matches the maximized substring at the beginning, the end, or both boundaries;

(2) the length of the node is shorter than or equal to that of the entry.

For example, consider the lattice in Figure 3 with a maximized substring “陈德铭”. All of the character-level nodes of “陈” and “铭” are encoded with maximized substring features. A segmenter will only obtain information on those possible word boundaries that are identified by maximized substrings. The maximized substrings are not directly treated as single words, because a maximized substring can sometimes be a compound word or phrase.

For each match with a maximized substring entry, the technique encodes the following features on a node.

Sentence $s = \cdots c_{-1}c_0c_1 \ldots c_{n-1}c_nc_{n+1} \ldots$	Represen-tation
Maximized substring $x = c_0c_1c_2 \ldots c_n$	
Lexicon entry $l = c_ic_{i+1}c_{i+2} \ldots c_j$	

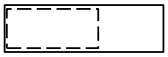
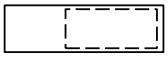
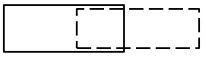
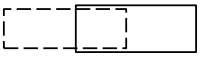
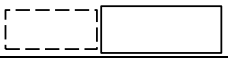
ID	Positional Relation	
L1	$0 = i < j < n$	
L2	$0 < i < j = n$	
L3	$0 = i < n < j$	
L4	$i < 0 < j = n$	
L5	$0 < i < n < j$	
L6	$i < 0 < j < n$	
L7	$i = n + 1$	
L8	$j = -1$	

Table 12. Lexicon features. Each one represents a positional relation between a maximized substring and a contextual substring which exists in system's lexicon.

ID	At Beginning	ID	At Ending
B1	<L1,L6>	E1	<L2,L5>
B2	<L6,L8>	E2	<L5,L7>
B3	<L1,L8>	E3	<L2,L7>

Table 13. Lexicon Composition features. Each one represents a combination of two Lexicon features that fire simultaneously.

Basic: a binary feature that indicates whether the node matches at the beginning or end of the maximized substring. It is encoded both individually and as a combination with each other feature types.

Lexicon: there is a particular kind of noise in the extracted list of maximized substrings, namely, those like the substring "中美经", which has resulted from the two phrases "中美经济" (China and U.S. economy) and "中美经贸" (China and U.S. economic and trade). This happens when the boundary of a maximized substring is a shared boundary character of multiple other words. In this example, the last character "经" of the maximized substring is the character at the beginning of "经济" (economy) and "经贸" (economic and trade). This kind of noise can be identified by checking the context of maximized substrings in system's lexicon.

For a node that matches the maximized substring, we check the context of the maximized substring in the input sentence and compares it with the system's lexicon. If any item in the lexicon is found that forms a positional relation with the maximized substring entry (as listed in Table 12) then the corresponding features are encoded on the node.

It should be noted that, the positional relations listed in Table 12 is in fact a subset of the thirteen interval relations defined in (Allen 1983); the omitted relations are: "equal", "before", "before-inverse", "during" and "during-inverse". "equal" means that the maximized substring matches a lexicon entry, in which case it becomes redundant; and in all the other omitted relations, the maximized substring and the lexicon entry don't have either shared or adjacent characters on their boundaries, which would give us little clue to indicate the reliability of a extracted maximized substring.

Lexicon Composition: when a maximized substring is a match to more than one item in the lexicon, a combination of multiple lexicon features is more informative than individual features. We encode the combinations of lexicon features listed as in Table 13.

Frequency: we sort the list of maximized substrings by their frequencies. If a maximized substring is among the 10% most frequent it is classed as “highly frequent”, if it is among the top 30% it is “normal”, and all other cases are “infrequent”.

3.3 Unknown Word Extraction

The extracted maximized substrings cannot be directly treated as words, for two reasons: first, many of the extracted substrings are chains of words, including compounds and short phrases; while this is fine in the task of word segmentation as long as they still correctly identify word boundaries, it is necessary to break them into atoms for word extraction. Second, random noise that violates word boundaries can be introduced during the extraction process. In this section, we introduce two post-processing strategies for unknown word extraction that are effective in filtering out redundant compound words, phrases, and substrings that violate word boundaries.

3.3.1 Redundancy Reduction

The extracted list of maximized substrings contains not only single words but also compound words and phrases, which are detrimental when extracting unknown words. With the help of a lexicon, we have applied a simple remedy by discarding all those extracted substrings that contain multiple non-overlapping multi-character words.

3.3.2 Lexicon-based Voting

The errors described in section 3.2 are another influence of the performance of maximized substrings in unknown word extraction. We need to filter out these errors before evaluation, unlike the case of word segmentation where we can encode lexicon-based features in the context of test sentences. If there is any word found in the lexicon that forms the positional relation L3, L4, L5, or L6 (Table 4) with an occurrence

of a maximized substring, a vote is cast to discard the maximized substring. We remove the maximized substring from the extracted list if at least 50% of the occurrences vote for discarding.

3.4 Evaluation

3.4.1 Experiments on Chinese Word Segmentation

Settings To evaluate our approach, we have conducted Chinese word segmentation experiments on several datasets. The first experiment is conducted on Chinese Treebank 5 (CTB5) and Chinese Treebank 7 (CTB7), which are two widely used datasets for word segmentation evaluation. The statistics of CTB5 and CTB7 are shown in Table 15: we have adopted the same setting of data division as (Wang et al. 2011): the training set, dev set and test set; the OOV rate is defined as the percentage of tokens in the datasets that are not found in the corresponding training set.

The second experiment is conducted on the second international Chinese word segmentation bakeoff (SIGHAN Bakeoff-2005) (Emerson 2005), which has four independent subsets: the Academia Sinica Corpus (AS), the Microsoft Research Corpus (MSR), the Hong Kong City University Corpus (CityU) and the Peking University Corpus (PKU). Since POS tags are not available in this dataset, we have omitted all templates that include them.

We have used two different types of unlabeled data. One is the test set itself, which means there is no external resource included in the system. Another is the Chinese Gigaword Second Edition (LDC2007T03), a large-scale dataset, This dataset is a collection of news articles from 1991 to 2004 published by Central News Agency (Taiwan), Xinhua News Agency and Lianhe Zaobao Newspaper. It includes both simplified Chinese characters and traditional Chinese characters, with a total amount of over 1.2 billion. In order to make fair comparison with the previous work, we have adopted the same setting as (Wang et al. 2011) in the experiments conducted on CTB5 and CTB7: we only used the XIN_CMN portion of the Chinese Gigaword,

and we excluded the news articles published between 1994 and 1998 which overlap with CTB data. The resulted unlabeled data has approximately 204 million words.

Maximized Substring	Translation
内蒙古自治区政协	Political Consultative Conference of Inner Mongolia Autonomous Region
温布尔登俱乐部	Wimbledon Football Club
火星环球勘探者	Mars Global Surveyor
丁村文化遗址	Ruins of Dingcun culture
克拉玛依油田	Karamay oil field
黑海舰队问题	Black Sea Fleet porblem
骨质疏松症	Osteoporosis
博斯曼法案	Bosman ruling
加快教育改革	Accelerate education reform
贝纳通车队	Benetton racing team
小阪善太郎	Kosaka Zentaro
黎以和谈	Israeli-Lebanese peace talks
普济禅院	Puji Temple
军事五项	Military pentathlon
利率管理体制	Interest rate regulation system
诗琳通公主	Princess Sirindhorn
珞巴族	Lhoba people
黄玉斌	Huang Yu-bin
颗粒体病毒	Granulosis virus

Table 14. Examples of extracted maximized substrings from Chinese Gigaword.

Table 14 shows some of the extracted maximized substrings from the Chinese Gigaword. We have chosen these instances to demonstrate the approximate distribution of different types of maximized substrings in the entire extracted list. As we can see from the table, many of these maximized substrings are named entities, including names of persons such as “黄玉斌”, “诗琳通公主” and “小阪善太郎”, organizations such as “温布尔登俱乐部”, “内蒙古自治区政协” and “贝纳通车队”, and locations such as “丁村文化遗址”, “克拉玛依油田”, and “普济禅院”. Some of them are technical terms,

such as “骨质疏松症” and “颗粒体病毒”. These strings are commonly accepted words in Chinese. Other instances in this table are compound nouns such as “黑海舰队问题”, and verb phrases such as “加快教育改革”. Although these strings are not single words, they still correctly identify word boundaries and are thus helpful in the task of word segmentation.

We have trained all models using the averaged perceptron algorithm (Collins 2002), which we selected because of its efficiency and stability. To learn the characteristics of unknown words, we built the system’s lexicon using only the words in the training data with a frequency higher than a threshold h. This threshold was tuned using the development data. In order to use the maximized substring features, we have used training data as unlabeled data for supervised models, and used both the training data and Chinese Gigaword for transductive learning models.

We have applied the same parameters for all models, which are tuned on the CTB7 dev set: $h = 2$, $\lambda = 50$, and $\theta = 3$.

We have used precision, recall and the F-score to measure the performance of segmentation systems.

We have also evaluated statistical significance of the results we obtained. We use McNemar's test with Yates correction with correction factor 0.5 to compare the correctness (correct vs. incorrect) between the output of proposed systems against the output of baselines.

	Number of sentences			OOV rate per token	
Dataset	Training	dev	test	dev	test
CTB5	18,089	350	348	0.0811	0.0347
CTB7	31,131	10,136	10,180	0.0549	0.0521

Table 15. Statistics of CTB5 and CTB7.

	CTB5				CTB7			
System	P	R	F	OOV Recall	P	R	F	OOV Recall
Baseline	97.48	98.44	97.96	75.28	95.19	95.86	95.52	74.90
MaxSub-Test	97.52	98.55	98.03	76.47	95.30	95.97	95.63	76.40
MaxSub-U	97.73	98.79	98.26	81.86	95.53	96.15	95.88	80.78

Table 16. Evaluation results on CTB5 and CTB7.

System	CTB5	CTB7
Baseline	97.96	95.52
MaxSub-Test	98.03	95.63*
MaxSub-U+	98.26*	95.88*
Kruengkrai 09a	97.87	95.40
Kruengkrai 09b	97.98	95.46
Wang 11+	98.11	95.65
Zeng 13+	98.27	96.72

Table 17. F-measure on CTB5 and CTB7 test sets compared with previous work. "+": transductive learning systems. "*": $p < 0.05$ in McNemar's test vs. Baseline.

Experimental Results on In-domain Data We have compared the performance between the baseline system and our approach. The results are shown in Table 16. Each row in this table shows the performance of the corresponding system. "Baseline" refers to our baseline hybrid word segmentation and POS-tagging system.

"MaxSub-Test" refers to the method that just uses the test set as unlabeled data. "MaxSub-U" refers to the method that uses both the test set and the large-scale unlabeled data (Chinese Gigaword). We have focused on the segmentation performance of our systems.

The results show that, using the test data as an additional source of information, "MaxSub-Test" outperforms the baseline method by 0.07 and 0.11 points in F-score on CTB5 and CTB7 test set, respectively. This result indicates that our method of using maximized substrings can enhance the segmentation performance even with a purely supervised approach. "MaxSub-U" further increased the improvements to 0.30 and 0.36 points in F-score, which demonstrates the effectiveness of using large-scale unlabeled data. Furthermore, we have observed that "MaxSub-Test" increased the recall of OOV words by 1.19 and 1.50 on CTB5 and CTB7 test set, respectively, and "MaxSub-U" further increased these numbers to 6.58 and 5.88. This result shows that systems enhanced with maximized substrings are much more effective in handling OOV words compared to the baseline. Since the OOV rate in the test set of CTB5 and CTB7 are both less than 6%, it is likely that the improvement can be higher in a dataset with larger percentage of OOV words; this assumption is verified in our experiment in section 3.4.1.4.

We have compared our approach with previous work in Table 17. Two methods from (Kruengkrai et al. 2009a; 2009b) are referred to as "Kruengkrai 09a" and "Kruengkrai 09b", and are taken directly from the report of (Wang et al. 2011); "Wang 11" refers to the semi-supervised system in (Wang et al. 2011); "Zeng 13" refers to the semi-supervised system in (Zeng et al. 2013).

We have observed that our system "MaxSub-U" achieves the better accuracy compared to "Kruengkrai 09a", "Kruengkrai 09b" and "Wang 11". Also, although the performance of our baseline is lower than the systems "Kruengkrai 09a" and "Kruengkrai 09b" because of differences in implementation, the system "MaxSub-Test" (which has used no external resource) has achieved a comparable result. Our proposed method however does not show better accuracy compared to "Zeng 13".

"Zeng 13" processes raw sentences with the mixture of a character-based model and a word-based model; the two models are trained on labeled data, and then further updated using unlabeled examples, where the segmentation agreements are used as constraints to bias the models. On the other hand, our method and the one of "Wang 11" both collect substrings from unlabeled data which have high statistical likelihood to be words, and the information of these substrings can be used as feature and incorporated into character-based, word-based, or hybrid word segmentation models. We think that the method in (Zeng et al. 2013) and ours are non-overlapping, and further improvements can be obtained by using both methods simultaneously. Another concern that we have about the comparison with "Zeng 13" is that, the method performs word segmentation on unlabeled data using two models and iteratively updates both models by optimizing the objective function, which is a more complicated task compared to frequent substring extraction; although the efficiency of their method hasn't been discussed in their original paper, it is likely that performing semi-supervised learning with such a setting can be computationally prohibitive when the size of unlabeled data becomes very large (e.g. billions of web pages). Thus we found that the results obtained by our system and that of "Wang 11" are more directly comparable.

The results for the SIGHAN Bakeoff-2005 dataset are shown in Table 18. The first three rows ("Tseng 05", "Asahara 05" and "Chen 05") show the results of systems that have reached the highest score on at least one corpus (Tseng et al. 2005; Asahara et al. 2005; Chen et al. 2005). "Best closed" summarizes the best official results on all four corpora. "Zhao 07" and "Zhang 06" represent the supervised segmentation systems in (Zhao and Kit 2007; Zhang et al. 2006). "Baseline", "MaxsubTest" and "MaxSub-U" refer to the same systems as in Table 16. For the unlabeled data, we have used the test sets of corresponding corpora for "MaxSub-Test", and the Chinese Gigaword for "MaxSub-U". Other parameters were left unchanged. The results do not indicate that our approach performs better than other systems. However, the joint model we used relies heavily on part-of-speech annotation in the

training data; since part-of-speech information is unavailable in the SIGHAN Bakeoff-2005 dataset, the advantages of our joint model is largely compromised. We therefore mainly focus on comparing the performance of the systems before and after adding maximized substring features. When compared with the baseline, our approach has yielded consistent improvements across the four corpora, and on the PKU corpus we have performed better than previous work.

System	AS	CityU	MSR	PKU
Baseline	95.15	94.52	96.31	95.15
MaxSub-Test	95.22	94.59	96.43*	95.30
MaxSub-U+	95.36*	94.82*	96.64*	95.57*
Tseng 05	94.7	94.3	96.4	95.0
Asahara 05	95.2	94.1	95.8	94.1
Chen 05	94.5	94.0	96.0	95.0
Zhang 06	95.1	95.1	97.2	95.1
Zhao 07	95.5	95.6	97.5	95.4
Best closed	95.2	94.3	96.4	95.0

Table 18. F-measure on SIGHAN Bakeoff-2005 test set compared with previous work. "+": transductive learning systems. "*": $p < 0.05$ in McNemar's test vs. Baseline.

System	CTB5	CTB7
Baseline	97.96	95.52
+Basic&Freq	98.04	95.61
+All	98.10	95.76
+All+Naive	98.15	95.79
+All+STS	98.26	95.88

Table 19. Influence of activated feature types and short-term store on CTB7 test data.

System	Scientific articles corpus
Baseline	92.01
MaxSub-Test	92.52*

Table 20. F-measure on out-of-domain data. "*": $p < 0.01$ in McNemar's test vs. Baseline.

Impacts of Transductive Learning Features and Short-term Store In Table 19, we show the effects of the different maximized substring feature types. We activated different combinations of feature types in turn and trained separate models. We also investigated the impact of the short-term store by training models without this feature. The rows of this table represent models and corresponding F-score, trained and tested on CTB5 and CTB7 with different configurations. The row "Baseline" is baseline system as in Table 16. "+Basic&Freq" represents the system "MaxSub-U" with only basic and frequency features activated, and STS turned off. The row "+All" represents a system activating all maximized substring features but still without STS. The last row "+All+STS" is identical to the system "Maxsub-U". It is clear that lexicon-based features are effective in discriminating unreliable maximized substring from reliable ones, and the short-term store improves the segmentation performance by filtering out noises during the extraction of maximized substrings. The combination of these two techniques yields an improvement of 0.22 point in F-score, and thus are essential when using maximized substrings.

Moreover, to further investigate the effect of short-term store, we have compared "+All+STS" with another system, "+All+Naive", where we replaced short-term store with a naive strategy to limit the context of maximized substring extraction: we split the unlabeled data into small chunks of 50 sentences and extracted maximized substrings from each chunk; then we combined all extracted maximized substrings into a single list and merged their occurrence lists. The result shows that short-term store outperforms this simple strategy on both CTB5 and CTB7 test sets. By com-

paring the maximized substring lists extracted with these two strategies, we found that the simple strategy left out many good word candidates which are infrequent. We think that this is a result of the fixed length chunking of the unlabeled data, as the occurrences of an infrequent maximized substring can be distributed across two or more adjacent chunks and therefore be dropped by the extraction algorithm. On the other hand, short-term store can adjust the context length of maximized substring extraction based on extracted occurrences, which avoided this problem.

Experimental Results on Out-of-domain Data To demonstrate the effectiveness of our method on out-of-domain text, we have conducted an experiment on a test set that was drawn from a corpus of scientific articles. This test set contains 510 sentences that have been manually segmented by a native Chinese speaker. We used the test set as the unlabeled data.

As the results show (Table 20), the system "MaxSub-Test" exceeded the baseline method by 0.51 in F-score, and the improvement is statistically significant ($p < 0.01$ in McNemar's test vs. Baseline). Considering that the amount of unlabeled data is relatively small, it is likely that acquiring large-scale unlabeled data in the same domain will further benefit the accuracy.

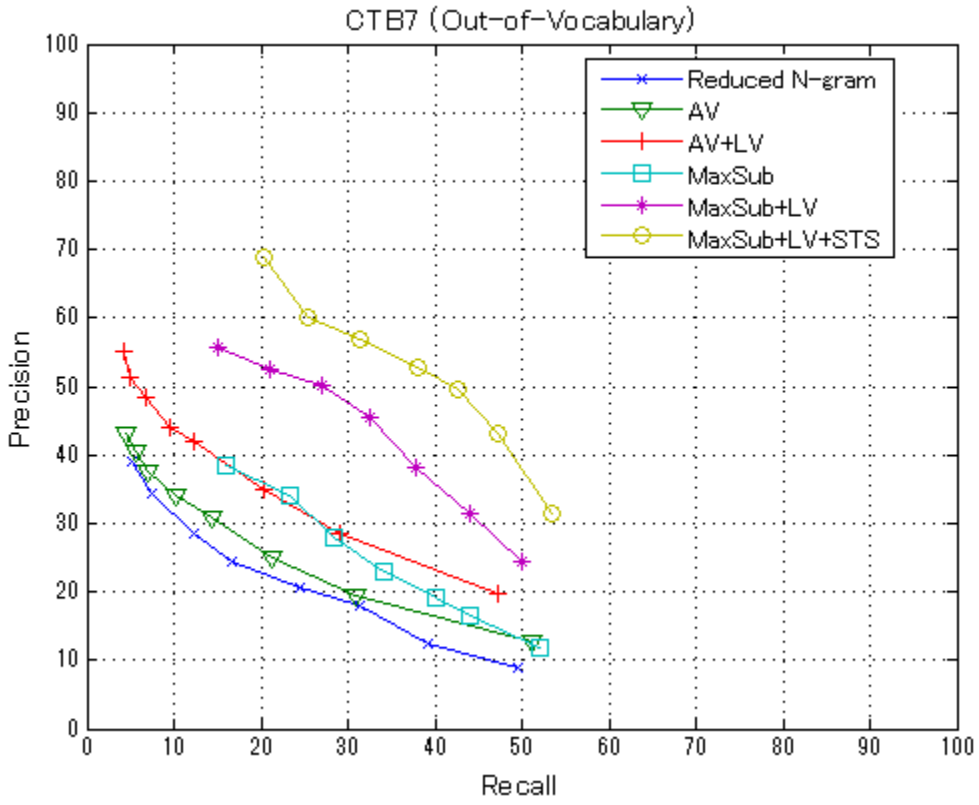


Figure 4. Precision-Recall curves of maximized substring extraction and the accessor variety method on CTB7.

3.4.2 Experimental Results on Unknown Word Extraction

Settings We have also conducted unknown word extraction experiments. It is difficult to directly evaluate the precision and recall of a list of extracted unknown words, since there is no complete list of unknown words to be compared with. Previous studies have adopted evaluation methods based on hand annotation (Feng et al. 2004; Zhang et al. 2011). We instead used the word list of CTB7 as gold-standard data for evaluation. We used the entire CTB7 dataset as an input text for maximized substring extraction, which has 51,447 sentences. We used the same lexicon that has been used in previous studies (Feng et al. 2004; Zhang et al. 2011), which

has 119,803 Chinese words of two to seven characters[22]. With the words in the lexicon being known, there are 11,722 unknown words remaining in the word list of CTB7. The result is compared with the reduced N-gram method (Lin and Yu 2001) as well as the accessor variety method (Feng et al. 2004), which is one of the most widely applied Chinese word recognition method (Zhao and Kit 2007; Zhao and Kit 2008; Sun and Xu 2011).

Performance Comparison In Figure 4 we show the performance of three maximized substring-based systems: "MaxSub" represents the maximized substring extraction method described in section 2.2, "MaxSub+LV" represents the previous system plus the post-processing technique of lexicon-based voting, and "MaxSub+LV+STS" represents the second system, which uses the short-term store. The parameters of the short-term store are $\theta = 2$ and $\lambda = 20$, which are the optimum combination for this dataset. We have applied the redundancy reduction method described in section 3.1 to all three systems. The number of substrings have been reduced by 22.0% for "MaxSub" after applying the redundancy reduction method, and by 39.1% and 28.3% respectively for "MaxSub+LV" and "MaxSub+LV+STS" after applying both the redundancy reduction and lexicon-based voting method.

[22] http://www.mandarintools.com/segmenter.html

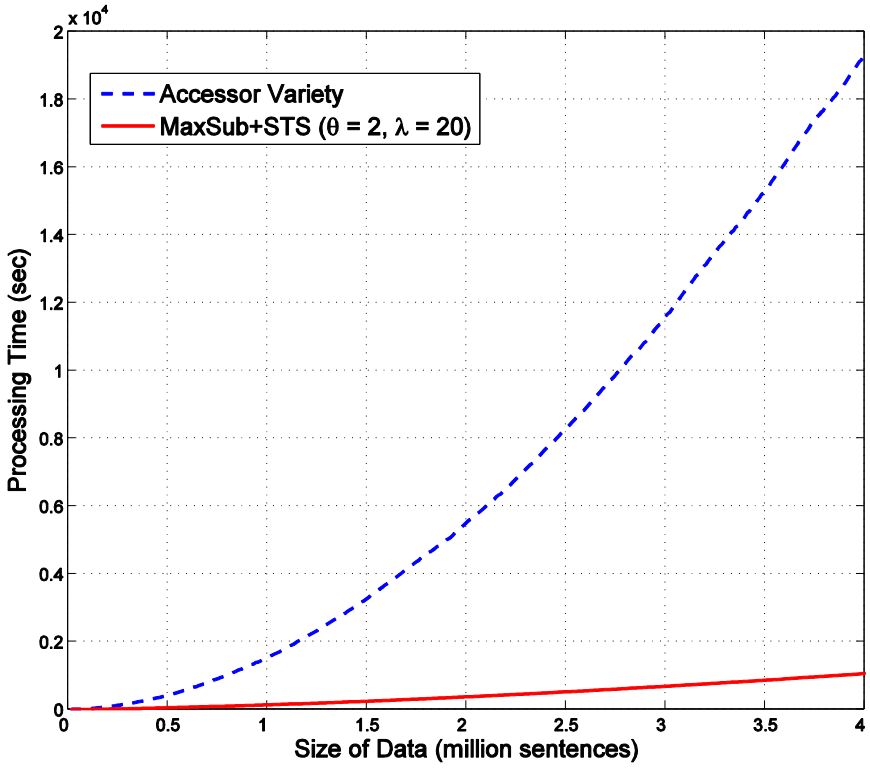


Figure 5. Processing time comparison between maximized substring extraction and the accessor variety method on a large-scale text.

We have also implemented the methods of reduced N-gram (Lin and Yu 2001) and accessor variety (Feng et al. 2004) for comparison. The reduced N-gram method and the accessor variety method are represented as "Reduced N-gram" and "AV" respectively; "AV+LV" represents the method of "AV" plus the lexicon-based voting technique, where 14.9% of the substrings are removed compared to "AV". We have calculated the reduced N-gram and accessor variety for substrings of two to seven characters; for the accessor variety methods, we have also applied the adhesive judge rules (Feng et al. 2004) to reduce errors.

The plot points in Figure 4 are obtained by varying the threshold of minimum frequency of substrings that we treat as words, i.e. we only regard an extracted substring as a word candidate if and only if its frequency is no less than the threshold.

This threshold varies from 2 to 8 for "MaxSub", "MaxSub+LV", and "MaxSub+LV+STS", and from 2 to 9 for "Reduced N-gram", "AV", and "AV+LV". The result shows that our method substantially outperforms the methods of reduced N-gram and accessor variety, and both the short-term store and the lexicon-based voting techniques can largely contribute to the overall performance of unknown word extraction.

It should be noticed that the precision and recall of these systems are generally lower than those reported by (Feng et al. 2004), which is due to two reasons. We have considered only out-of-vocabulary words in the evaluation, while Feng et al. consider both known words and unknown words. Also, a major drawback of our automatic evaluation method is that, although commonly acceptable as single words by native Chinese speakers, named entities and compound nouns are often regarded as multiple words in the gold-standard data. So the precision and recall values tend to be much lower than in the case of human evaluation.

Efficiency To demonstrate the efficiency of our approach for processing large-scale data, in Figure 5 we have compared the processing time of our system ("MaxSub+STS", no post-processing) against the method of accessor variety. We have used the first four million sentences of as the Xinhua Newswire section in Chinese Gigaword Second Edition. The results show that up to 12 times the processing time can be required for this dataset. In addition, our method shows a quasi-linear time complexity while the time taken for the accessor variety method is $O(n^2)$.

3.4.3 Error Analysis

In the extracted list of maximized substrings, we regard instances representing single words, compound words and phrases as meaningful strings, as they correctly identify word boundaries. Instances which violate word boundaries are regarded as errors. During the analysis of the output of maximized substring extraction, we found that the majority of errors involves person's name. A typical instance is the string "舍维奇" (šević), which is a commonly used suffix in the transliteration of person's names like "米洛舍维奇" (Milošević) and "伊比舍维奇" (Ibišević). Since instances of transliteration are relatively rare in a Chinese lexicon, it leaves no trace for lexicon-based voting or features to reject this kind of instances.

In fact, this kind of errors happens not only in the transliteration of foreign person's names. For instance, a maximized substring "记者马" (the journalist Ma) is extracted in our experiment from the context "记者马维坤" (the journalist Ma Weikun) and "记者马建国" (the journalist Ma Jianguo). Unlike in western languages, it is not common to split the surname and the given name in person's names in Chinese, in which case the character "马" (Ma) violates word boundaries in person's name "马维坤" (Ma Weikun) and "马建国" (Ma Jianguo). Like the case in transliteration, lexicon-based techniques are not helpful in rejecting this instance. More unfortunately, the given name "建国" in "记者马建国" (the journalist Ma Jianguo) also means "founding of a nation" and is therefore regarded as a lexicon entry, so it actually "supported" the right boundary identified by the instance "记者马" by applying lexicon-based techniques. This problem is not rare in our experiment, since given names in Chinese are often meaningful strings registered in a lexicon. Many instances of this error pattern is however a result of word segmentation standard adopted in CTB. For example, the same maximized substring "记者马" (the journalist Ma) would not be considered as violating word boundaries if the gold-standard word

segmentation follows the standard in the PKU corpus, as it treats Chinese family names and given names as separate words.

Another type of errors we observed in the CTB5 and CTB7 word segmentation experiments is that, although a maximized substring correctly identified the boundaries of a phrase, the segmenter still made incorrect segmentation decisions regarding word boundaries inside that phrase. This could happen when an extracted maximized substring corresponds to a phrase in the gold-standard dataset, while the system has no knowledge regarding the internal word boundaries of that phrase from either the lexicon or extracted maximized substrings. For example, "麦格赛赛奖" (Magsaysay Prize) is a maximized substring extracted from CTB5 test set, while the OOV person's name "麦格赛赛" (Magsaysay) is not included in the list of maximized substrings. The best model in our experiments segmented this phrase as "麦格 赛赛奖" which is incorrect. We think this problem can be largely relieved by adding word formation knowledge in Chinese, e.g. a person's name and the word for "prize" can form a phrase, into the model as features; it is however outside the scope of this study to discuss the methods of obtaining such knowledge.

3.5 Comparison to Related Work

The idea of using frequent strings collected from text in word segmentation and recognition have been long employed by previous works.

Nagao and Mori (1994) proposed a method for collecting character N-grams from large scale texts. They investigated the characteristics of the resulted frequent strings, and demonstrated that it is possible to automatically recognize compounds, set phrases and idiomatic phrases in Japanese by merging adjacent frequent strings. Lin and Yu (2001) proposed a method which extracts unknown words from a corpus based on string frequency. They showed that by discounting a string's frequency from its superstrings ("reduced N-gram"), the resulted set of frequent strings can be used as reliable indicators of word boundaries. This type of frequent strings has also

been exploited by Zhao and Kit (2007) for enhancing Chinese word segmentation, and Sung et al. (2008) for calculating more reliable term frequencies.

Another line of research studied the properties of frequent substrings from the information theory aspect. Feng et al. (2004) proposed the accessor variety-based method, a widely applied method in Chinese word segmentation and word recognition (Sun and Xu 2011; Yang et al. 2011; Zhao and Kit 2007; Zhao and Kit 2008), which is another view of measuring the likelihood of a string being a Chinese word. The accessor variety of a string s is defined as

$$AV(s) = \min\{L_{av}(s), R_{av}(s)\}$$

where $L_{av}(s)$ and $R_{av}(s)$ are the number of distinct surrounding characters of s from its left and right-hand side, respectively. Accessor variety is closely related to the concept of branching entropy (Jin and Tanaka-Ishii, 2006) and can be seen as the continuous version of it.

Though accessor variety is a good indicator of word candidates in unknown word extraction, the candidates extracted with a high threshold of accessor variety can still sometimes be meaningless strings. For example, consider the string "常谈" which is not generally considered as a word or phrase in contemporary the Chinese language. It is a suffix of the Chinese idiom "老生常谈" (cliche), so the value $R_{av}(\text{常谈})$ should be high in a Chinese corpus which contains this idiom. On the other hand, $L_{av}(\text{常谈})$ should also be high if the same corpus contains the phrase "常谈起" (often talk about). By the definition, we can draw the conclusion that $AV(\text{常谈})$ is high in this corpus and therefore should be a word, which is incorrect. This problem can happen because in the definition of accessor variety, the numbers of distinct surrounding characters of a string are counted from its left and right-hand side independently. On the other hand, this problem is avoided in maximized substring extraction, since all occurrences of a maximized substring must have distinct sur-

rounding characters from both left and right-hand sides at the same time according to the definition.

Maximized substring extraction does not require the extra step of statistical substring reduction (SSR), which makes the implementation simple. Unlike the above methods where it is necessary to collect statistics from all possible strings in a corpus that takes quadratic time (or quasilinear time if implemented with suffix arrays), maximized substring extraction only scans the corpus once which is a linear time algorithm in average case. The efficiency advantage of maximized substring extraction is demonstrated in section 5.3.3.

Another advantage of maximized substring extraction is its ability to utilize the short-term store technique, which enhances the performance of unknown word extraction and Chinese word segmentation. The effect of short-term store is demonstrated in section 5.2.3. and section 5.3.2.

3.6 Summary

In this chapter we have proposed a simple yet effective approach for extracting maximized substrings from unlabeled data, which are a particular type of substrings that provide good estimations of unknown word boundaries. We have evaluated our approach in two tasks. In the task of Chinese word segmentation, the extracted maximized substrings are incorporated with a supervised segmentation system through discriminative learning. We have demonstrated the effectiveness of our approach through experiments in both in-domain and out-of-domain data and have achieved significant improvements over the baseline systems across all datasets. In the task of Chinese unknown word extraction, our approach has also been demonstrated to substantially outperform the previous work of accessor variety in both the accuracy and efficiency.

Chapter 4

Chinese Morphological Analysis with Character-level POS Tagging

In recent years, the focus of research on Chinese word segmentation, part-of-speech (POS) tagging and parsing has been shifting from words toward characters. Character-based methods have shown superior performance in these tasks compared to traditional word-based methods (Ng and Low, 2004; Nakagawa, 2004; Zhao et al., 2006; Kruengkrai et al., 2009; Xue, 2003; Sun, 2010). Studies investigating the morphological-level and character-level internal structures of words, which treat character as the true atom of morphological and syntactic processing, have demonstrated encouraging results (Li, 2011; Li and Zhou, 2012; Zhang et al., 2013). This line of research has provided great insight in revealing the roles of characters in word formation and syntax of the Chinese language.

However, existing methods have not yet fully utilized the potentials of Chinese characters. While Li (2011) pointed out that some characters can productively form new words by attaching to existing words, these characters consist only a portion of all Chinese characters and appear in 35% of the words in Chinese Treebank 5.0 (CTB5) (Xue et al., 2005). Zhang (2013) took one step further by investigating the character-level structures of words; however, the machine learning of inferring these internal structures relies on the character forms, which still suffers from data sparseness.

In our view, since each Chinese character is in fact created as a word in origin with complete and independent meaning, it should be treated as the actual minimal morphological unit in the Chinese language, and therefore should carry specific

part-of-speech. For example, the character “打” (beat) is a verb and the character “破” (broken) is an adjective. A word on the other hand, is either single-character, or a compound formed by single-character words. For example, the verb “打破” (break) can be seen as a compound formed by the two single-character words with the construction “verb + adjective”.

Character-level Part-of-Speech	Examples of Verb
verb + noun	投资 (invest：throw + wealth)
noun + verb	心疼 (feel sorry：heart + hurt)
verb + adjective	认清 (realize：recognize + clear)
adjective + verb	痛恨 (hate：pain + hate)
verb + verb	审查 (inspect：examine + review)

Table 21. Character-level POS sequence as a more specified version of word-level POS: an example of verb.

Under this treatment, we observe that words with the same construction in terms of character-level POS tend to also have similar syntactic roles. For example, the words having the construction “verb + adjective” are typically verbs, and those having the construction “adjective + noun” are typically nouns, as shown in the following examples:

(a) verb：verb + adjective
“打破”(break)：“打”(beat) + “破”(broken)
“更新”(update)：“更”(replace) + “新”(new)
“漂白”(bleach)：“漂”(wash) + “白”(white)

(b) noun：adjective + noun
“主题”(theme)：“主”(main) + “题”(topic)
“新人”(newcomer)：“新”(new) + “人”(person)
“快车”(express)：“快”(fast) + “车”(car)

This suggests that character-level POS can be used as cues in predicting the part-of-speech of unknown words.

Another advantage of character-level POS is that, the sequence of character-level POS in a word can be seen as a more fine-grained version of word-level POS. An example is shown in Table 21. The five words in this table are very likely to be tagged with the same word-level POS as verb in any available annotated corpora, while it can be commonly agreed among native speakers of Chinese that the syntactic behaviors of these words are different from each other, due to their distinctions in word constructions. For example, verbs having the construction "verb + noun" (e.g. 投资) or "verb + verb" (e.g. 审查) can also be nouns in some context, while others cannot; and verbs having the constructions "verb + adjective" (e.g. 认清) require exact one object argument, while others generally do not. Therefore, compared to word-level POS, the character-level POS can produce information for more expressive features during the learning process of a morphological analyzer.

In this chapter, we investigate the usefulness of character-level POS in the task of Chinese morphological analysis. We propose the first tagset designed for the task of character-level POS tagging, based on which we manually annotate the entire CTB5. We propose a method that performs character-level POS tagging jointly with word segmentation and word-level POS tagging. Through experiments, we demonstrate that by introducing character-level POS information, the performance of a baseline morphological analyzer can be significantly improved.

Tag	Part-of-Speech	Example
n	noun	法案/NN (bill)
v	verb	发布/VV (publish)
j	adj./adv.	广阔/VA (vast)
t	numerical	三点一四/CD (3.14)
m	quantifier	一/CD 件/M (a piece of)
d	date	九五年/NT (1995)
k	proper noun	中美/NR (sino-US)
b	prefix	副市长/NN (vice mayor)
e	suffix	建筑业/NN (construction inductry)
r	transliteration	阿尔帕德/NR (Árpád)
u	punctuation	查尔斯·狄更斯/NR (Charles Dickens)
f	foreign chars	X 射线/NN (X-ray)
o	onomatopoeia	隆隆/AD (rumble)
s	surname	王新民/NR (Wang Xinmin)
p	pronoun	他们/PN (they)
c	other functional	用于/VV (be used for)

Table 22. Tagset for character-level part-of-speech tagging. The underlined characters in the examples correspond to the tags on the left-most column. The CTB-style word-level POS are also shown for the examples.

4.1 Character-level POS Tagset

We propose a tagset for the task of character-level POS tagging. This tagset contains 16 tags, as illustrated in Table 22. The tagset is designed by treating each Chinese character as a single-character word, and each (multi-character) word as a phrase of single-character words. Some of these tags are directly derived from the commonly accepted word-level part-of-speech, such as noun, verb, adjective and adverb. It should be noted that, for single-character words, the difference between adjective and adverb can almost be ignored, because for any of such words that can be used as an adjective, it usually can also be used as an adverb. Therefore, we have merged these two tags into one.

On the other hand, some other tags are designed specifically for characters, such as transliteration, surname, prefix and suffix. Unlike some Asian languages such as Japanese, there is no explicit character set in Chinese that are used exclusively for expressing names of foreign persons, places or organizations. However, some characters are used much more frequently than others in these situations. For example, in the person's name "阿尔帕德" (Árpád), all the four characters can be frequently observed in words of transliterations. Similarly, surnames in Chinese are also drawn from a set of limited number of characters. We therefore assign specific tags for this kind of character sets. The tags for prefixes and suffixes are motivated by the previous studies (Li, 2011; Li and Zhou, 2012).

We have annotated character-level POS for all words in CTB5[23]. Fortunately, character-level POS in most words are independent of context, which means it is sufficient to annotate word forms unless there is an ambiguity. The annotation was conducted by two persons, where each one of them was responsible for about 70% of the documents in the corpus. The redundancy was set for the purposes of style unification and quality control, on which we find that the inter- annotator agreement is 96.2%. Although the annotation also includes the test set, we blind this portion in all the experiments.

[23] http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?CharPosCN

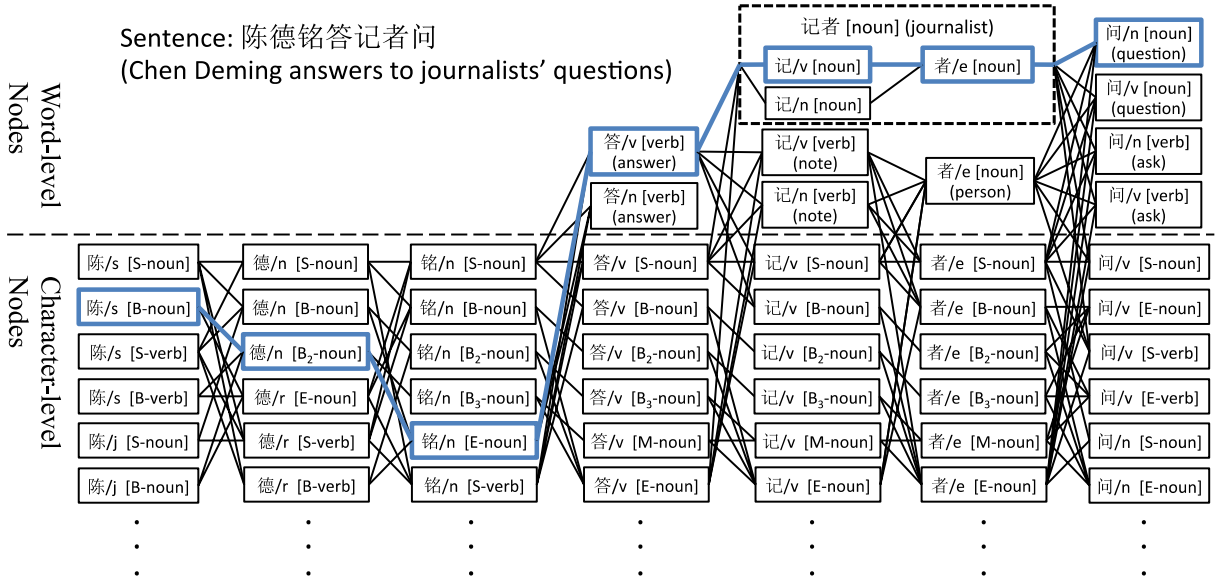


Figure 6. A Word-character hybrid lattice of a Chinese sentence. Correct path is represented by blue bold lines.

4.2 Chinese Morphological Analysis with Character-level POS

4.2.1 System Description

Previous studies have shown that jointly processing word segmentation and POS tagging is preferable to pipeline processing, which can propagate errors (Nakagawa and Uchimoto, 2007; Kruengkrai et al., 2009). Based on these studies, we propose a word-character hybrid model which can also utilize the character-level POS information. This hybrid model constructs a lattice that consists of word-level and character-level nodes from a given input sentence. Word-level nodes correspond to words found in the system's lexicon, which has been compiled from training data. Character-level nodes have special tags called position-of-character (POC) that indicate the word-internal position (Asahara, 2003; Nakagawa, 2004). We have adopted the 6-tag tagset, which (Zhao et al., 2006) reported to be optimal. Figure 6 shows an example

of a lattice for the Chinese sentence: “陈德铭答记者问” (Chen Deming answers to journalists’ questions). The correct path is marked with blue bold lines. The upper part of the lattice (word-level nodes) represents known words, where each node carries information such as character form, character-level POS, and word-level POS. A word that contains multiple characters is represented by a sub-lattice (the dashed rectangle in the figure), where a path stands for a possible sequence of character-level POS for this word. For example, the word “记者” (journalist) has two possible paths of character-level POS: “verb + suffix” and “noun + suffix”. Nodes that are inside a sub-lattice cannot be linked to nodes that are outside, except from the boundaries. The lower part of the lattice (character-level nodes) represents unknown words, where each node carries a position-of-character tag, in addition to other types of information that can also be found on a word-level node. A sequence of character-level nodes are considered as an unknown word if and only if the sequence of POC tags forms one of the cases listed in Table 10 in Chapter 3. This table also illustrates the permitted transitions between adjacent character-level nodes. We use the standard dynamic programming technique to search for the best path in the lattice. We use the averaged perceptron (Collins, 2002), an efficient online learning algorithm, to train the model.

Category	Template	Condition
Baseline-unigram	$\langle w_0 \rangle \langle p_0 \rangle \langle w_0, p_0 \rangle \langle l_0, p_0 \rangle \langle \text{begin}(w_0), p_0 \rangle \langle \text{end}(w_0), p_0 \rangle$	W_0
	$\langle \text{begin}(w_0), \text{end}(w_0), p_0 \rangle$	
	$\langle c_{-2}, p_0 \rangle \langle c_{-1}, p_0 \rangle \langle c_0, p_0 \rangle \langle c_1, p_0 \rangle \langle c_2, p_0 \rangle$	C_0
	$\langle c_{-2}, c_{-1}, p_0 \rangle \langle c_{-1}, c_0, p_0 \rangle \langle c_0, c_1, p_0 \rangle \langle c_1, c_2, p_0 \rangle \langle c_{-1}, c_1, p_0 \rangle$	
Baseline-bigram	$\langle w_{-1}, w_0 \rangle \langle p_{-1}, p_0 \rangle \langle w_{-1}, p_0 \rangle \langle p_{-1}, w_0 \rangle \langle w_{-1}, p_{-1}, w_0 \rangle \langle w_{-1}, w_0,$	$W_{-1} \times W_0$
	$\langle w_{-1}, p_{-1}, p_0 \rangle \langle p_{-1}, w_0, p_0 \rangle \langle w_{-1}, p_{-1}, w_0, p_0 \rangle \langle l_{-1}, p_{-1}, l_0 \rangle \langle l_{-1},$	
	$\langle l_{-1}, p_{-1}, p_0 \rangle \langle p_{-1}, l_0, p_0 \rangle \langle l_{-1}, p_{-1}, l_0, p_0 \rangle \langle \text{end}(w_{-1}), p_0 \rangle$	
	$\langle p_{-1}, \text{begin}(w_0) \rangle \langle \text{end}(w_{-1}), p_{-1}, p_0 \rangle \langle p_{-1}, \text{begin}(w_0), p_0 \rangle$	
	$\langle c_{-1}, c_0 \rangle \langle p_{-1}, p_0 \rangle \langle c_{-1}, p_{-1}, c_0 \rangle \langle c_{-1}, c_0, p_0 \rangle$	$C_{-1} \times C_0$
	$\langle c_{-1}, p_{-1}, p_0 \rangle \langle p_{-1}, c_0, p_0 \rangle \langle c_{-1}, p_{-1}, c_0, p_0 \rangle$	
	$\langle p_{-1}, p_0 \rangle$	Otherwise
Proposed-unigram	$\langle \text{CP}(c_0), p_0 \rangle$	C_0
Proposed-bigram	$\langle \text{CP}_{pair}(w_{-1}), p_0 \rangle \langle \text{CP}_{pair}(w_{-1}), p_{-1}, p_0 \rangle$	$W_{-1} \times N_0$
	$\langle \text{CP}_{all}(w_{-1}), p_0 \rangle \langle \text{CP}_{all}(w_{-1}), p_{-1}, p_0 \rangle$	
	$\langle \text{CP}_{pair}(w_{-1}), p_{-1}, \text{CP}(c_0) \rangle \langle \text{CP}_{pair}(w_{-1}), \text{CP}(c_0), p_0 \rangle$	$W_{-1} \times C_0$
	$\langle \text{CP}_{all}(w_{-1}), p_{-1}, \text{CP}(c_0) \rangle \langle \text{CP}_{all}(w_{-1}), \text{CP}(c_0), p_0 \rangle$	
	$\langle p_{-1}, \text{CP}(c_0) \rangle \langle p_{-1}, \text{CP}(c_0), p_0 \rangle$	$N_{-1} \times C_0$
	$\langle \text{CP}(c_{-1}), p_0 \rangle \langle \text{CP}(c_{-1}), p_{-1}, p_0 \rangle$	$C_{-1} \times N_0$
	$\langle \text{CP}(c_{-1}), p_{-1}, \text{CP}(c_0) \rangle \langle \text{CP}(c_{-1}), \text{CP}(c_0), p_0 \rangle \langle \text{CP}(c_{-1}), p_{-1}, \text{CP}($	$C_{-1} \times C_0$

Table 23. Feature templates. The "Condition" column describes when to apply the templates: W_{-1} and W_0 denote the previous and the current word-level node; C_{-1} and C_0 denote the previous and the current character-level node; N_{-1} and N_0 denote the previous and the current node of any types. Word-level nodes represent known words that can be found in the system's lexicon.

4.2.2 Features

We show the feature templates of our model in Table 23. The features consist of two categories: baseline features, which are modified from the templates proposed in (Kruengkrai et al., 2009); and proposed features, which encode character-level POS information.

Baseline features: for word-level nodes that represent known words, we use the symbols w, p and l to denote the word form, POS tag and length of the word, respectively. The functions begin(w) and end(w) return the first and last character of w. If w has only one character, we omit the templates that contain begin(w) or end(w). We use the subscript indices 0 and -1 to indicate the current node and the previous node during a Viterbi search, respectively. For character-level nodes, c denotes the surface character, and p denotes the combination of POS and POC (position-of-character) tags.

Proposed features: for word-level nodes, the function $\mathrm{CP}_{pair}(w)$ returns the pair of the character-level POS tags of the first and last characters of w, and $\mathrm{CP}_{all}(w)$ returns the sequence of character-level POS tags of w. If either the pair or the sequence of character-level POS is ambiguous, which means there are multiple paths in the sub-lattice of the word-level node, then the values on the current best path (with local context) during the Viterbi search will be returned. If w has only one character, we omit the templates that contain $\mathrm{CP}_{pair}(w)$. For character-level nodes, the function $\mathrm{CP}(c)$ returns its character-level POS. The subscript indices 0 and -1 as well as other symbols stand for the same meaning as they are in the baseline features.

4.3 Evaluation

4.3.1 Settings

To evaluate our proposed method, we have conducted two sets of experiments on CTB5: word segmentation, and joint word segmentation and word-level POS tagging. We have adopted the same data division as in (Jiang et al., 2008a; Jiang et al., 2008b; Kruengkrai et al., 2009; Zhang and Clark, 2010; Sun, 2011): the training set, dev set and test set have 18,089, 350

and 348 sentences, respectively. The models applied on all test sets are those that result in the best performance on the CTB5 dev set.

We have annotated character-level POS information for all 508,768 word tokens in CTB5. As mentioned in section 4.1, we blind the annotation in the test set in all the experiments. To learn the characteristics of unknown words, we built the system's lexicon using only the words in the training data that appear at least 3 times. We applied a similar strategy in building the lexicon for character-level POS, where the threshold we choose is 2. These thresholds were tuned using the development data.

We have used precision, recall and the F-score to measure the performance of the systems.

4.3.2 Experimental Results

We compare the performance between a baseline model and our proposed approach. The results of the word segmentation experiment and the joint experiment of segmentation and POS tagging are shown in Table 24(a) and Table 24(b), respectively. Each row in these tables shows the performance of the corresponding system. "CharPos" stands for our proposed model which has been described in section 4.2. "Baseline" stands for the same model except it only enables features from the baseline templates.

The results show that, while the differences between the baseline model and the proposed model in word segmentation accuracies are small, the proposed model achieves significant improvement in the experiment of joint segmentation and POS tagging[24]. This suggests that our proposed method is particularly effective in predicting the word-level POS, which is consistent with our observations mentioned in the beginning of this chapter.

[24] $p < 0.05$ in McNemar's test.

(a) Word Segmentation Results

System	P	R	F
Baseline	97.48	98.44	97.96
CharPOS	97.55	98.51	98.03

(b) Joint Segmentation and POS Tagging Results

System	P	R	F
Baseline	93.01	93.95	93.48
CharPOS	93.42	94.18	93.80

Table 24. Experimental results on CTB5.

System	Segmentation	Joint
Baseline	97.96	93.48
CharPOS	98.03	93.80
Jiang2008a	97.85	93.41
Jiang2008b	97.74	93.37
Kruengkrai2009	97.87	93.67
Zhang2010	97.78	93.67
Sun2011	98.17	94.02

Table 25. Comparison with previous studies on CTB5.

In Table 25 we compare our approach with morphological analyzers in previous studies. The accuracies of the systems in previous work are directly taken from the original paper. As the results show, despite the fact that the performance of our baseline model is relatively weak in the joint segmentation and POS tagging task, our proposed model achieves the second-best performance in both segmentation and joint tasks.

4.4 Summary

In this chapter we have proposed a method that performs character-level POS tagging jointly with word segmentation and word-level POS tagging. Through experiments, we have demonstrated that by introducing character-level POS information, the performance of a baseline morphological analyzer can be significantly improved.

Chapter 5

Dependency Parsing Reranking with Rich Subtree Features

Graph-based models are prevalent in dependency parsing because of their state-of-the-art accuracy and efficiency. These traits are gained from their ability to combine exact inference and discriminative learning methods. The ability to perform efficient exact inference relies on the so-called factorization technique, which breaks down a parse tree into smaller substructures to perform an efficient dynamic programming search. This treatment, however, restricts the representation of features to a local context, for example, single edges or adjacent edges. Such restriction prohibits the model from exploring large or complex structures for linguistic evidence. This is considered to be the major drawback of graph-based approaches.

Attempts to develop more complex factorization techniques and corresponding decoding methods have been made. Higher-order models that use grand-child, grand-sibling, or tri-sibling factorization were proposed (Koo and Collins, 2010) to explore more expressive features, and these have proven to significantly improve parsing accuracy. However, the power of higher-order models comes with the cost of expensive computation, which sometimes necessitates aggressive pruning during pre-processing.

Another line of research that explores complex feature representations is parse reranking. In its general framework, a K-best list of parse tree candidates is first produced from the base parser. A reranker is then applied to determine the best parse among these candidates. Successful results have been reported for constituent parsing (Collins, 2000; Charniak and Johnson, 2005; Huang, 2008). In terms of dependency parsing, a number of efficient algorithms for producing the K-best list for graph-based parsers have been proposed for projective parsing (Huang and Chiang, 2005) and non-projective parsing (Hall, 2007). Improvements in dependency accuracy have also been achieved (Hall, 2007; Hayashi et al., 2011).

However, the feature sets in these studies explored a relatively small context, either by emulating the feature set in the constituent parse reranking, or by factorizing the search space. A desirable approach for the K-best list reranking is to encode features on subtrees extracted from the candidate parse with arbitrary orders and structures, as long as the extraction process is tractable. The design of a subtree extraction process that provides reliable linguistic evidence is an open question. Another related challenge is to design a proper back-off strategy for the extracted structures, because large subtree instances are always sparse in the training data.

In this work, we explore a feature set that makes full use of dependency grammar, can capture global information with less restriction on the structure and size of the subtrees, and can be encoded efficiently. We exhaustively explore a candidate parse tree for features, from the most simple to the most expressive, while maintaining efficiency in the sense that our method does not impose additional complexities over the K-best parsing. This work is an extension of the dependency parse reranking method proposed by Shen et al. (2012, 2013).

We choose the K-best list reranking framework rather than forest reranking (Huang, 2008), because an explicit representation of parse trees is needed to compute the features for reranking. We implement an edge-factored parser and a second-order sibling-factored parser, which emulate the models in the MSTParser, as our base parsers (McDonald et al., 2005; McDonald and Pereira, 2006).

The main contributions of this work can be summarized as follows.

- We describe a subtree extraction process that collects different types of subtrees from a dependency parse. The targets of our subtree extraction are the most informative and reliable substructures, which can improve the discriminative learning of a dependency parser by representing global information.
- We propose a feature encoding method for large and flexible substructures in a dependency parse. Without using a template, our method takes advantage of the trace of subtree extraction to generate various levels of back-off features.
- We employ a reranking framework to utilize rich subtree features. Empirical results show that our approach substantially improves the baseline and outperforms other state-of-the-art supervised dependency parsing systems.

5.1 Related Work

McDonald et al. proposed an edge-factored parser and a second-order parser that were both trained by discriminative online learning methods (McDonald et al., 2005; McDonald and Pereira, 2006). Huang and Chiang proposed an efficient algorithm for producing the K-best list for graph-based parsers, which add a factor of $K \log K$ to the parsing complexity of the base parser (Huang and Chiang, 2005). Sangati et al. has shown that a discriminative parser is very effective at filtering out bad parses from a factorized search space (Sangati et al., 2009). This agreed with the conclusion of Hall (2007), who found that an edge-factored model can attain good oracle performance when generating a relatively small K-best list. Successful results have been reported for constituent parse reranking (Collins, 2000; Charniak and Johnson, 2005; Huang, 2008). These works proposed feature sets defined on constituent parses that are able to capture rich non-local information. These feature sets, however, cannot be directly applied to parse trees under dependency grammar. Attempts have been made to use similar feature sets in dependency parse reranking (Hall, 2007) by defining a feature set similar to that of Charniak and Johnson (Charniak and Johnson, 2005). Hayashi et al. (2011) presented a forest reranking model that applies third-order factorizations, emulating Model 1 and Model 2 in Koo and Collins (2010), on the search space of the reranker.

5.2 Dependency Parsing

The aim of dependency parsing is to find a tree structure for a sentence in which edges represent the head-modifier relationship between words: each word is linked to a unique "head" such that the link forms a semantic dependency, while the main predicate of the sentence is linked to a dummy "root". An example of dependency tree is illustrated in Figure 7. A dependency tree is called projective if the links can be drawn between the linearly ordered words without any cross links. In this work, we have focused on projective trees.

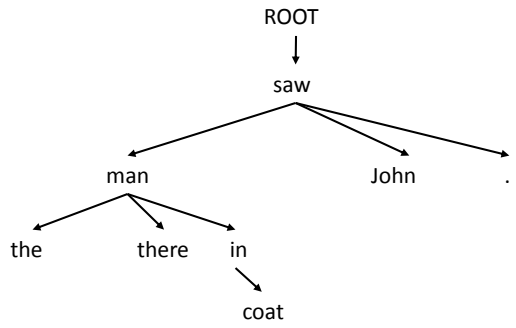


Figure 7. A dependency parse tree of the sentence “the man there in coat saw John”.

We now formally define the task of dependency parsing. Given a sentence x, the best parse tree is obtained by searching for the tree with the highest score:

$$\tilde{y} = argmax_{y \in \mathcal{Y}(x)} Score(y, x) \qquad (1)$$

where $\mathcal{Y}(x)$ is the search space of possible parse trees for x, and y is a parse tree in $\mathcal{Y}(x)$. The problem in solving Equation (1) is that the number of candidates in the search space grows exponentially with the length of the sentence, which can render the search infeasible. A common remedy for this problem is to factorize a parse tree into small subtrees, called factors, which are scored independently. The score of a parse tree under such factorization is the summation of the scores of its factors:

$$Score(y, x) = \sum_{t \in y} Score(t, x) \qquad (2)$$

where t is a factor of y. The search space can therefore be encoded in a compact form, allowing dynamic programming algorithms to perform efficient exact inference. The score function for each factor is assigned as an inner product of a feature vector and a weight vector w:

$$Score(t, x) = w \cdot f(t, x) \qquad (3)$$

The feature vector is defined on the factor t, which means it is only able to capture tree-structure information from a small context. This can be seen as the offset for performing exact inference. The goal when training a parser is to learn the weight vector that assigns scores to effectively discriminate good from bad parses.

We use the edge factorization and sibling factorization models (McDonald et al., 2005; McDonald and Pereira, 2006) to construct our base parsers. We learn the weight vector by applying the averaged perceptron algorithm (Collins, 2002), which is efficient and stable. An illustration of the generic perceptron algorithm is shown in Algorithm 2.

Algorithm 2: Generic Perceptron Learning

```
1  for iteration t = 1..T
2    for training data (x_i, y_i), i = 1..N
3      ỹ = argmax_{y∈Y(x)} w · f(y, x_i)
4      if ỹ ≠ y_i
5        w ← w + f(y_i, x_i) − f(ỹ, x_i)
6    end for
7  end for
```

5.3 Parse Reranking

In this section, we describe our reranking approach and introduce a feature set consisting of three different types.

5.3.1 Overview of Subtree-based Parse Reranking

The task of reranking is similar to that of parsing, except that the search of the parse tree is performed on a K-best list with selected parse candidates, rather than the entire search space:

$$\tilde{y} = argmax_{y \in Kbest(x)} Score'(y, x) \quad (4)$$

The scoring function is defined as:

$$Score'(y,x) = L(y,x) + w \cdot f(y,x) \quad (5)$$

where $L(y,x)$ is the score of y output by the base parser. We define the oracle parse y^+ to be the parse in the K-best list with the highest accuracy, as compared with the gold-standard parse. The goal of reranking is to learn the weight vector so that the reranker can pick up the oracle parse as many times as possible. Note that in the reranking framework, the feature $f(y,x)$ is defined on an entire parse tree y, which enables the encoding of global information.

Algorithm 3: Perceptron Learning for Reranking

```
1  for iteration t = 1..T
2    for training data (x_i, Kbest(x_i), y_i^+), i = 1..N
3      ỹ = argmax_{y∈Kbest(x)} L(y, x_i) + w · f(y, x_i)
4      if ỹ ≠ y_i^+
5          w ← w + f(y_i^+, x_i) − f(ỹ, x_i)
6    end for
7  end for
```

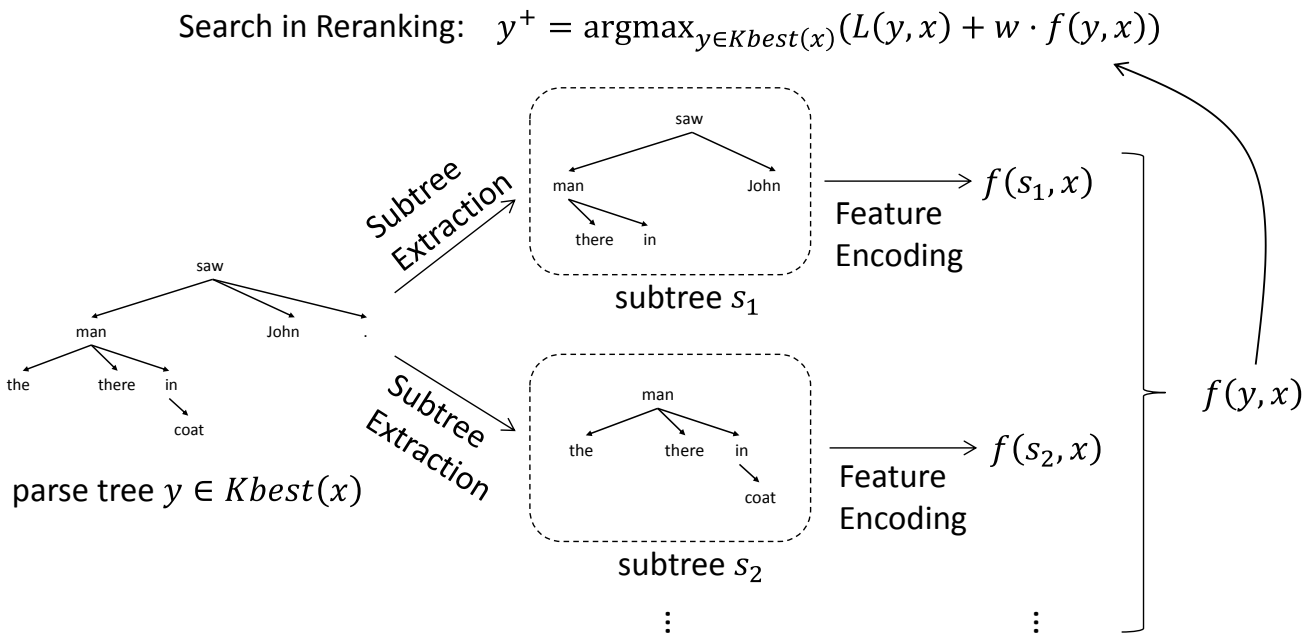


Figure 8. Feature evaluation in reranking. This is processed in two stages: "Subtree Extraction" and "Feature Encoding".

Figure 8 illustrates the evaluation of $f(y,x)$ in Equation (5). The evaluation is performed in two stages: in the *subtree extraction* stage, we select a set of subtrees of the current parse tree that are concrete and informative. These provide linguistic evidence to discriminate good parses from bad ones. In the *feature encoding* stage, we encode each extracted subtree into a string representation with proper lexical, grammatical, and structural information.

We illustrate the learning algorithm for a K-best reranker in Algorithm 3. This algorithm is modified from the one used for learning a parser.

In the following subsection, we will introduce the string representation of tree structures, for an easier description of the two sub-processes of our reranking framework.

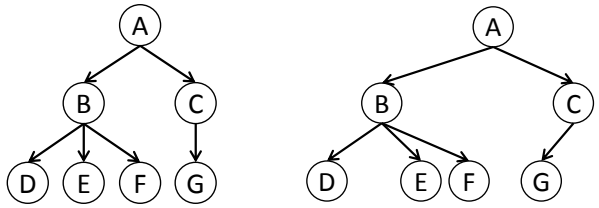


Figure 9. Two types of tree structures. (a) A rooted ordered tree. (b) A dependency tree, where children are either on the left or on the right of its head.

5.3.2 String Representation of Tree Structures

The methods of encoding subtrees in the string representation have been studied in the field of structured data mining. Asai et al. (Asai et al., 2003) and Nijssen and Kok (Nijssen and Kok, 2003) have independently defined equivalent encoding methods for *rooted ordered trees* using depth sequences. An example of *rooted ordered tree* is illustrated in Figure 9(a). In their encoding, the depth of a vertex is explicitly recorded in the string. A pre-order depth-first traversal is used to transform the tree structure into a string. For the rooted ordered tree shown in Figure 9(a), Asai's and Nijssen's string encoding is:

$$S_T = ((\mathrm{A}, 0), (\mathrm{B}, 1), (\mathrm{D}, 2), (\mathrm{E}, 2), (\mathrm{F}, 2), (\mathrm{C}, 1), (\mathrm{G}, 2)),$$

where each pair represents a vertex: the 1st symbol in the pair is the vertex label, and the 2nd number is the depth of the vertex.

A *dependency tree* is a *rooted ordered tree* that every vertex except the root is either a left child or a right child of another vertex. An example of *dependency tree* is illustrated in Figure 9(b).

For a *dependency tree*, we adapt Asai's and Nijssen's encoding with some modifications. For a node in a dependency tree, we encode its label, its depth, and a binary value which indicates whether the node is on the left of its head or not. For the ease of implementing the

subtree extraction component which we will introduce in the next section, we use an in-order depth-first traversal to transform the tree into a string. For the dependency tree shown in Figure 9(b), our encoding is

$$S_T = ((\mathrm{D}, 2, \mathrm{L}), (\mathrm{B}, 1, \mathrm{L}), (\mathrm{E}, 2, \mathrm{R}), (\mathrm{F}, 2, \mathrm{R}), (\mathrm{A}, 0, -), (\mathrm{G}, 2, \mathrm{L}), (\mathrm{C}, 1, \mathrm{R})),$$

where each 3-tuple represents a node: the 1st symbol is the node's label, which in practice we will substitute with the node's word form, the part-of-speech tag, or the combination of them. This symbol may also be replaced by a hyphen, which means the word form and the POS tag of this nodes are omitted in a feature. The 2nd number is the depth of this node. For a subtree that is extracted from a parse tree, the depth is counted from the root of the subtree and not the root of the parse tree. The 3rd symbol indicates the branching direction of the node. For the root node, however, this symbol is replaced with a hyphen since the depth is not applicable.

5.3.3 Subtree Extraction

Benefitting from the K-best list obtained in the parsing stage, we can perform discriminative learning to select a good parse from among candidates in a smaller search space, which allows the utilization of global features. We define three types of features below: *trimmed subtrees*, which represent a subset of subtrees in a dependency parse that have compact structures; *coupled subtrees*, which are composed of two linked trimmed subtrees; and *sibling subtrees*, which is an extension of sibling structures that both sibling modifiers are expanded by a trimmed subtree.

Trimmed subtree: for each node in a given parse tree, we check the subtrees it dominates to see whether they are likely to appear in a good parse tree. To efficiently obtain these subtrees, we set a local window that bounds a node from its left, right, and below. We then extract the maximum subtree inside this window, i.e., we cut off those nodes that are too distant in sequential order or too deep in a tree.

The above subtree extraction often results in a very large set of instances that are extremely sparse in the training data. Therefore, it is necessary to keep smaller subtrees as back-offs. In most cases, however, it is prohibitively expensive to enumerate all the smaller subtrees.

Instead of enumeration, we design a back-off strategy that selects subtrees by discarding nodes that are far away from the subtree's root, and keeping those that are nearby. Specifically, after extracting the first subtree of a node, we vary the three boundaries (left, right, and below) from their original positions to positions that are closer to the root of the subtree, thus tightening up the local window. For each possible combination of the variable boundaries, we extract the largest subtree from the new local window, and add it to the set of so called “trimmed subtrees” of the node. We illustrate the detailed extraction process of trimmed subtrees in Algorithm 4.

Algorithm 4: Trimmed Subtree Extraction

```
Input: Parse tree T, Boundary parameters L, R, H
Initialization: nodeList_l ← ∅, nodeList_r ← ∅,
anchorList_l ← ∅, anchorList_r ← ∅, treeString ← ∅
Procedure:
for w ∈ vertices of T
    (nodeList_l, nodeList_r) = DepthFirstTravel(w, T)
    ◁ In-order depth-first traversal from w that stores w's
        left and right descendent in separate lists
    anchorList_l = FindAnchors(nodeList_l)
    anchorList_r = FindAnchors(nodeList_r)
    ◁ Find "anchor" nodes which are on the outmost path
        from w to leaves
    for i = H, H-1, . . ,1
        anchorList_l.deleteAboveDepth(i)
        anchorList_r.deleteAboveDepth(i)
        ◁ Remove nodes with depth larger than i
    end for
    for j ∈ anchorList_l, k ∈ anchorList_r
        s = nodeList_l.lastNode()
        until s == j.previousNode()
            treeString.append(s)
            s = nodeList_l.getPrevious(s)
        treeString.append(w)
        t = nodeList_r.firstNode()
        until t == k.nextNode()
            treeString.append(t)
            s = nodeList_r.getNext(t)
        w.subTreeList().add(treeString)
        ◁ Add a subtree string into Trim(w), the set
            of trimmed subtrees rooted at w.
        treeString.clear()
    end for
end for
```

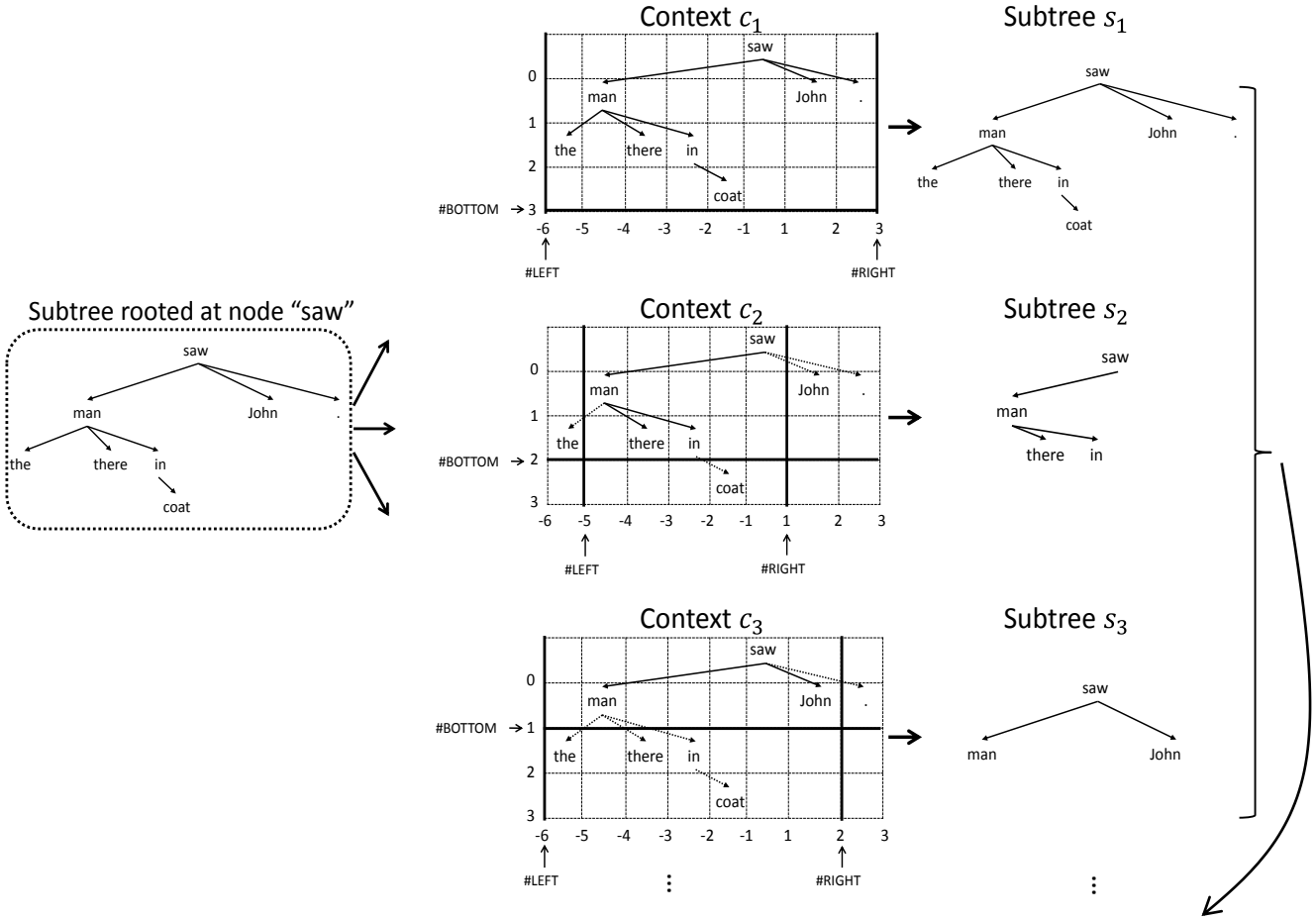


Figure 10. Extraction of trimmed subtrees from the node “saw”. “#LEFT”, “#RIGHT”, and “#BOTTOM” represent the three boundaries that can vary along the corresponding axis. Contexts c_1, c_2, and c_3 represent three instances of possible combinations of boundary positions. s_1, s_2, and s_3 are the resulting subtrees, which are elements of the trimmed subtrees set of the node “saw”.

This back-off strategy is based on our observation that nodes that are close to the root may provide more reliable information than those that are distant. In n-gram language modeling, the likelihood of a sentence is estimated by multiplying the conditional probability of each word given a context, where the context stands for a window of preceding words. We estimate

a parse tree's likelihood by multiplying the likelihood of subtrees rooted at each node, where each subtree is extracted from a restricted context. Here restricted context means a 2-dimentional window which selects nodes that are close to the root of the subtree, but not in terms of the linear distance between them, as opposed to the case of a sequence of words. The assumptions behind both are very similar: nodes or words in closer distance tend to have stronger correlation and therefore provide more relevant and reliable information.

Figure 10 illustrates the construction of the set of the trimmed subtrees rooted at the node "saw" for the sentence "the man there in coat saw John." The initial boundary parameters are sufficiently large that the local window contains the entire parse tree. #LEFT, #RIGHT, and #BOTTOM represent the three boundary variables, which range from -6 to -1, 3 to 1, and 3 to 0, respectively. Contexts c_1, c_2, and c_3 represent three different combinations of boundary positions. Subtrees s_1, s_2, and s_3 are the extracted subtrees for the corresponding context. They and other similarly extracted subtrees together comprise the set $Trim$("saw"), the trimmed set of subtrees for the node "saw". We use this set in two ways. First, for each element, we encode a series of features. Second, this set is reused in another two types of features, which will be described later. We repeat this extraction process for all nodes in a parse tree, and keep the sets of trimmed subtrees for the feature encoding process.

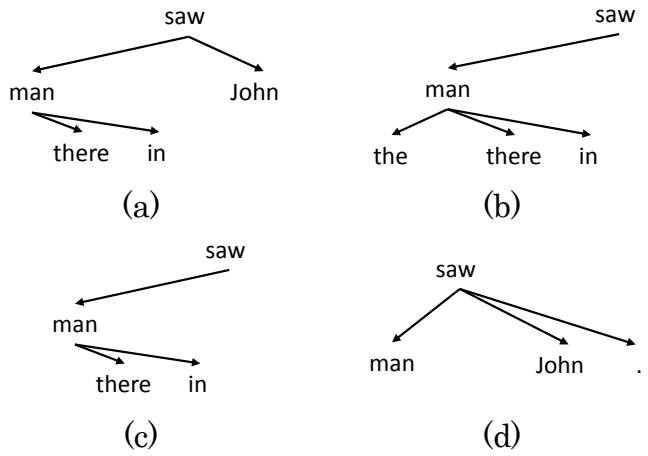


Figure 11. Some of the extracted trimmed subtrees, where (c) is identical to a grand-sibling factor in a third-order parsing model and (d) is similar to a tri-sibling factor, but with siblings on both sides of the head.

In Figure 11, we illustrate some of the extracted subtrees in the set $Trim$("saw"). Among these, subtree (c) can be regarded as a grand-sibling factor and subtree (d) is similar to a tri-sibling factor (Koo and Collins, 2010), except the siblings are located on both sides of the head node. Subtrees (a) and (b) cannot be represented in common factorization methods, which confirms the ability of this feature set to capture a large variety of structures.

It should be noted that, while in a direct calculation there are 72 (6×3×4) possible combinations for boundary positions in the example in Figure 10, this number can almost always be reduced in practice. In this example, when #LEFT reached the index -4, the entire left branch of the root node was cut, so no further movement for #LEFT was allowed. Moreover, after #BOTTOM moved to index 1, the sequential order distance between “man” and “saw” was updated and reduced to 1, which restricts #LEFT to only two possible positions, either left or right of the word “man”. Therefore, it can be verified that the true number of combinations of boundary positions is actually 25. Briefly, for a given node, we decompose the subtree extracted from the initial local window into three parts: the node itself, the sequence of its

left descendants, and the sequence of its right descendants. The two sequences of descendants are preordered in a depth-first search, during which we mark "anchor" nodes as the next-possible cut-in positions for the left/right boundary variables. Furthermore, the list of anchor nodes will be updated whenever the bottom boundary variable moves to a new position. As a result, we are able to minimize the number of boundary combinations to speed up the subtree extraction.

Coupled subtree: with trimmed subtrees, we are able to utilize the most informative substructures in a dependency parse by removing redundant nodes and edges. However, we have observed some instances in which the long-distance prediction of dependency becomes even worse than the baseline system. This is because, during the trimming process, nodes tend to be centralized to the root. To compensate for this, we propose a complementary structure called a coupled subtree.

The basic idea of coupled subtrees is to augment the head and modifier of a dependency edge with the trimmed subtrees rooted on them. By combining two trimmed subtrees with a dependency edge, the subtree structure is no longer restricted to a 2-dimensional window. Moreover, there is no need to search any "new" subtrees, because a coupled subtree is constructed by combining previously extracted trimmed subtrees, which can easily be done using trimmed subtrees stored in memory.

Specifically, for each dependency edge $h - m$ in a candidate parse, its corresponding coupled subtree features are the collection of all the pairs:

$$\langle f(h, p_1, i_1), f(m, p_2, i_2) \rangle$$

where h and m represent the head and modifier of a dependency. The expression $f(a, p, i)$ represents the i_{th} feature encoded on a trimmed subtree in the set $Trim(a)$, such that the trimmed subtree is the one extracted within the local window p.

Sibling subtree: we extend the idea behind coupled subtrees from augmenting single dependency edges to sibling edges. We define the sibling subtree structures for reranking as a natural extension of sibling factorization (McDonald and Pereira, 2006) from the word-based case to the subtree-based case.

Specifically, for each node m in a candidate parse, its sibling subtree features are the collection of all 3-tuples:

$$\langle h, f(m_1, p_1, i_1), f(m_2, p_2, i_2)\rangle$$

where h represents the word form, the part-of-speech tag, or the combination of the word form and the part-of-speech tag of the head node of m_2. m_1 is the nearest sibling node of m_2 between h and m_2, and the expression $f(a, p, i)$ represents the i_{th} feature encoded on a trimmed subtree in the set $Trim(a)$, such that the trimmed subtree is the one extracted within the local window p. As mentioned before, by retaining the trimmed subtree extraction history, we eliminate the need to re-compute any subtree structures on the sibling nodes.

5.3.4 Feature Encoding

With an extracted trimmed subtree, we encode a set of features. For discriminative structural classifiers to learn to make good inference, features must represent different levels of the structural, grammatical, and lexical information about a subtree. This can be achieved, in the case of factorized dependency parses where a factor is a small subtree that usually consists of at most three edges, by designing a template to enumerate the back-off features (Koo and Collins, 2010; McDonald et al., 2005; McDonald and Pereira, 2006). However, it is difficult to directly apply a feature template to trimmed subtrees. This is because a trimmed subtree usually has more nodes as well as a more complex structure than a factor in factorized models, which can lead to sparseness problem if all the nodes are used in feature encoding. It is therefore necessary to design a node enumeration or selection method for encoding back-off features.

A naïve way to overcome the difficulty of designing feature template seems to be enumerating all the subtrees of the trimmed subtree, and use this set of subtrees to indicate in turn which nodes of the trimmed subtree will be included in a feature. This strategy, however, is usually infeasible due to the fact that there are exponentially many such subtrees.

We therefore use the following strategy to address this problem: for each trimmed subtree t rooted at a node r, we find a set of trimmed subtrees $\{t_1, t_2, \ldots, t_n\}$ such that $t_i \in Trim(r)$ and t_i

is a subtree of t. This set can be found easily by storing and sorting the extracted trimmed subtrees according to the values of their context variables. For each pair of trimmed subtrees $\langle t, t_i \rangle$, we divide the nodes of t into three categories, namely,

(1) the root node r;
(2) $\{a | a \in t \cap t_i\}$;
(3) $\{b | b \in t \setminus t_i\}$.

For each a, we then encode its part-of-speech tag, branching direction, and depth. For each b, we ignore the part-of-speech tag and encode only its structural information. For the root node r, we encode two different levels of back-off information: either the part-of-speech tag, or both the word form and part-of-speech tag. Each version of r is combined with the other nodes that generate a feature.

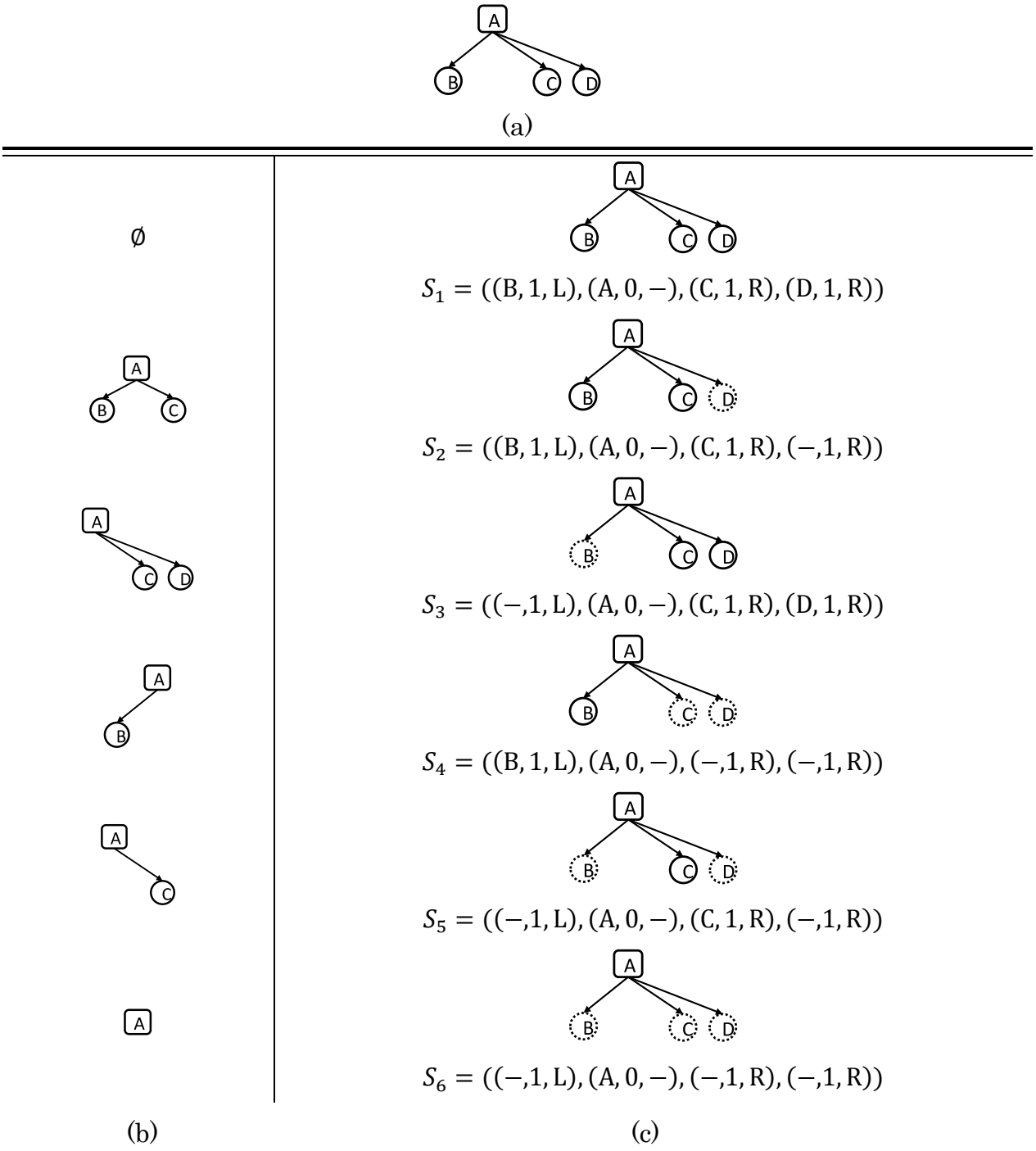


Figure 12. Feature encoding for trimmed subtrees. (a) A trimmed subtree rooted at node A. (b) Trimmed subtrees in $Trim(A)$ with smaller context. (c) Corresponding subtree features: root nodes are encoded in most detail; nodes represented as circles with solid border are encoded without considering word forms; other nodes are encoded with structural information only.

This process is illustrated in Figure 12. Suppose we are encoding features for a trimmed subtree as shown in Figure 12(a). For this trimmed subtree, we first collect its smaller subtrees that are also in $Trim(A)$. This can be easily done by keeping $Trim(A)$, the list of trimmed subtrees with the same root node A, during the subtree extraction stage. These subtrees are shown in Figure 12(b). Then, by comparing with each subtree in Figure 12(b), we divide the nodes in the original trimmed subtree into three categories: root node, nodes that are also in the smaller subtree (represented with solid circles), and nodes that are not (represented with dash circles). The original trimmed subtree is then transformed into string representation, where each node is encoded with different level of information. These string representations and the corresponding subtree illustrations are shown in Figure 12(c). Finally, for each string representation, two feature strings are derived by substituting the first element in the 3-tuple of each node with its word form and/or POS tag. For example, the two feature strings derived from the string representation S_3 in Figure 12(c) are

$$f_1 = ((-,1,\mathrm{L}),(pos(\mathrm{A}),0,-),(pos(\mathrm{C}),1,\mathrm{R}),(pos(\mathrm{D}),1,\mathrm{R}))$$

and

$$f_2 = ((-,1,\mathrm{L}),(word(\mathrm{A}),pos(\mathrm{A}),0,-),(pos(\mathrm{C}),1,\mathrm{R}),(pos(\mathrm{D}),1,\mathrm{R}))$$

This method has several advantages: first, the set of smaller subtrees provide automatic "templates" of features which control what combination of information to encode for each node. Second, this method is efficient since there is no need to extract or enumerate any new subtrees. Third, this method handles feature sparseness problem by encoding back-off features, which on most of the nodes only keep information regarding subtree's structure and discard sparse information such as word form and part-of-speech.

Given the features of trimmed subtrees, it is straightforward to encode features for coupled subtrees and sibling subtrees according to their definitions, as described in the previous sub-

section.

5.4 Evaluation

We present experimental results for two languages, English and Chinese.

5.4.1 Settings

For the English experiment, we use the Penn Treebank Wall Street Journal part. We convert the constituent structure in the Treebank into a dependency structure with the Penn2Malt tool, and use a previously reported head-extraction rule (Yamada and Matsumoto, 2003). To align our experiments with previous work, we use the standard data division: sections 02-21 for training, section 24 for development, and section 23 for testing. As our system assumes part-of-speech tags as input, we use MXPOST, a MaxEnt tagger (Ratnaparkhi, 1996), to automatically tag the test data. The tagger is trained using the same training data.

For the Chinese experiment, we use the Chinese Treebank 5.0 with the following data division: files 1-270 and files 400-931 for training, files 271-300 for testing, and files 301-325 for development. We use Penn2Malt to convert the Treebank into a dependency structure, and a different set of head-extraction rules (Zhang and Clark, 2008). Moreover, for Chinese, we use the gold standard part-of-speech tags in the evaluation.

We use the unlabeled attachment score (UAS) to measure the effectiveness of our method. The UAS is the percentage of words with correctly identified heads. For each experiment, we use the parameters tuned by the development set.

We train two base parsers, which are re-implementations of the first- and second-order parsers in the MSTParser, with 10 iterations on the English and Chinese training datasets (McDonald et al., 2005; McDonald and Pereira, 2006). We use 30-way cross-validation on the identical training dataset to provide training data for the rerankers. The left, right, and bottom boundaries for the trimmed subtree features that we use throughout the experiments are 10, 10, and 5 respectively, which are tuned on the development data of the Penn Treebank. For the main experiments, we set K, the size of the list of parse tree candidates, to be 50.

5.4.2 Experimental Results

The experimental results for English are shown in Table 26. Each row in this table shows the UAS of the corresponding system. “McDonald05” and “McDonald06” stand for the first- and second-order models in the MSTParser (McDonald et al., 2005; McDonald and Pereira, 2006). “Zhang11” denotes the transition-based parser proposed by Zhang and Nivre (Zhang and Nivre, 2011). “Koo10” denotes Model 1 in Koo and Collins (Koo and Collins, 2010), which is a third-order model. “Martins10” is the turbo parser proposed by Martins et al. (2010). “Hall07” stands for a Maximum Entropy k-best reranker (Hall, 2007). “Sangati09” stands for a discriminative k-best parser pipelined with a generative reranker (Sangati et al., 2009). “Hayashi11” is a forest reranker based on a variant of Eisner's generative model (Hayashi et al., 2011). “Roux12” stands for the system which combines a generative k-best constituent parser with a discriminative dependency reranker (Roux et al., 2012). “Ren13” is a discriminative reranking system which employs a k-best PCFG parser (Ren et al., 2013). “Order 1” and “Order 2” are our re-implementations of the MSTParser, and are used as the base parsers for our reranking experiments. “Order 1 reranked” and “Order 2 reranked” are the rerankers pipelined on the two base parsers. “Koo08”, “Chen09”, “Suzuki09”, and “Chen13” are parsers that use semi-supervised methods (Koo et al., 2008; Chen et al., 2009; Suzuki et al., 2009; Chen et al., 2013). In Table 27, we show the results for the Chinese experiment. “Duan07” and “Yu08” denote two different probabilistic parsers (Duan et al., 2007; Yu et al., 2008), “Chen09” and “Chen13” denote the same systems as in Table 26, and “Chen12” denotes a semi-supervised parser which utilizes automatic subtrees collected from large-scale unannotated data (Chen et al., 2012). Note that although the score reported for “Chen13” is lower than other systems, it is largely due to the fact that automatic POS tags are used during the evaluation.

System	English UAS
McDonald05	90.9
McDonald06	91.5
Zhang11	92.9
Koo10	93.04
Martins10	93.26
Hall07*	87.81
Sangati09*	93.09
Hayashi11*	92.89
Roux12✧	93.9
Ren13✧	93.71
Order 1	90.91
Order 2	91.88
Order 1 reranked*	92.22
Order 2 reranked*	**93.42**
Koo08+	93.16
Chen09+	93.16
Suzuki09+	93.79
Chen13+	93.77

Table 26. English UAS of previous work, our base parsers, and reranked results. "+": Semi-supervised parsers. "*": Dependency parse rerankers. "✧": Constituent-to-dependency parsers.

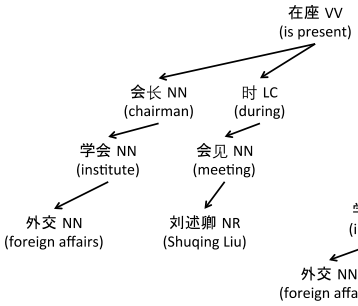
System	Chinese UAS
Duan07	84.36
Yu08	87.26
Order 1	85.44
Order 2	87.39
Order 1 reranked	88.47
Order 2 reranked	**89.25**
Chen09+	89.91
Chen12+	91.59
Chen13+	83.08

Table 27. Chinese UAS of previous work, our base parsers, and reranked results. "+": Semi-supervised parsers.

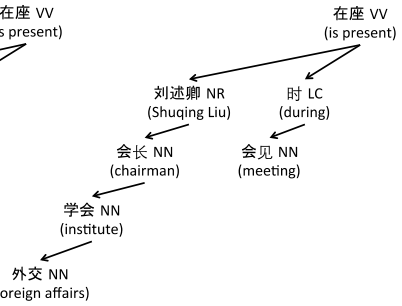
As we can see from the results, for English, the accuracy increased from 90.91% ("Order 1") to 92.22% ("Order 1 reranked") for the first-order parse reranker, and from 91.88% ("Order 2") to 93.42% ("Order 2 reranked") for the second-order parse reranker. For Chinese, the accuracy increased from 85.44% to 88.47% for the first-order parse reranker, and for the second-order case it increased from 87.39% to 89.25%. This shows that our best reranking system obtains the highest accuracy among supervised systems, except for "Roux12" and "Ren13", which are both based on constituent parsers. Our rerankers are also outperformed by semi-supervised systems such as "Suzuki09" and "Chen09", but as our method is orthogonal to semi-supervised methods, it is possible to further improve the accuracy by combining these techniques.

5.4.3 Analysis of Results

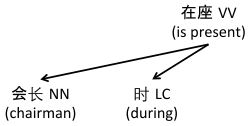
In this section, we present our analysis of the reranking models. We have investigated the influence of subtree features of different types and sizes in reranking accuracies, the oracle accuracies of the base parser for K values in the range 5–500, the coverage of the subtree extraction algorithm, and the differences in the processing time between the base parser and the reranker.



(a)



(b)

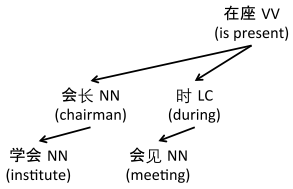


$Score_{parse} = 25.50$

(c)

$Score_{parse} = 12.14$

(d)

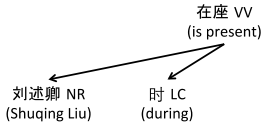


$Score_{rerank} = 18.77$

(e)

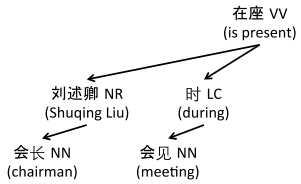
$Score_{rerank} = 4.82$

(f)



$Score_{parse} = 21.06$

(g)



$Score_{rerank} = 26.92$

(h)

Figure 13. The 1-best parse, the reranked parse, and their scored subtrees of the sentence “外交学会会长刘述卿会见时在座” (The chairman of the Institute of Foreign Affairs, Shuqing Liu, is present in the meeting).

1-best Versus K-best We first show an example in Figure 13 of how the reranker chooses a correct parse from the candidate list when the 1-best output of the parser is incorrect. Figure 13(a) shows the output of the baseline parser for a sentence in the testing data of CTB5: "外交学会会长刘述卿会见时在座" (The chairman of the Institute of Foreign Affairs, Shuqing Liu, is present in the meeting). The parser made several mistakes: the person's name "Shuqing Liu" is wrongly attached to "meeting", and "chairman" is attached to the predicate "is present". Figure 13(b) shows the correct parse, which is the output of the reranker. In Figure 13(c) - (h) we show some subtrees of the two parses which are either factorized by the parser or extracted by the reranker. We also show the total feature scores of these subtrees. In the 1-best parse tree, the subtrees shown in Figure 13(c) and Figure 13(d) are given relatively high scores by the parser. This is because subtree structures such as a verb ("VV") having two sibling modifiers on its left with POS tags "NN" and "LC", as well as a word "会见" (meet) modified by a person's name, can both be frequently observed in the training data of CTB5. The reranker however gives much lower scores to subtrees in Figure 13(e) and Figure 13(f), where the former is an extension of the subtree in Figure 13(c) that takes the grandchildren of the root into consideration, and the later includes the head of the subtree in Figure 13(d). On the other hand, the score given by the parser to the subtree in Figure 13(g), a subtree with the root "在座" (be present) from the correct parse tree, is lower than the score given by the reranker to the subtree in Figure 13(h), which is also a result benefit from the larger context employed in the reranker.

System	UAS
$\text{Reranker}_{\text{Trim+Couple+Sib}}$	93.32
$\text{Reranker}_{\text{Trim}}$	92.76
$\text{Reranker}_{\text{Trim+Couple}}$	93.08
$\text{Reranker}_{\text{Trim+Sib}}$	93.18

Table 28. Influence of activated feature types of English development data.

System	UAS
Reranker	89.29
Parser (2nd-order)	87.55
Reranker (3rd-order)	88.14
Reranker (4th-order)	88.50
Reranker (5th-order)	88.61

Table 29. Influence of maximum order of activated subtrees on Chinese development data.

K	1	10	20	30	50
English	91.88	95.61	96.30	96.65	97.02
Chinese	87.39	90.43	91.28	92.02	92.54

Table 30. Oracle accuracies of top-K candidates.

System	Parsing	50-best candidate generation	Reranking
Order 2	0.24 sec/sent	N/A	N/A
Order 2 reranked	11.54 sec/sent	9.96 sec/sent	1.86 sec/sent

Table 31. Processing speed comparison for English.

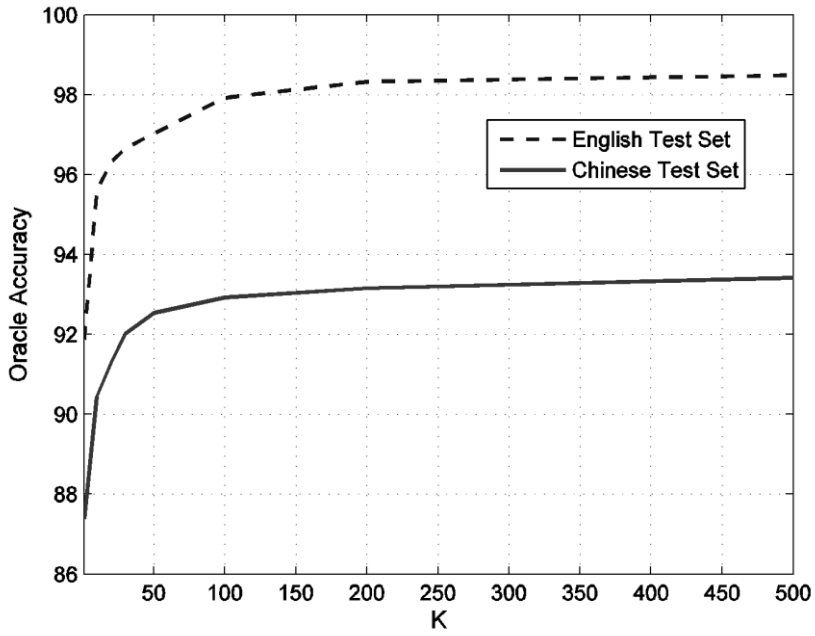


Figure 14. Oracle accuracies versus different K values on test datasets, with solid lines representing English and dashed lines representing Chinese.

Feature Ablation Study We investigated the effects of the three types of subtree features. Based on the "Order 2 reranked" system for English, we trained a series of rerankers that activate each feature type and their combinations using the Penn Treebank. We tested these models on the development data, the results are shown in Table 28. The first row represents the system with all feature types activated; the other rows are systems with corresponding feature sets activated in the training and evaluation.

In Table 29, we show the impact of the subtree features by training reranking models with a limit on the maximum size of activated subtrees. The first two rows show the result of the 2nd-order parser and the reranked result for the development data of the Penn Treebank. The rows in the lower part of the table show the results of a series of rerankers that we trained with a limitation of the size of the activated subtrees. For example, "Reranker (4th-order)" stands for the reranker which activates subtrees up to 4th-order. These results show,

3rd-order subtree features contribute the most to the improvement over the base parser. As the size of the activated subtrees grows, the improvement the model yields becomes smaller. However, the large difference between the results yielded by "Reranker (5th-order)" and by the reranking model activating all extracted subtrees suggests that features from large subtrees are also important to the final result.

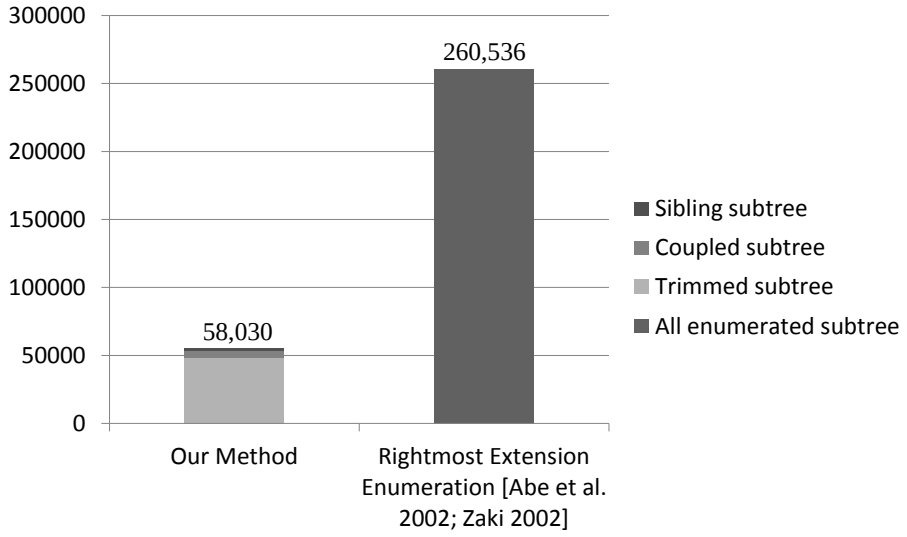


Figure 15. Ratio of subtrees extracted by our method compared with exhaustive enumeration on 350 sentences in the CTB5 dev set.

Oracle Accuracies In Table 30, we show the oracle accuracies of the top-K candidates using the "Order 2" parser. The oracle accuracies can increase by as much as 5.14% in absolute terms for English, and 5.15% for Chinese, when compared with the 1-best accuracies.

Furthermore, in Figure 14 we show the oracle accuracies for K values of up to 500 for the two languages. The change in oracle accuracies shows little difference for K = 200 and K = 500. This indicates a convergence to the oracle accuracy, which is about 98.5% for English and about 93.9% for Chinese.

Coverage of Subtree Extraction We also investigated the ratio of subtrees that our method is able to extract compared to all the subtrees of a dependency parse. To count the number of subtrees of a dependency parse, we implemented the Rightmost Extension algorithm (Abe et

al., 2002; Zaki, 2002), which is an efficient subtree enumeration algorithm. We compared the number of extracted subtrees on the development data of CTB5. The result is shown in Figure 15.

Our method extracts about 22% of all subtrees, among which the dominant type are trimmed subtrees. This result indicates that our subtree extraction method removes a large number of the redundant subtrees, which leads to an efficient feature encoding process. Although for a complete comparison it is ideal to directly evaluate the performance of a reranker based on all enumerated subtrees, in practice it is hard to train such a model in limited time, given that not only the amount of subtrees are boosted considerably, but also the efficient feature encoding process we have described in section 3.3 becomes no longer applicable. The key to our feature extraction method is that, by keeping a short list of trimmed subtrees for each node during the subtree extraction stage, there is no need to extract any new subtrees during the feature encoding stage. These trimmed subtrees can then be used as "templates" of subtree features. A reranker based on all enumerated subtrees, however, has no such advantage. It has to go through all possible subtrees, in both subtree extraction and feature encoding stage.

It should also be noted that, given the nature of our subtree extraction method, the exact ratio of subtrees that will be extracted depends largely on the structure of the dependency parse. Thus, the average ratio may vary for different languages or domains.

Efficiency We show the processing speed of the base parser "Order 2" and the pipelined reranker "Order 2 reranked" in Table 31. Both systems operated on a Xeon 2.4 GHz CPU. We calculated the parsing time by running the systems on the first 100 sentences of the development data of the Penn Treebank. As the table shows, the reranker is much slower than the base parser with a speed factor of 48 in processing new sentences, which is mainly due to the time needed to produce the 50-best candidates list. In fact, the results suggest that over 80% of the processing time is consumed by the candidate generating process. This can be seen as an unavoidable trade-off to obtain high accuracy in the reranking framework. However, as the running speed of the reranker is dominated by the K-best parser, it is possible to improve the system's efficiency by employing faster K-best parsing algorithms in the future.

5.5 End-to-end Evaluation

So far we have described all the proposed methods for improving Chinese word segmentation, part-of-speech tagging and dependency parsing in previous chapters. Although these methods have been evaluated individually, theoretically these methods are non-overlapping and therefore can be used together to obtain the benefits from each single components. In this subsection, we demonstrate the effectiveness of systems which use a combination of these methods.

5.5.1 Settings

We have conducted word segmentation and part-of-speech tagging experiments on the re-annotated CTB5, and we have further conducted word segmentation, part-of-speech tagging and dependency parsing experiments on the original CTB5. Due to the fact that we have re-annotated a subset of three thousand sentences in CTB5 with dependencies, which doesn't include the test set, we have evaluated the dependency parsing performance only on the original CTB5.

For the maximized substring method, we have used the same setting of "MaxSub-U" which has been described in section 3.4, which uses both the test set and the Chinese Gigaword as unlabeled data for maximized substring extraction. For our method that utilizes the character-level POS information, we have used manual the annotation in the training set of CTB5. For the subtree-based parse reranking method, we have used the same setting of "Order 2 re-ranked" in the previous subsection.

In all the following experiments, the baseline morphological analyzer we use for word segmentation and part-of-speech tagging is the same system we have described in section 4.2; the baseline dependency parser is the second-order graph-based parser we have introduced in section 5.4.

For the experiments conducted on the re-annotated CTB5, we training, dev, and test set are the same as we have described in section 2.4.1. For the experiments conducted on the original CTB5, we have adopted the same data division as in (Jiang et al., 2008a; Jiang et al., 2008b; Kruengkrai et al., 2009; Zhang and Clark, 2010; Sun, 2011): the training set, dev set and test set have 18,089, 350 and 348 sentences, respectively.

All the parameters we have used in this evaluation are tuned on the CTB5 dev set. We have trained all models using the averaged perceptron algorithm (Collins 2002). We use precision, recall and F-score to measure the performance of our word segmentation and POS tagging systems, and unlabeled attachment score (UAS) for dependency parsing systems.

5.5.2 Experimental Results

We first present the experimental results of morphological analyzing on the re-annotated CTB5, with the maximized substring and/or character-level POS-based methods. The F-score each models evaluated on the test set of CTB5 are shown in Table 32. The baseline model is denoted as "Re-annotated", and the three other models, denoted as "+MaxSub", "CharPos" and "+MaxSub+CharPos", represent a combination of the baseline with the maximized substring method, a combination of the baseline with the character-level POS-based method, and the combination of the baseline with both methods, respectively. The results show that, "+MaxSub" and "+MaxSub+CharPos" both significantly outperformed the baseline in word segmentation as well as POS tagging evaluation, which suggests that the maximized substring method is compatible with our re-annotation strategy and can obtain a F-score of 94.84 in joint word segmentation and POS tagging. On the other hand, "CharPos" didn't show its effectiveness when applied on the re-annotated treebank. We consider the reason behind to be that, given the fact that our re-annotation strategy for word segmentation and POS tagging have already take character-level information into consideration, the character-level POS annotation in the same corpus became partially redundant. In fact, our re-annotation strategy introduced in chapter 2 was largely motivated by the results of the experiments using character-level POS information, which we have described in chapter 4.

We further show the experimental results of morphological analyzing and dependency parsing on the original CTB5 in a pipelined setting: the output of the morphological analyzers are used as the input of the dependency parsers. The results are shown in Table 33: "Baseline" denotes the baseline morphological analyzer and dependency parser models; "+MaxSub", "CharPos" and "+MaxSub+CharPos" denote the corresponding morphological analyzer models in Table 32 with the baseline dependency parser model; rows with "+Rerank" and the row "+All" denote the corresponding morphological analyzer models with the parse

reranking model. The results show that, using any combination of the methods we proposed in a single pipelined system can significantly improve its accuracies in word segmentation, POS tagging and dependency parsing.

5.6 Summary

In this chapter we have proposed a novel feature set for dependency parse reranking that successfully extracts complex structures for collecting linguistic evidence. We also presented an efficient feature back-off strategy to relieve data sparsity. Through a series of experiments, we confirmed the effectiveness and efficiency of our method, and observed significant improvement over the base system as well as other known systems. In the end-to-end evaluation, we have demonstrated the effectiveness of systems which use a combination of methods proposed in this chapter and previous chapters.

	SEG	POS
Re-annotated	98.28	94.46
+MaxSub	98.54*	94.80*
+CharPos	98.29	94.65
+MaxSub+CharPos	98.59*	94.84*

Table 32. Experimental results on re-annotated CTB5 with maximized substring and/or character-level POS information. "*": $p < 0.05$ in McNemar's test vs. Baseline.

	SEG	POS	DEP
Baseline	97.96	93.48	78.65
+MaxSub	98.26*	93.94*	79.09*
+CharPos	98.03	93.80*	78.92
+MaxSub+CharPos	98.28*	94.04*	79.15*
+Rerank	97.96	93.48	79.27*
+MaxSub+Rerank	98.26*	93.94*	79.63*
+CharPos+Rerank	98.03	93.80*	79.57*
+All	98.28*	94.04*	79.81*

Table 33. Experimental results on CTB5 with maximized substring, character-level POS information, and/or subtree-based reranking. "*": $p < 0.05$ in McNemar's test vs. Baseline.

Chapter 6

Conclusion

In this thesis, we have proposed an entire framework, from treebank annotation to dependency parse reranking, to address the various issues observed in Chinese syntactic analysis, and have demonstrated the effectiveness of all the methods through comprehensive experiments.

First, we have proposed a complete set of annotation guidelines for Chinese word segmentation, a tagset for part-of-speech tagging and a label set for dependency labelling to overcome the inconsistency and data sparsity problems in existing treebanks.

Second, we have proposed an algorithm that extracts substrings as reliable word boundary indicators which significantly enhance the accuracy of word segmenters. The algorithm takes linear time in the average case, which is a big advantage in processing large-scale data.

Third, we have investigated the usefulness of character-level POS in the task of Chinese part-of-speech tagging. We have proposed the first tagset designed for the task of character-level POS tagging, based on which we manually annotate the entire CTB5. We have further proposed a method that performs character-level POS tagging jointly with word segmentation and word-level POS tagging.

Finally, we have explored a feature set that makes fully use of dependency grammar, can capture global information with less restriction in the structure and the size of the sub-tree context, and can be encoded efficiently. It exhaustively explores a candidate parse tree for features from the most simple to the most expressive while it maintains the efficiency in terms of training and parsing time.

6.1 Future Work

In Chapter 1, we have described our Chinese word segmentation annotation approach which recognizes the morphemes in a compound based on the analysis of its internal structure as words; the internal structure between the morphemes is further captured in our POS tagging and dependency labeling annotation. The smallest units in this set of annotation methods are still (monosyllabic or disyllabic) morphemes instead of characters.

It has been demonstrated in (Zhang et al., 2013) that the accuracy of dependency parsing can be improved through expanding the existing annotation with the information of dependency relations between characters. However, in the end-to-end evaluation, the models which added character-level POS information on top of other methods didn't show the extra improvements we have expected. We believe that the current character-level POS tagging annotation method is not being fully utilized when used together with our word segmentation annotation method.

We consider that a promising solution to this problem is to extend our word segmentation, POS tagging, and dependency labeling annotation methods to be based on characters and completely abandon the role of the theory of morphology in treebank annotation, which means other than certain exceptions such as named entities, abbreviations, etc., all the syntactic information should be annotated directly on characters. This solution would unify our character-level POS tagging annotation with our treebank annotation approach so that the benefits from both can be reflected in an integrated system.

The difficulty of this proposal is, however, as we have described in Chapter 2, the internal structure of disyllabic words in some cases can be very hard to capture, since many of these words are formed in or derived from classical Chinese and the relation between characters in these words do not necessarily conform to the syntax of contemporary Chinese. We will continue this study in our future work.

References

Giuseppe Attardi. 2006. Experiments with a Multilanguage Non-projective Dependency Parser. In Proceedings of CoNLL, pages 166–170.

Kenji Abe, Shinji Kawasoe, Tatsuya Asai, Hiroki Arimura, and Setsuo Arikawa. 2002. Optimized Substructure Discovery for Semi-structured Data. In Proceedings of PKDD 2002, pages 1–14.

Rakesh Agrawal and Ramakrishnan Srikant. 1994. Fast Algorithms for Mining Association Rules. In Proceedings of 1994 Int. Conf. Very Large Data Bases, pages 487–499.

James F. Allen. 1983. Maintaining Knowledge about Temporal Intervals. Communications of the ACM, 26(11), pages 832-843.

Tatsuya Asai, Hiroki Arimura, Takeaki Uno, and Shin-ichi Nakano. 2003. Discovering Frequent Substructures in Large Unordered Trees. In Proceedings of the 6th International Conference on Discovery Science, pages 47–61.

Masayuki Asahara. 2003. Corpus-based Japanese Morphological Analysis. Nara Institute of Science and Technology, Doctor's Thesis.

Masayuki Asahara, Kenta Fukuoka, Ai Azuma, Chooi-Ling Goh, Yotaro Watanabe, Yuji Matsumoto, and Takashi Tsuzuki. 2005. Combination of Machine Learning Methods for Optimum Chinese Word Segmentation. In Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing, pages 134–137.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine N-best Parsing and MaxEnt Discriminative Reranking. In Proceedings of ACL 2005.

Aitao Chen, Yiping Zhou, Anne Zhang, and Gordon Sun. 2005. Unigram language model for Chinese word segmentation. In Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing, pages 138–141.

Keh-Jiann Chen and Wei-Yun Ma. 2002. Unknown Word Extraction for Chinese Documents. In Proceedings of COLING 2002, pages 1–7.

Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009. Improving Dependency Parsing with Subtrees from Auto-Parsed Data. In Proceedings of EMNLP2009, pages 570–579.

Wenliang Chen, Min Zhang, and Yue Zhang. 2013. Semi-Supervised Feature Transformation for Dependency Parsing. In Proceedings of EMNLP2013, pages 1303-1313.

Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2012. Exploiting Subtrees in Auto-Parsed Data to Improve Dependency Parsing. Computational Intelligence, 28(3), pages 426–451.

Xiangyu Duan, Jun Zhao, and Bo Xu. 2007. Probabilistic Models for Action-based Chinese Dependency Parsing. In Proceedings of ECML/ECPPKDD.

Michael Collins. 2000. Discriminative Reranking for Natural Language Parsing. In Proceedings of ICML 2000.

Michael Collins. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In Proceedings of EMNLP, pages 1–8.

HuiMing Duan, XiaoJing Bai, BaoBao Chan, and ShiWen Yu. 2003. Chinese word segmentation at Peking University. In Proceedings of the second SIGHAN workshop on Chinese language processing, pages 152-155.

Thomas Emerson. 2005. The Second International Chinese Word Segmentation Bakeoff. In Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing, pages 123–133.

Haodi Feng, Kang Chen, Xiaotie Deng, and Weimin Zheng. 2004. Accessor Variety Criteria for Chinese Word Extraction. Computational Linguistics, 30(1), pages 75–93.

Johannes Fischer, Volker Heun, and Stefan Kramer. 2005. Fast Frequent String Mining Using Suffix Arrays. In Proceedings of ICDM 2005, IEEE Computer Society, pages 609–612.

Chooi-Ling Goh, Masayuki Asahara, and Yuji Matsumoto. 2003. Chinese Unknown Word Identification Using Character-based Tagging and Chunking. In Proceedings of ACL 2003.

Chooi-Ling Goh, Masayuki Asahara, and Yuji Matsumoto. 2005. Training Multi-classifiers for Chinese Unknown Word Detection. Journal of Chinese Language and Computing, 15(1), pages 1-12.

Keith Hall. 2007. K-best Spanning Tree Parsing. In Proceedings of ACL 2007.

Katsuhiko Hayashi, Taro Watanabe, Masayuki Asahara, and Yuji Matsumoto. 2011. Third-order Variational Reranking on Packed-Shared Dependency Forests. In Proceedings of EMNLP 2011, pages 1479–1488.

Liang Huang. 2008. Forest reranking: Discriminative Parsing with Non-local Features. In Proceedings of ACL 2008, pages 586–594.

Liang Huang and David Chiang. 2005. Better K-best Parsing. In Proceedings of IWPT 2005, pages 53–64.

ChuRen Huang, KehJiann Chen, FengYi Chen, and LiLi Chang. 1997. Segmentation Standard for Chinese Natural Language Processing. Computational Linguistics and Chinese Language Processing vol. 2, no. 2, August 1997, pages 47-62.

Mike Tian-Jian Jiang, Shih-Hung Liu, Cheng-Lung Sung, and Wen-Lian Hsu. 2010. Term Contributed Boundary Feature using Conditional Random Fields for Chinese Word Segmentation Task. In ROCLING.

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.

Wenbin Jiang, Haitao Mi, and Qun Liu. 2008b. Word Lattice Reranking for Chinese Word Segmentation and Part-of-speech Tagging. In Proceedings of COLING.

Zhihui Jin and Kumiko Tanaka-Ishii. 2006. Unsupervised Segmentation of Chinese Text by Use of Branching Entropy. In Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions, pages 428-435.

Daisuke Kawahara and Sadao Kurohashi. 2006. A Fully-Lexicalized Probabilistic Model for Japanese Syntactic and Case Structure Analysis. In Proceedings of the Human Language Technology Conference of the NAACL, pages 176–183.

Daisuke Kawahara and Sadao Kurohashi. 2006. Case Frame Compilation from the Web Using High performance Computing. In Proceedings of LREC 2006.

Philipp Koehn. 2004. Statistical Significance Tests for Machine Translation Evaluation. In Proceedings of EMNLP 2004, pages 388-395.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion, Demo and Poster Session, pages 177-180.

Terry Koo and Michael Collins. 2010. Efficient Third-order Dependency Parsers. In Proceedings of ACL 2010, pages 1–11.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple Semi-supervised Dependency Parsing. In Proceedings of ACL 2008, pages 595–603.

Canasai Kruengkrai, Kiyotaka Uchimoto, Jun'ichi Kazama, YiouWang, Kentaro Torisawa, and Hitoshi Isahara. 2009. An Error-Driven Word-Character Hybird Model for Joint Chinese Word Segmentation and POS Tagging. In Proceedings of ACL-IJCNLP, pages 513-521.

Canasai Kruengkrai Kiyotaka Uchimoto, Jun'ichi Kazama, Yiou Wang, Kentaro Torisawa, and Hitoshi Isahara. 2009. Joint Chinese Word Segmentation and POS Tagging Using an Error-Driven Word-Character Hybrid Model. IEICE transactions on information and systems, 92(12), pages 2298-2305.

Roland Kuhn and Renato De Mori. 1990. A Cache-based Natural Language Model for Speech Recognition. IEEE Transaction on Pattern Analysis and Machine Intelligence, 12(6), pages 570-583.

Sadao Kurohashi, Toshihisa Nakamura, Yuji Matsumoto, and Nagao Makoto. 1994. Improvements of Japanese Morphological Analyzer JUMAN. In Proceedings of the International Workshop on Sharable Natural Language, pages 22-28.

Zhongguo Li and Maosong Sun. 2009. Punctuation as Implicit Annotations for Chinese Word Segmentation. Computational Linguistics, 35(4), pages 505-512.

Zhongguo Li. 2011. Parsing the Internal Structure of Words: A New Paradigm for Chinese Word Segmentation. In Proceedings of ACL-HLT, pages 1405–1414.

Zhongguo Li and Guodong Zhou. 2012. Unified Dependency Parsing of Chinese Morphological and Syntactic Structures. In Proceedings of EMNLP, pages 1445–1454.

Yih-Jeng Lin and Ming-Shing Yu. 2001. Extracting Chinese Frequent Strings without a Dictionary from a Chinese Corpus and its Applications. Journal of Information Science and Engineering, 17, pages 805-824.

Xueqiang Lü, Le Zhang, and Junfeng Hu. 2004. Statistical substring reduction in linear time. In Proceeding of IJCNLP-2004, pages 320--327.

Wei-Yun Ma and Keh-Jiann Chen. 2003. A Bottom-up Merging Algorithm for Chinese Unknown Word Extraction. In Proceedings of the second SIGHAN workshop on Chinese language Processing.

André FT Martins, Noah A. Smith, Eric P. Xing, Pedro MQ Aguiar, and Mário AT Figueiredo. 2010. Turbo Parsers: Dependency Parsing by Approximate Variational Inference. In Proceedings of EMNLP 2010, pages 34–44.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online Large-Margin Training of Dependency Parsers. In Proceedings of ACL 2005, pages 91–98.

Ryan McDonald and Fernando Pereira. 2006. Online Learning of Approximate Dependency Parsing Algorithms. In Proceedings of EACL 2006, pages 81–88.

Makoto Nagao and Shinsuke Mori. 1994. A New Method of N-gram Statistics for Large Number of N and Automatic Extraction of Words and Phrases from Large Text Data of Japanese. In Proceedings of COLING 1994, pages 611–615.

Tetsuji Nakagawa. 2004. Chinese and Japanese Word Segmentation Using Word-level and Character-level Information. In Proceedings of COLING 2004, pages 466–472.

Tetsuji Nakagawa and Kiyotaka Uchimoto. 2007. Hybrid Approach to Word Segmentation and Pos Tagging. In Proceedings of ACL 2007 Demo and Poster Sessions, pages 217-220.

Mark Nelson. 1996. Fast String Searching with Suffix Trees. Dr.Dobb's Journal.

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 Proceedings of EMNLP, pages 277–284.

Siegfried Nijssen and Joost N. Kok. 2003. Efficient Discovery of Frequent Unordered Trees. In Proceedings of the 1st International Workshop on Mining Graphs,Trees and Sequences, pages 55–64.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2004. Memory-based Dependency Parsing. In Proceedings of CoNLL, pages 49–56.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, pages 160-167.

Adwait Ratnaparkhi. 1996. A Maximum Entropy Model for Part-Of-Speech Tagging. In Proceedings of EMNLP 1996, pages 133–142.

Xiaona Ren, Xiao Chen, and Chunyu Kit. 2013. Combine Constituent and Dependency Parsing via Reranking. In Proceedings of IJCAI 2013, pages 2155–2161.

John Richardson, Raj Dabre, Chenhui Chu, Fabien Cromières, Toshiaki Nakazawa, and Sadao Kurohashi. 2015. KyotoEBMT System Description for the 2nd Workshop on Asian Translation. In Proceedings of the 2nd Workshop on Asian Translation, pages 54-60.

Joseph Le Roux, Benoit Favre, Alexis Nasr, and Seyed Abolghasem Mirroshandel. 2012. Generative Constituent Parsing and Discriminative Dependency Reranking: Experiments on English and French. In Proceedings of the ACL 2012 Joint Workshop on Statistical Parsing and Semantic Processing of Morphologically Rich Languages, pages 89–99.

Federico Sangati, Willem Zuidema, and Rens Bod. 2009. A Generative Re-ranking Model for Dependency Parsing. In Proceedings of IWPT 2009, pages 238–241.

Mo Shen, Daisuke Kawahara, and Sadao Kurohashi. 2012. A Reranking Approach for Dependency Parsing with Variable-sized Subtree Features. In Proceedings of 26th Pacific Asia Conference on Language Information and Computing, pages 308-317.

Mo Shen, Daisuke Kawahara, and Sadao Kurohashi. 2013. Dependency Parse Reranking Based on Subtree Extraction. In Proceedings of the 19st Annual Meeting of the Association for Natural Language Processing, pages 58–61.

Mo Shen, Hongxiao Liu, Daisuke Kawahara, and Sadao Kurohashi. 2014. Chinese Morphological Analysis with Character-level POS Tagging. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Short Papers), pages 253–258.

Richard Sproat and Thomas Emerson. 2003. The First International Chinese Word Segmentation Bakeoff. In Proceedings of the Second SIGHAN Workshop on Chinese Language Processing.

Weiwei Sun. 2010. Word-based and Character-based Word Segmentation Models: Comparison and Combination. In Proceedings of COLING Poster Sessions, pages 1211–1219.

Weiwei Sun. 2011. A Stacked Sub-word Model for Joint Chinese Word Segmentation and Part-of-speech Tagging. In Proceedings of ACL-HLT, pages 1385–1394.

Weiwei Sun and Jia Xu. 2011. Enhancing Chinese Word Segmentation Using Unlabeled Data. In Proceedings of EMNLP 2011, pages 970–979.

Jun Suzuki, Hideki Isozaki, Xavier Carreras, and Michael Collins. 2009. An Empirical Study of Semi-supervised Structured Conditional Models for Dependency Parsing. In Proceedings of EMNLP 2009, pages 551–560.

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A Conditional Random Field Word Segmenter for SIGHAN Bakeoff 2005. In Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing, pages 168–171.

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 IJCNLP 2011.

Fie Xia. 2000. The Segmentation Guidelines for the Penn Chinese Treebank (3.0). http://www.cis.upenn.edu/~chinese/segguide.3rd.ch.pdf.

Nianwen Xue. 2003. Chinese Word Segmentation as Character Tagging. In International Journal of Computational Linguistics and Chinese Language Processing.

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.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical Dependency Analysis with Support Vector Machines. In Proceedings of IWPT 2003, pages 195–206.

Mikio Yamamoto and Kenneth W. Church. 2001. Using Suffix Arrays to Compute Term Frequency and Document Frequency for All Substrings in a Corpus. Computational Linguistics, 27(1), pages 1-30.

Ting-hao Yang, Tian-Jian Jiang, Chan-hung Kuo, Richard Tzong-han Tsai, and Wen-lian Hsu. 2011. Unsupervised Overlapping Feature Selection for Conditional Random Fields Learning in Chinese Word Segmentation. In Proceedings of the 23rd Conference on Computational Linguistics and Speech Processing, pages 109-122.

Yunming Ye, Qingyao Wu, Yan Li, K.P. Chow, Lucas C.K. Hui, and S.M. Yiu. 2013. Unknown Chinese Word Extraction Based on Variety of Overlapping Strings. Information Processing & Management, 49(2), pages 497–512.

Kun Yu, Daisuke Kawahara, and Sadao Kurohashi. 2008. Chinese Dependency Parsing with Large Scale Automatically Constructed Case Structures. In Proceedings of COLING 2008, pages 1049–1056.

Mohammed J. Zaki. 2002. Efficiently Mining Frequent Trees in a Forest. In Proceedings of SIGKDD 2002, pages 71–80.

Xiaodong Zeng, Derek F. Wong, Lidia S. Chao, and Isabel Trancoso. 2013. Co-regularizing Character-based and Word-based Models for Semi-supervised Chinese Word Segmentation. In Proceedings of ACL, pages 171-176.

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2013. Chinese Parsing Exploiting Characters. In Proceedings of ACL, page 125-134.

Yue Zhang and Stephen Clark. 2010. A Fast Decoder for Joint Word Segmentation and POS-tagging Using a Single Discriminative Model. In Proceedings of EMNLP, pages 843–852.

Yue Zhang and Stephen Clark. 2008. A Tale of Two Parsers: Investigating and Combining Graph-based and Transition-based Dependency Parsing. In Proceedings of EMNLP 2008, pages 562–571.

Yue Zhang and Joakim Nivre. 2011. Transition-based Dependency Parsing with Rich Non-local Features. In Proceedings of ACL 2011, pages 188–193.

Ruiqiang Zhang, Genichiro Kikui, and Eiichiro Sumita. 2006. Subword-based Tagging for Confidence Dependent Chinese Word Segmentation. In COLING/ACL 2006, pages 961–968.

Kevin Zhang, Qun Liu, Hao Zhang, and Xue-Qi Cheng. 2002. Automatic Recognition of Chinese Unknown Words Based on Roles Tagging. In Proceedings of the first SIGHAN workshop on Chinese language processing.

Kaixu Zhang, Ruining Wang, Ping Xue, and Maosong Sun. 2011. Extract Chinese Unknown Words from a Large-scale Corpus Using Morphological and Distributional Evidences. In Proceedings of IJCNLP 2011, pages 837-845.

Hai Zhao, Chang-Ning Huang, Mu Li, and Bao-Liang Lu. 2006. Effective Tag Set Selection in Chinese Word Segmentation via Conditional Random Field Modeling. In Proceedings of PACLIC 20, pages 87-94.

Hai Zhao and Chunyu Kit. 2007. Incorporating Global Information into Supervised Learning for Chinese Word Segmentation. In Proceedings of PACLING 2007, pages 66-74.

Hai Zhao and Chunyu Kit. 2008. Exploiting Unlabeled Text with Different Unsupervised Segmentation Criteria for Chinese Word Segmentation. Research in Computing Science, Vol. 33, pages 93-104.

Guo-Dong Zhou. 2005. A Chunking Strategy Towards Unknown Word Detection in Chinese Word Segmentation. In Proceedings of IJCNLP 2005, pages 530-541.

List of Publications

Journals

Mo Shen, Chenhui Chu, Daisuke Kawahara and Sadao Kurohashi. 2015. Improve Chinese Word Segmentation Consistency via Treebank Re-annotation. Transactions of the Association for Computational Linguistics. (Submitted)

Mo Shen, Daisuke Kawahara and Sadao Kurohashi. 2015. Chinese Word Segmentation and Unknown Word Extraction by Mining Maximized Substring. Journal of Natural Language Processing. (To appear)

Mo Shen, Daisuke Kawahara and Sadao Kurohashi. 2014. Dependency Parse Reranking with Rich Subtree Features. IEEE Transactions on Audio, Speech, and Language Processing, 22(7), pp. 1208-1218.

International Conferences

Mo Shen, Hongxiao Liu, Daisuke Kawahara, and Sadao Kurohashi. 2014. Chinese Morphological Analysis with Character-level POS Tagging. In proceedings of the 52th Annual Meeting of the Association for Computational Linguistics (ACL 2014), pp. 253–258, Baltimore, USA.

Mo Shen, Daisuke Kawahara, and Sadao Kurohashi. 2013. Chinese Word Segmentation by Mining Maximized Substrings. In proceedings of the 6th International Joint Conference on Natural Language Processing (IJCNLP 2013), pp.171-179, Nagoya, Japan.

Mo Shen, Daisuke Kawahara and Sadao Kurohashi. 2012. A Reranking Approach for Dependency Parsing with Variable-sized Subtree Features. In proceedings of 26th Pacific Asia Conference on Language Information and Computing (PACLIC 26), pp.308-317, Bali, Indonesia.

Domestic Conferences

Mo Shen, Daisuke Kawahara and Sadao Kurohashi. 2014. Chinese Unknown Word Extraction by Mining Maximized Substrings. In proceedings of the 20th Annual Meeting of the Association for Computational Linguistics (NLP 2014), pp.384-387, Sapporo, Japan.

Mo Shen, Daisuke Kawahara and Sadao Kurohashi. 2013. Dependency Parse Reranking Based-on Subtree Extraction. In proceedings of the 19th Annual Meeting of the Association for Computational Linguistics (NLP 2013), pp.58-61, Nagoya, Japan.