An Investigation on Chinese Noun Phrase Extraction

陳貫中
CHAN Kun-Chung Timothy

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


香港中文大學
圖書館系統
28 JUN 2001
UNIVERSITY
LIBRARY SYSTEM
THE CHINESE UNIVERSITY OF HONG KONG

Abstract

Noun phrases are commonly used for generating index terms for information retrieval systems. Therefore, we need an effective noun phrase extraction approach. In this thesis, we develop an approach to extract maximal noun phrases from Chinese text. Although previous studies have been proposed to extract noun phrases, most of them are only applicable to Western languages. To the best of our knowledge, very few handle Chinese text. Many existing approaches for Western languages made use of statistical methods. However, due to the complicated structure of maximal Chinese noun phrase, pure statistical approaches may not be effective.

We attempt to improve the performance of a statistical method by integrating it with the transformation-based error-driven learning (TEL) technique. Our methodology includes two stages. The first stage applies a statistical method to extract Chinese noun phrases. The performance of this approach, in terms of precision and recall, is investigated. The second stage

applies the TEL algorithm to further refine the output of the first stage. The TEL algorithm automatically learns a set of transformation rules to fix the errors that are obtained through comparing the output of the first stage with the correctly annotated corpus. The learned rules can be applied to an unseen corpus one by one to correct the errors. The TEL algorithm is shown to be effective in removing incorrect noun phrases.

摘要

在信息檢索系統中，名詞短詞常用於生成索引詞。因此一個有效的名詞短語抽取系統，在信息檢索的研究中是必要的。在這篇論文中，我們嘗試抽取中文文章中的最大名詞短語。過去已有不少研究探討如何從自然語言中抽取名詞短語，不過，大多數的研究只適用於西方語言，很少能夠處理中文。此外，多數的研究採用統計方法，但是，由於最大中文名詞短語的結構十分複雜，單純統計方法未能有效地將之處理。

本文嘗試集成統計方法及錯誤驅動轉換方法以提高統計方法的表現。我們首先以統計方法抽取中文名詞短語，探討這方法在精確率及召回率上的表現，然後再採用錯誤驅動轉換方法去改善統計方法的結果。透過比較統計方法的結果及正確標註的語料庫，錯誤驅動轉換方法能夠學習一套轉換規則，把錯誤抽取的名詞短語糾正。這些轉換規則更可應用在其他語料庫上，實驗結果証明錯誤驅動轉換方法能有效地把錯誤標註的名詞短語刪除。

Acknowledgments

I would like to express my gratitude to my supervisor Prof. Chun-Hung Cheng as well as Prof. Kam-Fai Wong for all the guidance they have given me throughout the research. Special Thanks are due to Dr. Wen-Jie Li who has given me many invaluable advises.

Next, I would like to thank all my lovely colleagues in the department. Ah Kin, Wai Ip, Silvia and Timmy, deserve a great deal of thanks for their constant support and encouragement. Without their help, I may not be able to finish this thesis. Thanks are also due to Ah Su, Ada Luk, all guys in the IS Lab, HCCL and Room 224. Last but not the least, I want to express my sincere thank to my parents and my sister.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The research of information retrieval (IR) has started for decades and many sophisticated techniques have been proposed. However, most of the work only concentrated on Western languages. Few focused on Asian languages. Unlike Western languages, Chinese does not have a well-defined grammar. This makes it difficult to adopt grammar rules in information extraction from Chinese text. Also, compared with those of Western languages, the sentence structure of Chinese is more complex. This makes it difficult to develop a IR application for Chinese.

In a natural language, noun phrases are basic building blocks for sentences. Useful entities and concepts are commonly represented as noun

phrases. Therefore, by identifying noun phrases, we can capture the critical information of a text.

Noun phrase extraction is usually served as a preliminary step for natural language applications. For example, in parsing, analyzing the pattern of a sentence, especially a long one, is a complicated task. In order to reduce the complexity, noun phrases are identified and assigned to the "noun phrase" category such that the details in the noun phrases are ignored. In this way, every noun phrase is treated as a single unit and the resultant pattern of the sentence is simplified. This in turn simplifies the structure of the parse tree.

Furthermore, a noun phrase extraction system is also useful as it can provide researchers with a better means to conduct quantitative studies on lexical terms generation and technical terminologies extraction. Thus, an effective noun phrase extraction system is essential in information retrieval research.

One approach of noun phrase extraction is to parse a sentence. Noun phrases are then extracted from the nodes of the parse tree with noun phrase label. This approach requires sufficient linguistic knowledge and the procedure is complicated. Greater difficulty is encountered in parsing Chinese sentence because of its complicated, recursive and ambiguous structure. A more popular approach for noun phrase extraction is statistics-based. This approach is simple as little linguistic knowledge is needed and this has been

shown effective. Recently, hybrid approaches, i.e. statistics-based and rule-based combined methods, are becoming popular. Researchers are trying to build a system that contains the merits of both approaches. However, the design of rules often requires sufficient linguistic knowledge and is only applicable to a specific domain.

In this thesis, a maximal Chinese noun phrase extraction system is proposed. The system consists of two modules. The first one is a statistical noun phrase extraction module which conducts a preliminary identification of noun phrases. By analyzing the output of this module, we observe that the statistical algorithm applied in our module is insufficient to recognize some kind of noun phrases. Therefore, a transformation-based error correction model is developed to refine the output of the preliminary noun phrase extraction module automatically.

1.2 Outline of Thesis

The rest of this thesis is organized as follows: Chapter 2 reviews some related works and their performance is evaluated. In Chapter 3, some background information for noun phrase extraction is provided. This is followed by an overview of our Chinese noun phrase extraction system. Chapter 4 describes the first part of the system, which is a statistical noun phrase extraction

module. The main component of the module is the pairing of candidate boundaries. Different approaches for pairing are evaluated. The dynamic programming based approach is adopted and is modified to enhance the extraction performance. The second part of our system is outlined in Chapter 5. The transformation-based error-driven learning (TEL) algorithm is discussed and applied to our system to refine the output of the extraction module introduced in the previous chapter. Chapter 6 concludes the thesis and provides some directions for future research.

Chapter 2

Background

In this chapter, we first describe the structure of Chinese noun phrase briefly. Then we give a review on the previous works on noun phrase extraction and outline their pros and cons.

2.1 Chinese Noun Phrase Structure

Researchers are interested in two kinds of noun phrases. They are minimal and maximal noun phrase. There is no standard definition on minimal noun phrase. In this thesis, minimal noun phrase can be a nominal noun (e.g. 公司), a numeral phrase (e.g. 百分之二點六), a measure phrase (e.g. 一億美元), a locative noun phrase (e.g. 門後) or a temporal noun phrase (e.g. 一個月) etc. It may also consist of a modifier and a nominal head.

For maximal noun phrase, we define it as the longest noun phrase that can be found in a sentence. Therefore, it can be (i) a simple noun phrase that described before; (ii) a coordinate noun phrase that made up of two or more parts which are usually linked with conjunction words; (iii) an appositive noun phrase which is a combination of two consecutive noun phrases in which one modify the other and; (iv) a subordinate noun phrase that consists of an attribute[1] as its modifier and a nominal head as its central part.

The structure of minimal noun phrase is simple and has little variation, and therefore it is easier to extract. For maximal noun phrase, although the complicate structure increases the difficulty of extraction, more useful information is included in the noun phrase and it is worth doing. Therefore, in our work, we will focus on maximal noun phrase extraction.

2.2 Literature Review

Church (1988) [16] proposed a part-of-speech tagger and a noun phrase extractor for non-recursive English noun phrase extraction. The extractor identified the noun phrases according to two probability matrices: starting noun phrase matrix and ending noun phrase matrix. Out of 243 noun phrases, only five were missing. However, the size of the testing corpus was too small to

[1] The attribute can be a verb phrase, a noun phrase, an adjective phrase, a prepositional phrase or a sentence.

justify the performance of the system. In his later work (1992) [17], he tried to identify noun phrases containing sequences of determiners, premodifiers and nominal heads.

Bourigault's LECTER (1992) [5] was a surface-syntactic analyzer designed to extract maximal noun phrases (mainly sequences of determiners, premodifiers, nominal heads, and certain kinds of post-modifying prepositional phrases and adjectives) from French texts. LECTER worked in two stages: *analysis* stage that extracted maximal noun phrases and, *parsing* stage that parsed the maximal noun phrases to obtain the terminology embedded inside. Based on a testing corpus containing 46,000 noun phrases, the recall of the system was 95%. This result was validated by human expert.

Voutilainen (1993) [35] proposed a maximal noun phrase extraction system known as NPtool. The input text was subjected to a preprocessor which provided each word with tags indicating part-of-speech, inflection, derivation and syntactic functions. The text was then parsed according to the assigned tags. Afterwards, NPtool applied two finite state mechanisms, namely NP-hostile and NP-friendly, to solving parsing ambiguities. This two mechanisms produced two sets of noun phrases. The final result was formed by the union of the 2 sets. The recall of NPtool was 98.5–100% and the precision 95–98%. These figures were validated manually by around 20,000 words. However, Chen and Chen (1994) [15] pointed out that the recall was only about 85%

for some inconsistencies were found among those extracted noun phrases.

Chen and Chen (1994) [15] proposed an English NP extraction system, namely NP-TRACTOR. which applied statistical approach and linguistic knowledge to extract three kinds of noun phrases (maximal noun phrase (MNP), minimal noun phrase (mNP), and maximal and also minimal noun phrase (MmNP)) from the SUSANNE Corpus [30, 31]. The input text was first tagged with part-of-speech tags. A probabilistic partial parser with dynamic programming was then used to partition the tagged text into chunks. Syntactic and semantic heads were then assigned to each chunk. The plausible noun phrases were then connected and extracted according to the information of the syntactic heads and semantic heads using a finite state mechanism. Only 7 states were involved in the mechanism. The average precision was 95% and the recall for MNP, mNP and MmNP were 70%, 72% and 95%, respectively.

Schiller (1996) [33] reported a multi-lingual (Dutch, English, French, German, Italian, Portuguese and Spanish) noun phrase extraction tool for extraction of non-recursive noun phrases from technical manuals. Raw input text was first divided into tokens by a tokenizer. The output was further processed by a HMM-tagger. The tagger required a finite-state lexicon, which contained part-of-speech and morphological tags. After part-of-speech tagging, noun phrases were extracted by a finite-state mechanism. The tagger's

average accuracy was 97–98%. In an open test on a small corpus (145 sentences) of a technical manual, the accuracy for noun phrase extraction was about 98%.

Li (1997) [23] proposed a statistical-based maximal Chinese noun phrase extraction system. The system operated in two stages. In the first stage, candidate noun phrase boundaries were determined from statistical data. The noun phrase boundaries were then paired up with a heuristic-based model. The second stage was designed to resolve the structural ambiguities due to relative clauses and prepositional phrase modifiers. Semantic relations between word pairs such as verb-noun and preposition-noun were studied. Through the study, rules for resolving structural ambiguities were determined. These rules were applied to the text and the performance of noun phrase extraction was improved. The overall recall and precision were 93.4% and 91.9%, respectively.

Skut and Brants (1998) [34] described a Chunk Tagger which used a stochastic approach to recognize syntactic structure of a sentence of limited depth. The author defined a structural tag which contained 4 kinds of information: structural relation, part-of-speech, phrasal categories and information about the category of the grandparent node. The structural tag was denoted as $S_i =< r_i, t_i, c_i, g_i >$. For a sequence of part-of-speech tags, the tagger assigned a sequence of appropriate structural tags. After tagging,

one could figure out the NPs by examining the structural tags. The output, however, was created interactively, i.e., the boundary of a complex noun phrases or prepositional phrases were specified manually, and the tagger determined its category and internal structure. The training corpus for this application was created by extracting all noun phrases, prepositional phrases and adjective phrases occurring in the NEGRA treebank. The precision of this application was 94.30%.

Meanwhile Cardie and Pierce [14] presented a minimal noun phrase extraction system. In the training stage, the system extracted sequences of part-of-speech tags from each minimal noun phrase to form a set of grammar rules. Duplicate rules were removed. To select the rules for accurate identification of minimal noun phrases, they proposed an error-driven pruning procedure for the task. After the pruning procedure, the resulting rules could then be applied to the tag sequences of the testing corpus to identify minimal noun phrases. Ambiguities were solved by choosing the longest matching rule. The precision and recall of the system were 89.4% and 90.9% respectively. They then introduced some local repair heuristics to further improve the results by around 1%.

2.3 Observations

The following are observations made from the aforesaid research work:

1. Except Li's [23] work, most of the systems deal with simple noun phrases, i.e. those that are not combined with post-modifiers, such as prepositional phrases and relative clauses. The structure of a simple noun phrase does not vary so much and does not involve structural ambiguities. Therefore, they can be handled easily with either statistical or rule-based approaches.

2. Information other than part-of-speech tags, e.g. semantic class, grammar rules, structural tags, etc, are often required to aid extraction. However, in real situations, one may not be able to obtain so much information.

3. The linguistic rules used in the systems are tailor made to a specific language (mainly western languages) and they are defined manually. This increases the difficulty in applying the algorithm to other domains or languages as the user must have sufficient linguistic knowledge of the languages and spend a considerate effort to find out the rules.

2.4 Chapter Summary

In the past ten years, a number of researches have been done on noun phrase extraction. Their algorithms are reviewed in this chapter. Their results are found to be impressive. However, most studies are targeted for simple noun phrases. Moreover, little efforts have been made in noun phrase extraction for Asian language.

The common approaches for noun phrase extraction are statistical-based and rule-based approaches. Statistical-based approach is simple, effective and hence it is popular. Rule-based approach allows the user to apply linguistic knowledge to the system. However, the user must have sufficient linguistic knowledge and the linguistic rules are defined manually. Also, the rules are usually domain specific. In our work, we aim at building a maximal Chinese NP extraction system that requires little kinds of information, combines the advantages of statistical-based and rule-based approach and, the most important thing is, the linguistic rules can be learned automatically. In the next chapter, we outline our maximal Chinese noun phrase extraction system.

Chapter 3

Maximal Chinese Noun Phrase Extraction System

This chapter first introduces some background information on noun phrase extraction. This is followed by an overview of our Chinese noun phrase extraction system.

3.1 Background

3.1.1 Part-of-speech Tagset

A Part-of-speech (POS) tag describes the grammatical function of a word. It plays an important role in natural language processing (NLP). Further-

more, it is used to form grammar rules to represent the internal structure of a sentence. These rules are essential in applications such as noun phrase extraction, parsing and indexing, etc.

According to Yu [4], modern Chinese words can be classified into 18 basic categories (see Table 3.1). In addition, six categories were later added to improve the language processing performance (see Table 3.2).

The 24 general categories can be further divided into different levels according to applications and accuracy requirements. In the POS tagset provided by Tsinghua and Peking University [1], there are totally 108 POS tags defined based on the aforementioned categories. In Li's [23] thesis, two more tags were added. They are "nvg", which represents gerund and is added under the noun category, and "$", which represents the beginning of a sentence. In our research, the tag "xnull" is added which represents punctuation s including "，", "。", "；", "：", "？" and "！ ". Thus we have a POS tagset of 111 tags. The complete tagset is shown in Appendix A.

3.1.2 The Tagging System

Our corpus was word segmented and part-of-speech tagged based on the Tsinghua University's tagging system, namely TAGGER [1]. TAGGER applies first order statistical Markov model which is commonly used in tagging. The

	POS category		Abbreviation
1	noun	名詞	n
2	time word	時間詞	t
3	place word	處所詞	s
4	position word	方位詞	f
5	verb	動詞	v
6	adjective	形容詞	a
7	state word	狀態詞	z
8	distinguishing word	區別詞	b
9	numeral	數詞	m
10	measure word	量詞	q
11	pronoun	代詞	r
12	preposition	介詞	p
13	adverb	副詞	d
14	conjunction	連詞	c
15	particle	助詞	u
16	modal word	語氣詞	y
17	onomatopoeia	象聲詞	o
18	interjection	嘆詞	e

Table 3.1: 18 basic categories of Chinese POS tagset

	POS category		Abbreviation
19	prefix	前綴	h
20	suffix	後綴	k
21	idiom	成語	i
22	abbreviation	簡稱語	j
23	habitually used word	習用語	l
24	others	其他	x

Table 3.2: Six additional categories of Chinese POS tagset

accuracy of the tagging system in open test is 96.1%. Therefore, we revised the tagged corpus manually in order to correct the errors.

3.1.3 Chinese Corpus

The corpus we use is provided by Tsinghua University. It includes 5 texts selected from Renmin Rebao (人民日報) in different domains: computer science (C), military affairs (J), science and technology (K), news report (N) and information retrieval (S). There are totally 2,909 sentences and 74,556 words. Some basic statistics of the corpus are summarized in Table 3.3.

In the corpus, each word is followed by a POS tag and a "#" mark is inserted between the word and its POS tag. Furthermore, in the training corpus, all the maximal noun phrases was first manually marked by a pair of square brackets (" [" and "] "). An excerpt of the corpus with maximal

Domain	Num of Sentence	Num of Word	Num of NP	Max. NP length
Computer (C)	237	6,203	905	36
Military affairs (J)	598	14,589	2,264	53
Science and technology (K)	651	14,945	2,591	28
News report (N)	981	25,811	4,251	58
Information retrieval (S)	442	13,008	1,770	66
Total	2,909	74,556	11,703	66

Table 3.3: Statistics of the corpus

noun phrases marked is shown in Figure 3.1.

3.1.4 Grammar Rules and Boundary Information

There are two common approaches in noun phrase extraction. One makes use of grammar rules and the other employs noun phrase boundary information. Grammar rules are in fact noun phrase patterns. They can be obtained from linguistic knowledge (i.e. grammar books) or from a training corpus. When a phrase pattern matches a grammar rule, it will be regarded as a noun phrase. For the second approach, noun phrases are recognized by locating its starting and ending points. The boundary information represents the probability that a noun phrase begins or ends at a certain position.

Recognizing noun phrases by grammar rules is effective if the target is

$ 由于#p 涉及#vg 到#vc [U N I X#xch 源#b 程序#ng 向#p 中國#s 出口#vg 以及#cpw 其它#rn 法律#ng 條款#ng 等#x 問題#ng] ， 一直#d 拖延#vgn 了#utl [近#a 兩#mx 年#ng] ， 在#pzai [A T & T#xch 國際#ng 市場#ng 開發#vg 信息#ng 系統#ng 部#ng 副#b 總裁#ng E v a n s#xch 先生#ng 和#cpw A T & T#xch 中國#s 公司#ng 信息#ng 系統#ng 部#ng 許#nf 心仰#npf 先生#ng 的#usde 力#d 促#nvg] 下#f ， [1 9 8 7 年#t 9 月#t 底#t] 由#p [雙方#ng] 簽字#vgo 生效#vgo 。

Figure 3.1: An excerpt of corpus with maximal noun phrases marked

minimal noun phrases. This is because their structure is simple thus, only a limited number of grammar rules are required to represent a large number of noun phrases. This leads to high recall in testing. However this is not true for maximal noun phrase extraction. The complicated and recursive structure of maximal noun phrases makes it difficult to fully represent them by linguistic grammar rules. One way to compensate this is to involve more grammar rules. This can, however, only be achieved using a large amount of training text.

On the other hand, identifying noun phrases using boundary information is easier and more flexible. Many researchers prefer this approach [16, 23, 25, 26] and we adopt it in our maximal Chinese noun phrase extraction system.

3.1.5 Feature Selection

In this thesis, we define *feature* as the information required to determine the position of noun phrase boundary. POS tags around noun phrase boundary are chosen as the features. Complexity of feature space describes the number of features under consideration. In general, higher complexity supports a broader range of context and this leads to better accuracy in the determination of boundary position. On the other hand, if the training data is insufficient, there are fewer training instances in each slot of the feature space and thus leading to data sparseness problem.

3.2 Overview of Our Chinese Noun Phrase Extraction System

Our system consists of two modules: (1) preliminary noun phrase extraction and (2) automatic error correction. Both modules have its own training and testing processes.

3.2.1 Training

In the training stage (see Figure 3.2), statistics of the boundary information of the noun phrase annotated training corpus is learned (by process 1a) and

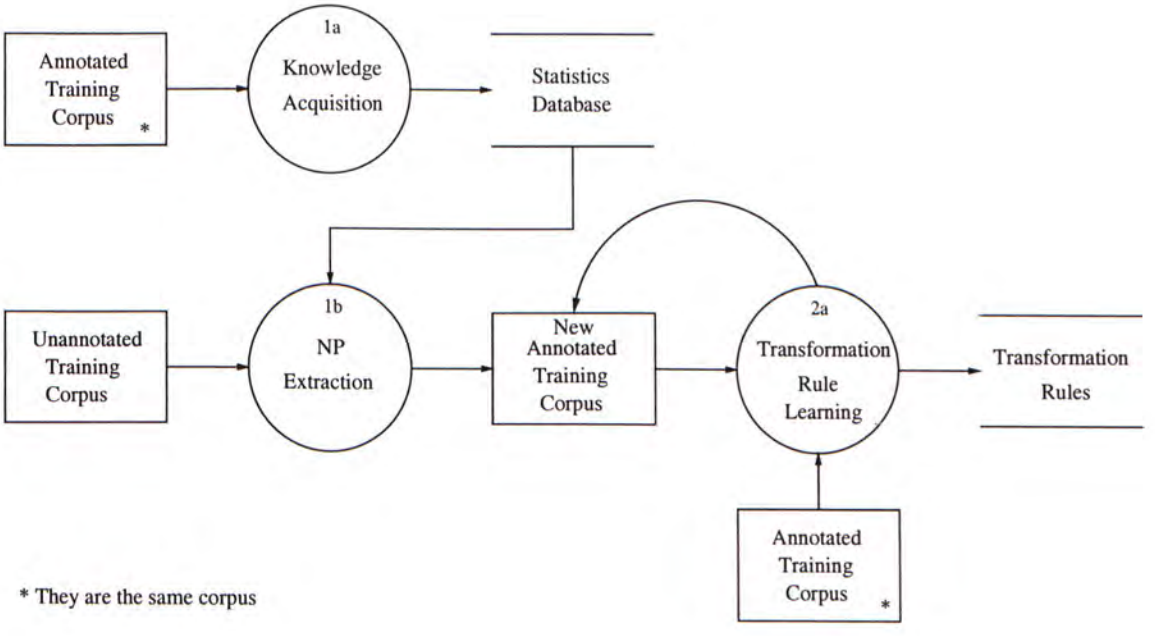

Figure 3.2: Training procedure of Chinese noun phrase extraction system

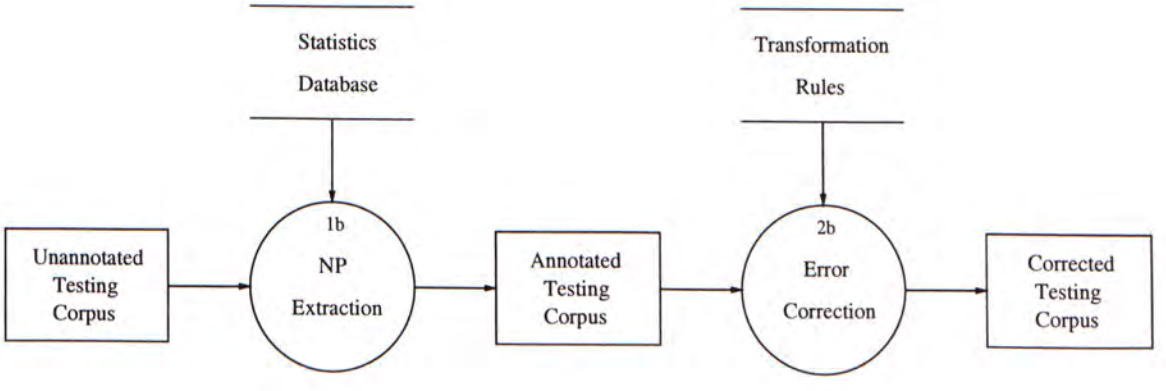

Figure 3.3: Testing procedure of Chinese noun phrase extraction system

stored in the statistics database. An unannotated training corpus is then prepared by eliminating all the noun phrase brackets of the training corpus. It is presented to the noun phrase extraction process (process 1b) in which noun phrase boundaries are marked according to the statistics to form the new annotated corpus. In process 2a, by comparing with the original training corpus, an ordered list of transformation rules are learned to correct the errors found in the newly annotated corpus.

3.2.2 Testing

Figure 3.3 shows the testing procedure. A testing corpus is first annotated by the noun phrase extraction process using the statistics learned in training. The output is passed to the error correction process (process 2b) in which the learned transformation rules are applied to the annotated testing corpus one by one until the list of rules is exhausted.

3.3 Chapter Summary

In this chapter, we have introduced the environment of our noun phrase extraction system, including part-of-speech tagset, the tagging system, and the Chinese corpus. We then briefly describe the algorithm of our Chinese noun phrase extraction system. The system consists of two modules.

The first one is a statistical noun phrase extraction model which conducts a preliminary identification of noun phrases boundaries. The second one is a transformation-based error correction model which automatically learns transformation rules to fix the errors induced due to the statistical extraction model. Details of the two modules are further discussed in Chapter 4 and 5, respectively.

Chapter 4

Preliminary Noun Phrase Extraction

In this chapter, we first depict the framework the preliminary noun phrase extraction model [21]. We then evaluate different approaches for candidate boundary pairing. The dynamic programming-based approach is adopted and revised to enhance the performance.

4.1 Framework

Our preliminary noun phrase extraction model is based on CNPext, a Chinese noun phrase extraction model introduced by Li [23]. The framework of the model is shown in Figure 4.1. In the training stage, a **Training**

Corpus with maximal noun phrase annotated is passed into the module **Boundary Information Acquisition** in which the statistics of boundary information is learned and store in the **Statistics Database**. In the testing stage, a **Testing Corpus** without maximal noun phrase annotated is passed to the **NP Extraction** process which consists of two steps. The first step is **Candidate Boundary Insertion**, where candidate boundaries are inserted between words. Afterwards, the left and right boundaries are paired up in the **Candidate Boundary Pairing** process. During the **NP Extraction** process, the boundary information statistics are used to determine the insertion of boundary and to evaluate the correctness of every paired boundaries, i.e. the extracted maximal noun phrase.

4.2 Boundary Information Acquisition

The main task of the Boundary Information Acquisition module is to determine the probability of a left or a right boundary exists between a pair of tags. Suppose w_i and w_{i+1} are two adjacent words, t_i and t_{i+1} are their corresponding part-of-speech (POS) tags, respectively. In the original CNPext model, the probabilities of the left and right boundary (P_l and P_r, respectively) between t_i and t_{i+1} are defined as:

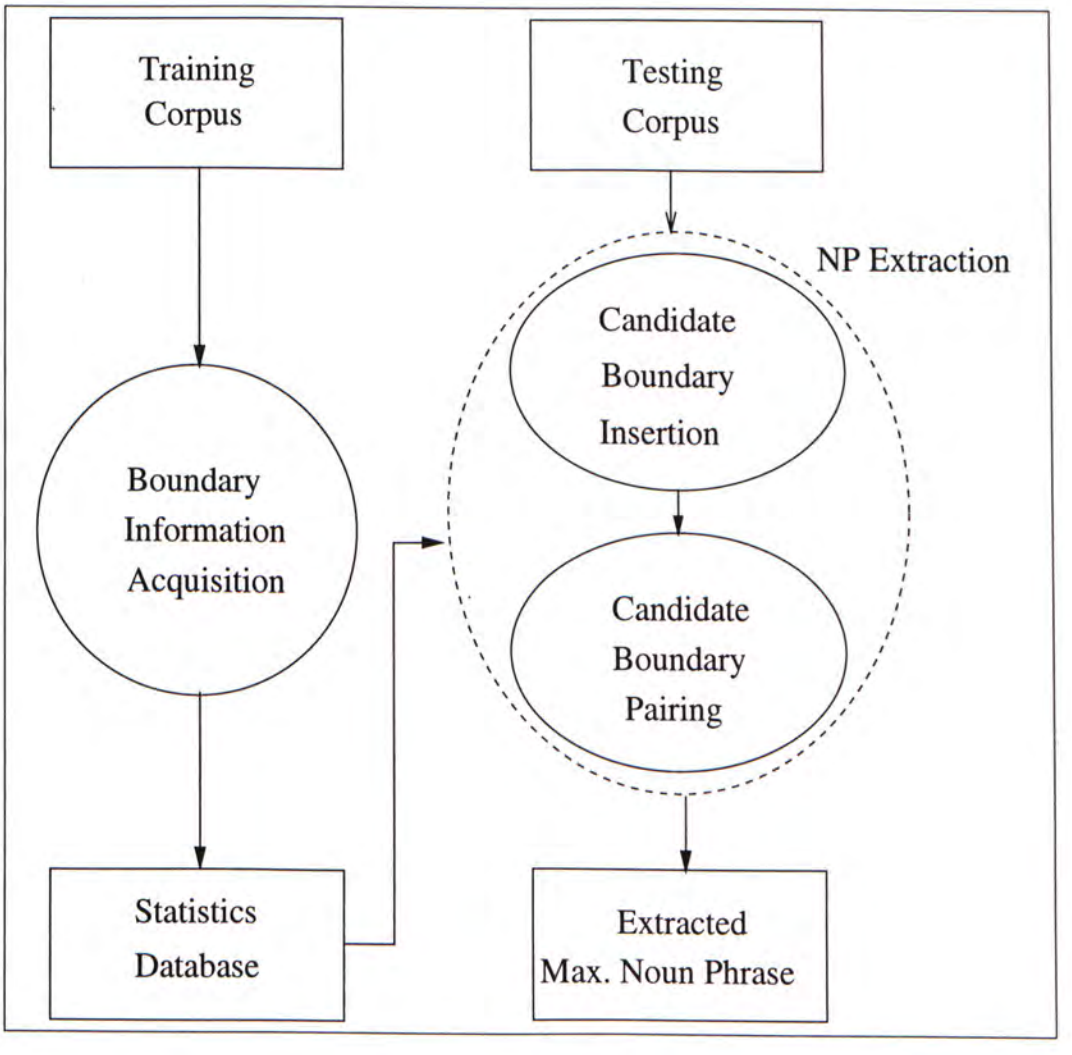

Figure 4.1: Framework of Maximal Chinese Noun Phrase Extraction System

$$P_l(i) = P_l(t_i, t_{i+1}) = \frac{freq(t_i, [\,, t_{i+1})}{freq(t_i, t_{i+1})} \quad (4.1)$$

$$P_r(i) = P_r(t_i, t_{i+1}) = \frac{freq(t_i,]\,, t_{i+1})}{freq(t_i, t_{i+1})} \quad (4.2)$$

where $freq(t_i, [\,, t_{i+1})$ and $freq(t_i,]\,, t_{i+1})$ are the number of times that the left and right boundary exist between t_i and t_{i+1} in the training corpus, respectively. The term $freq(t_i, t_{i+1})$ is the number of times that the tag pair t_i and t_{i+1} appear in the training corpus.

In addition, the probability of no boundary (P_n) and probability of a "] [" (P_b) between a tag pair t_i, t_{i+1} are respectively defined as:

$$P_n(i) = P_n(t_i, t_{i+1}) = (1 - P_l(i))(1 - P_r(i)) \quad (4.3)$$

$$P_b(i) = P_b(t_i, t_{i+1}) = P_l(i)P_r(i) \quad (4.4)$$

4.3 Candidate Boundary Insertion

In this process, for 2 adjacent tags t_i and t_{i+1}, if either P_l, P_r or P_b, exceeds or is equal to a threshold value, the appropriate boundary will be inserted between t_i and t_{i+1}. The inserted boundary is called a candidate boundary. The set of all candidate boundaries inserted in the whole corpus are defined

as:

$$\mathcal{L} = \{l_1, l_2, \ldots, l_m\}$$

$$\mathcal{R} = \{r_1, r_2, \ldots, r_n\}$$

where l_m represents the m^{th} left candidate boundary and r_n represents the n^{th} right candidate boundary.

The set of positions of each left and right candidate boundaries are defined as:

$$pos(\mathcal{L}) = \{pos(l_1), pos(l_2), \ldots, pos(l_m)\}$$

$$pos(\mathcal{R}) = \{pos(r_1), pos(r_2), \ldots, pos(r_n)\}$$

It follows that the set of boundary probabilities are defined as:

$$P_l(\mathcal{L}) = \{P_l(l_1), P_l(l_2), \ldots, P_l(l_m)\}$$

$$P_r(\mathcal{R}) = \{P_r(r_1), P_r(r_2), \ldots, P_r(r_n)\}$$

where $P_l(l_m) = P_l(pos(l_m))$, which represents the probability of a left boundary at $pos(l_m)$.

4.4 Pairing of Candidate Boundaries

After the candidate boundaries have been inserted, the left and right boundaries of each sentence are needed to be paired up to form a sequence of noun

phrases. For instance, in the sentence " [連接#vgn [計算機#ng 的#usde 方式#ng] 都#d 是#vy [由#p [各#rn 生產#vg 廠家#ng] 在#pzai [專用#b 的#usde 基礎#ng] 上#f] 確定#vg 的#y ", a sequence of candidate boundaries was inserted. Since different boundary pairing combinations are possible, more than one noun phrase(s) could be formed. Therefore, we have to determine the correct combination.

In Li's thesis, three mathematical models were proposed for candidate boundary pairing. They are namely *conditional probability-based (CP)*, *heuristic-based* and *dynamic programming-based (DP)* model. Each of them will be discussed briefly in the following [1].

4.4.1 Conditional Probability-based Model

The way that this model calculates the boundary probabilities are different from those mentioned in Equation 4.1 to 4.4. For example, given the following sequence of candidate boundaries:

$$\cdots \quad] \; t_{j-1} \; [\; t_j \; [\; t_{j+1} \;] \; \cdots \; [\; t_i \;] \; t_{i+1} \;] \quad \cdots$$

where t_i represents a part-of-speech tag at position i.

[1]The detail algorithms of *heuristic-based* and *DP-based* model can be found in Appendix B.1 and B.2. For *CP-based* model, please refer to [23] or [24].

The conditional probability of pairing the boundaries at position i and j is defined as:

$$Con\_P(r_i|l_j) = \frac{freq(t_{j-1},\ [\ , t_j, t_{i-1},\ ]\ , t_i)}{freq(t_{j-1},\ [\ , t_j)} \tag{4.5}$$

The conditional probabilities of all possible boundary pairs are calculated and stored in a matrix.

In Equation 4.5, the feature space under consideration is bi-gram, i.e. we only consider two adjacent tags around the noun phrase boundary. For a n-gram model, the size of matrix is obtained by:

$$|T|^n + l \times r \tag{4.6}$$

where $|T|$ is the size of POS tagset, l and r is the number of left and right candidate boundaries respectively. By further calculations basing on the matrix, the best pairs with highest probabilities will be selected.

4.4.2 Heuristic-based Model

The heuristic employed in this model is based on *maximal probability*. The candidate boundaries with the highest probability are chosen. The assumption is that the larger the probability a candidate boundary has, the more

likely it serves as a noun phrase boundary. The selection is described as

$$\begin{cases} l_k^* = \underset{i}{argmax}\ P_l(l_i|r_{k-1}^*) & \text{for all } i, pos(r_{k-1}^*) < pos(l_i) \leq pos(r_k^*) \\ r_k^* = \underset{j}{argmax}\ P_r(r_i|l_k^*) & \text{for all } j, pos(l_k^*) < pos(r_j) \leq pos(l_{k+1}^*) \end{cases} \tag{4.7}$$

where k is the k^{th} paired noun phrase and "*" means it has or will have been retained as the phrase boundary.

For example, given the following sequence of candidate boundaries:

$$\begin{array}{lccccccccccc} \cdots &] & t_{k+1} & [& t_j & [& t_{j+1} &] & \cdots & [& t_i &] \ t_{i+1} \ ] & \cdots \\ \cdots & & & & & \uparrow & & & & & \uparrow & & \\ & & & & & maxP_l & & & & & maxP_r & & \end{array}$$

where t_i represents a part-of-speech tag at position i.

Suppose $maxP_l(j)$ and $maxP_r(i)$ have the maximum probabilities of the left and right boundary among the sentence respectively. According to the *maximal probability* heuristic, they should be paired up to form a noun phrase.

4.4.3 Dynamic Programming-based Model

The objective of DP is to evaluate the probabilities of all possible combinations of left and right boundaries and select the best one. The evaluation is measured by a quantity measure, namely *score*. Take the sentence shown above as an example, the best score of boundary sequence end at position i is as follows:

$$score(i) = score(k) \times P_l(j) \times P_r(i) \times \prod_{l=k+1, l \neq j,i}^{n-1} P_n(l) \qquad (4.8)$$

where:

$score(k)$ is the score of the boundary sequence end at at position k;

$P_l(j)$ is the probability of placing a left boundary between tags t_j and t_{j+1};

$P_r(i)$ is the probability of placing a right boundary between tags t_i and t_{i+1};

$P_n(l)$ is the probability of no boundaries exists between tags t_l and t_{l+1};

$\prod_{l=k+1, l \neq j,i}^{n-1} P_n(l)$ is the probability of no boundaries exists beyond t_{k+1} (except at position j and i);

n is the number of tags in the sentence.

4.4.4 Model Selection

The *DP-based* model is adopted in our system because of the following reasons:

- Since our noun phrase extraction process is statistical-based, the sequence of boundary pairs with highest probability should be chosen as the best one. When calculating the probability of a sequence, we have to consider the probability of having or not having a boundary at each

position of the sentence. This is the way that *DP-based* model evaluates the correctness of a boundary sequence. However, the *heuristic-based* model considers only the probability of candidate boundaries and the information given by P_n were omitted, therefore, this evaluation is incomplete.

- When comparing the *DP-based* model with the *CP-based* model, although more information is considered by the *CP-based* model, too much memory space is required to store the statistical data (see Equation 4.6). This makes the computation infeasible if n is large.

- In the *CP-based* model, since the matrix size for storing the statistical data is so large, more training data is required to avoid data spareness problem.

4.4.5 Revised Dynamic Programming Model

From the probabilities definitions given by Li (Eq. 4.1 to 4.4) , we found that they are over-simplified:

Probability of Left/Right Boundary (P_l and P_r)

In counting the frequency of left boundary in the training corpus, the " [" in "] [" was also counted. That meant the frequency of a left boundary equals

to $freq(t_i, [\,, t_{i+1}) + freq(t_i,]\,[\,, t_{i+1})$. This would over-estimate the number of left boundary in the training corpus. The same situation also happened in counting right boundaries.

Probability of Null Boundary (P_n)

$(1 - P_l(i)) \times (1 - P_r(i))$ were used to calculate the probability of no boundary between t_i and t_{i+1}. However, this was over-simplified. The probability of no boundary between two consecutive tags would be more appropriate as $freq(t_i, null, t_{i+1})/freq(t_i, t_{i+1})$.

Probability of Both Boundary (P_b)

In counting the frequency of "] [", $P_r(i)P_l(i)$ were used. Again, this could be rectified as $freq(t_i,]\,[\,, t_{i+1})/freq(t_i, t_{i+1})$.

Revised Probabilities

To overcome the above problems, the probability definitions are revised. P_N, P_L, P_R, P_B, i.e. the probabilities of having a null, a left only boundary, a right only boundary and a consecutive left and right boundary (i.e. "] [")

between two consecutive tags t_i and t_{i+1}, respectively, are as follows:

$$P_N(i) = P_N(t_i, null, t_{i+1}) = \frac{freq(t_i, null, t_{i+1})}{freq(t_i,\ t_{i+1})} \tag{4.9}$$

$$P_L(i) = P_L(t_i,\ [\ , t_{i+1}) = \frac{freq(t_i,\ [\ , t_{i+1})}{freq(t_i,\ t_{i+1})} \tag{4.10}$$

$$P_R(i) = P_R(t_i,\ ]\ , t_{i+1}) = \frac{freq(t_i,\ ]\ , t_{i+1})}{freq(t_i,\ t_{i+1})} \tag{4.11}$$

$$P_B(i) = P_B(t_i,\ ]\ [\ , t_{i+1}) = \frac{freq(t_i,\ ]\ [\ , t_{i+1})}{freq(t_i,\ t_{i+1})} \tag{4.12}$$

Expressing the original probabilities in terms of the new ones, we have the following relationship:

$$P_n = (1 - (P_L + P_B))(1 - (P_R + P_B)) \tag{4.13}$$

$$P_l = P_L + P_B \tag{4.14}$$

$$P_r = P_R + P_B \tag{4.15}$$

$$P_b = (P_L + P_B)(P_R + P_B) \tag{4.16}$$

In summary, we define the new and old boundary occurrence probability sets between t_i and t_{i+1} as follows:

$$Pset_{new}(t_i, t_{i+1}) = \{P_N(i), P_L(i), P_R(i), P_B(i)\} \tag{4.17}$$

$$Pset_{old}(t_i, t_{i+1}) = \{P_n(i),\ P_l(i),\ P_r(i),\ P_b(i)\} \tag{4.18}$$

Type	P_N	P_L	P_R	P_B	Freq	Ratio
1	$-$	$+$	$+$	$-$	8,652	0.9522
2	$=$	$+$	$+$	$=$	34	0.0037
3	$+$	$=$	$=$	$+$	400	0.0440
				Total	9,086	

Table 4.1: Statistics of discrepancies of probabilities

4.4.6 Analysis of the Impact of the Revised DP Model

Objective

Before we conduct experiments to verify the revised model, we have analyzed its impact to the performance of noun phrase extraction. This is achieved by comparing all $Pset_{new}(t_i, t_{i+1})$ and $Pset_{old}(t_i, t_{i+1})$ in the corpora. Any (t_i, t_{i+1}) would be picked up whenever $Pset_{new}(t_i, t_{i+1}) \neq Pset_{old}(t_i, t_{i+1})$.

Procedure

We describe the difference between $Pset_{new}$ and $Pset_{old}$ in terms of *under* and *over* estimation. For example, if $P_l < P_L$, we said that P_L is under estimated, otherwise it is over estimated. In this analysis, the data are obtained from 90% of a randomized corpus.

Results

The statistics of the analysis are shown in Table 4.1. The discrepancies that we found are classified into three types and they are shown in the second column of the table. A '−' sign means that the probability is under estimated by the original model. On the contrary, a '+' sign means that it is over estimated and a '=' sign means that the probability calculated by both models are equal. The third column shows the frequency of appearance of each type in the data set. The last column is obtained by dividing the frequency with the sum of frequency, which is 9,086. Thus, discrepancy Type 1 should be interpreted as: P_N and P_B are under estimated and P_L and P_R are over estimated by the original model. This pattern type was found in the data set for 8,652 times, which is about 95.22% of all the discrepancies found.

The properties of the three types are illustrated in the following:

Type 1

Example:

$$Pset_{new}(\text{t, ng}) = \{0.7021, 0.0000, 0.0000, 0.2979\}$$

$$Pset_{old}(\text{t, ng}) = \{0.4930, 0.2979, 0.2979, 0.0887\}$$

According to the revised model, neither left nor right boundary would be inserted in the testing stage because P_L and P_R are equal to zero. However,

for the original model, as $P_r = P_R + P_B$ (Eq. 4.15) and $P_R = 0$, $P_r = P_B$, which is equal to 0.2979. Thus a right boundary would be inserted instead if the threshold is less than 0.2979. Therefore, in the testing stage, at these positions where neither left nor right boundary appears between tag t and ng, our model can give the correct answer. 95.22% of discrepancies belong to this type.

Type 2

Example:

$$Pset_{new}(\text{npf, mx}) = \{0.0000, 0.0000, 0.0000, 1.0000\}$$

$$Pset_{old}(\text{npf, mx}) = \{0.0000, 1.0000, 1.0000, 1.0000\}$$

This type is a more specific situation of Type 1. Nearly all the patterns of this type got the same $Pset$ values shown in this example. Only 0.37% of discrepancies belong to this type.

Type 3

Example:

$$Pset_{new}(\text{vg, vh}) = \{0.2500, 0.2500, 0.5000, 0.0000\}$$

$$Pset_{old}(\text{vg, vh}) = \{0.3750, 0.2500, 0.5000, 0.1250\}$$

The P_B value of this type is equal to zero. Therefore no '] [' were inserted during testing. Our model gives the correct answer if '] [' does not exist between the tag pairs of this type. 4.40% of the discrepancies belong to this type.

Observations

The actual impact of the revised model also depends on the threshold value that described in Section 4.3. Consider the example given in Type 1. The results of extraction will be affected only when (1) the P_l and P_r greater than the threshold such that candidate boundaries are inserted; (2) at the same time, P_L and P_R obtained by the revised model should be lower than the threshold value such that no candidate boundaries will be inserted. Otherwise, if P_l and P_r less than the threshold value, there is not much difference in the calculation using both models.

Furthermore, the total number of tag pairs found in the data set is 67,480. That means the number of discrepancies found is only 13.46% of the data set.

4.4.7 Experiments of Dynamic Programming-based Model

Objective

To verify the performance of the modified probability definitions and to show the performance improvement due to them.

Training and Testing Corpus

The Chinese corpora introduced in Section 3.1.3 were used in our experiments. In each experiment, 90% of the corpus were selected randomly as the training set and the remaining for testing.

Evaluation Method

The performance was measured by two standard metrics, namely precision and recall. Precision measures the correct extraction rate of the system. Recall measures the ratio between the correct noun phrases extracted and all correct noun phrases contained in the corpus. Precision and recall are expressed by the following equations:

$$Precision = \frac{a}{a+b} \times 100\% \tag{4.19}$$

$$Recall = \frac{a}{a+c} \times 100\% \tag{4.20}$$

where a is the number of noun phrases correctly extracted, b is the number of wrongly extracted noun phrases and c is the number of noun phrases failed to be extracted. The relationship between them is shown in Table 4.2.

Procedure

Our experiments used 10-fold cross validation. The corpus was first randomized and divided into ten partitions. Each experiment consisted of ten

		Corpus	
		NP	non-NP
System	NP	a	b
	non-NP	c	-

Table 4.2: Contingency table for evaluation

trials using different partitions for training and testing. Nine partitions of the corpus were selected for training and the remaining for testing.

Recall that a candidate boundary is inserted between a pair of tags if the probability that a boundary exists exceeds or is equal to a threshold value. If the threshold is too small, the number of candidate boundaries would increase. This would increase the possibility of wrong pairing and leads to a lower precision. On the other hand, if the threshold is too large, less candidate boundaries would be inserted and thus more noun phrases would be missed during extraction which leads to a lower recall. Therefore, we have to find an optimal threshold from experiments.

In the close test, for each corpus domain, we varied the threshold from 0 to 1 and performed 10-fold cross validation. The threshold that gave the best performance was chosen as the best threshold of that domain. In the open test, the system was tested under each domain using the best threshold obtained in the close test. Other than testing under different domains, we

Domains	Close Test					
	Original			Revised		
	Th.	Precision	Recall	Th.	Precision	Recall
C	0.05	81.13	79.71	0.05	82.50	83.86
J	0.06	82.00	81.42	0.06	82.17	83.25
K	0.04	84.58	85.78	0.04	85.12	86.99
N	0.08	82.04	80.33	0.12	83.37	85.33
S	0.05	77.77	77.35	0.05	78.19	78.39
All	0.07	78.89	79.22	0.07	78.93	79.24
Average		80.97	80.33		81.63	82.81

Table 4.3: Performance of close test in each domain

Domains	Open Test					
	Original			Revised		
	Th.	Precision	Recall	Th.	Precision	Recall
C	0.05	63.72	55.94	0.05	61.35	56.51
J	0.06	73.89	69.96	0.06	72.19	70.24
K	0.04	75.46	72.26	0.04	73.54	72.07
N	0.08	77.45	72.70	0.12	77.74	76.30
S	0.05	64.92	63.77	0.05	64.06	64.18
All	0.07	73.75	73.20	0.07	73.71	73.21
Average		72.58	68.92		72.19	69.89

Table 4.4: Performance of open test in each domain

amassed all the corpus and perform both close and open tests.

Results

The best threshold and the corresponding performance in each domain are summarized in Table 4.3. The open test performance using the threshold values that obtained from the close test are summarized in Table 4.4.

Type	P_N	P_L	P_R	P_B	Freq	Ratio
1	−	+	+	−	514	0.9536
2	=	+	+	=	5	0.0093
3	+	=	=	+	20	0.0370
				Total	539	

Table 4.5: Statistics of discrepancies of probabilities in open test

Observations

On the whole, the performance of our revised DP-based noun phrase extraction approach is better than the original DP approach. However, the improvement was rather small.

4.4.8 Result Analysis

From the Table 4.5, we observe that the ratio of each type of discrepancies is similar to our analysis shown in Section 4.4.6. However, the percentage of number of discrepancies found in the testing data is only 7.66%, which is nearly half of the value obtained from the training data.

In the following, we take a closer look at the noun phrases extracted by the two models. We identify those clauses that are (i) correctly extracted by the revised model but missed by the original and those (ii) correctly extracted by the original but missed by the revised model.

Analysis of Improved Clauses

We observe that nearly all the improved clauses contain '] ['. Furthermore, nearly all the differences between boundary sequences extracted by the revised and original models are occur at the positions where '] [' appears.

The pairing of boundaries is determined by the score value (see equation 4.8). For this reason, we analyze the discrepancies of the scores calculated by the two models. This provides an explanation on why the revised model is better. Take, for example, the following sentences:

1. [1 9 3 7 年#t] [盧溝橋#s 抗戰#ng] 後#f ，

2. [1 9 3 7 年#t] 盧溝橋#s 抗戰#ng 後#f ，

where the boundary sequences 1 and 2 represent the noun phrases extracted by the revised and original models, respectively.

The scores of boundary sequences 1 and 2 using the revised model are:
Boundary sequence 1:

$$\begin{aligned} score(3) &= P_L(0) \times P_B(1) \times P_N(2) \times P_R(3) \times P_N(4) \\ &= 1.00 \times 0.6154 \times 1.00 \times 0.6154 \times 0.9762 \\ &= 0.3787 \end{aligned}$$

Boundary sequence 2:

$$\begin{aligned} score(1) &= P_L(0) \times P_R(1) \times P_N(2) \times P_N(3) \times P_N(4) \\ &= 1.00 \times 0.00 \times 1.00 \times 0.3846 \times 0.9762 \\ &= 0.00 \end{aligned}$$

Since $score(3) > score(1)$ [2], boundary sequence 1 was extracted.

Similarly, the score of boundary sequences 1 and 2 using the original model are,

Boundary sequence 1:

$$\begin{aligned} score(3) &= P_l(0) \times P_b(1) \times P_n(2) \times P_r(3) \times P_n(4) \\ &= 1.00 \times 0.3787 \times 1.00 \times 0.6154 \times 0.9762 \\ &= 0.2330 \end{aligned}$$

Boundary sequence 2:

$$\begin{aligned} score(1) &= P_l(0) \times P_r(1) \times P_n(2) \times P_n(3) \times P_n(4) \\ &= 1.00 \times 0.6154 \times 1.00 \times 0.3846 \times 0.9762 \\ &= 0.2367 \end{aligned}$$

[2] $score(3)$ and $score(1)$ are the scores of the boundary sequence end at t_3 and t_1, respectively, where $t_1 = t$, $t_2 = s$, $t_3 = ng$, $t_4 = f$ and $t_5 =$ ' , '. P_N, P_L, P_R, P_B were defined in Eq 4.9 to 4.12

In this case, boundary sequence 2 was extracted because score(1) > score(3).

Comparing the calculations, all the values used in the revised model are the same as those used in the original except $P_B(1) \neq P_b(1)$ and $P_R(1) \neq P_r(1)$. This difference is in fact the Type 1 discrepancy:

$$Pset_{new}(\text{t, s}) = \{0.3846, 0.0000, 0.0000, 0.6154\}$$

$$Pset_{old}(\text{t, s}) = \{0.1479, 0.6154, 0.6154, 0.3787\}$$

In this example, $P_r = 0.6154$ which is greater than the optimal threshold (0.07). Therefore, a right boundary was inserted by the original model. That is why we extracted correctly in such cases and this matches our estimation before the experiments.

Analysis of Weakened Clauses

Example:

1. [該#rn 隊#ng 5 #mx 次#qvp 奪得#vgn 全#a 國#ng 甲級#b 聯賽#ng 或#cpw 足協#ng 杯#ng 賽#ng 冠軍#ng] ,

2. [該#rn 隊#ng] 5 #mx 次#qvp 奪得#vgn [全#a 國#ng 甲級#b 聯賽#ng 或#cpw 足協#ng 杯#ng 賽#ng 冠軍#ng] ,

The different probability sets are:

$$Pset_{new}(\text{ng, mx}) = \{0.7778, 0.0000, 0.0000, 0.2222\}$$

$$Pset_{old}(\text{ng, ng}) = \{0.9891, 0.0000, 0.0027, 0.0082\}$$

and

$$Pset_{new}(\text{ng, ng}) = \{0.6049, 0.2222, 0.2222, 0.0494\}$$

$$Pset_{old}(\text{ng, ng}) = \{0.9810, 0.0082, 0.0109, 0.0001\}$$

The above discrepancies also belong to Type 1. However, we could not make a correct extraction this time. It is because $P_R(ng, mx)$ is equal to zero, the revised model would not insert a boundary between them. However, this situation happened in the testing corpus (隊#ng] 5 #mx), therefore we failed to extract it. According to Eq. 4.15, P_r becomes 0.2222, which is greater than the optimal threshold (0.07). Therefore, a right boundary was inserted by the original model and thus the noun phrase could be extracted correctly. This example shows that the new probability assessment might weaken the extraction performance.

For the tag pair (ng, ng), since $\{P_L, P_R, P_B\}$ and $\{P_l, P_r, P_b\}$ are less than the threshold value (0.07), thus no boundary would be inserted by *both* model. Furthermore, as the difference between $\{P_L, P_R, P_B\}$ and $\{P_l, P_r, P_b\}$ are so small, their effect on the scores calculated by both the revised and original model, respectively, would have not much differences too. This case indicates that the performance of the revised model might be indifferent with the original one.

4.5 Concluding Remarks on DP-Based Model

When evaluating the correctness of a candidate noun phrase boundary sequence, our proposed model is trying to consider as much information as possible: all the boundary probabilities at each position of the sentence are included in the calculation (see Equation 4.8). However, the equation assumes that the boundary probabilities are independent to each other. In fact, there exists relationships between them and they are sometimes useful for the determination of boundary position. Consider the following example:

Correct: 提出#vgn 了#utl [自動#d 抽#vgn 詞#ng 標引#vg 的#usde 思想#ng] 。

Wrong: 提出#vgn 了#utl 自動#d 抽#vgn [詞#ng 標引#vg 的#usde 思想#ng] 。

There are two candidate left boundaries, one lies between "了#utl" and "自動#d", the other lies between "抽#vgn" and "詞#ng". The later one is chosen because $P_L(vgn, ng) > P_N(vgn, ng)$ and $P_N(utl, d) > P_L(utl, d)$. However, if we consider the word pair "提出#vgn 了#utl", the probability that the left boundary is placed after "了#utl" is higher because the tag sequence "vgn utl" are commonly used to introduce a noun phrase.

This example shows that independent probability may be insufficient to locate the correct boundary and conditional probability is useful in some situations. However, the conditional probability model that mentioned in

Section 4.4.1 is also not a good candidate for this purpose because:

- the size of feature space must be fixed before calculations. We cannot adjust the feature space for a specific instance. However, the same feature space may not be suitable for every instances. In this example, a size of 3-gram is sufficient. However, this may not be adequate at other instance. Therefore, the *CP-based* model is not flexible enough to handle different cases.

- the model is comparatively more demanding in terms of storage space, training data and training time

In order to enhance the performance of our extraction system with the properties of conditional probabilities, we applied an algorithm called transformation-based error-driven learning (TEL) to our system. By using TEL, the feature space need not be fixed before calculation. Different settings will be enumerated automatically in each run. Furthermore, statistical information such as the conditional probability that we are discussing are represented by simple and comprehensive transformation rules and they are learned automatically. Also, the computation time for testing is fast because it is proportional to the length of the testing corpus. The details of TEL and how it is applied to our system will be discussed in the next chapter.

4.6 Chapter Summary

In this chapter, we have proposed a noun phrase extraction model. The central component of the model is the pairing of candidate boundaries. Li [23] introduced a conditional probability-based (CP) model, a heuristic-based model call maximal probability (MP) and a dynamic programming-based (DP) model to achieve this task. The DP-based model is adopted after the evaluation and comparison with the other models. However, in our investigation on the DP-based model, we found that Li had over-simplified the probability definitions. Therefore, we revised the DP-based model and better results were obtained.

In the DP-based model, boundary probabilities are assumed to be independent and we have shown that this assumption make it unable to solve some kind of problems. Therefore we proposed to apply an algorithm called transformation-based error-driven learning (TEL) to tackle the problems and to enhance the performance. We will discuss in details the application of TEL algorithm in our system in the next chapter.

Chapter 5

Automatic Error Correction

Automatic Error Correction is the second module of our noun phrase extraction system. It applies transformation-based error-driven learning (TEL) technique to refine the output of the preliminary noun phrase extraction module [22]. In this chapter, we first briefly introduce the TEL algorithm. Then details of the settings of the main components of TEL is described.

5.1 Introduction

Transformation-based error-driven learning (TEL) [8] is a machine learning algorithm which combines the advantages of statistical-based and rule-based techniques. It has been successfully applied to a number of natural language problems including part-of-speech (POS) tagging [2, 6, 10, 11, 12, 28], prepo-

sitional phrase attachment disambiguation [13], syntactic parsing [7, 9, 32], spelling correction [27], Chinese word segmentation [19], and noun phrase extraction [20, 29].

The framework of TEL is shown in Figure 5.1. In the learning process, an unannotated corpus is first presented to the initial annotation. The initial annotator assigns initial values to the interested features of the corpus according to the pre-specified knowledge. For example, in POS tagging applications, the features are defined as POS tags of word, and the initial annotator should assign the most likely tag to every word.

After the corpus is annotated by the initial annotator, it is then compared with the true annotation, which is a manually annotated training corpus. Through the comparison, one could learn the errors produced by the initial annotator. Wherever there is an error, a transformation rule will be learned to correct it. The format of a transformation rule is pre-specified by a set of transformation templates. The transformation templates specify the features that trigger a transformation and the corresponding transformation actions. The transformation rule that can correct most errors are added to the transformation rule set. The learned rules are then applied to the whole corpus to correct the errors specified by the rule. This procedure repeats until the number of error reduction, or what we called the *gain*, is less than or equal to a pre-specified *stopping threshold*. Figure 5.2 shows an example

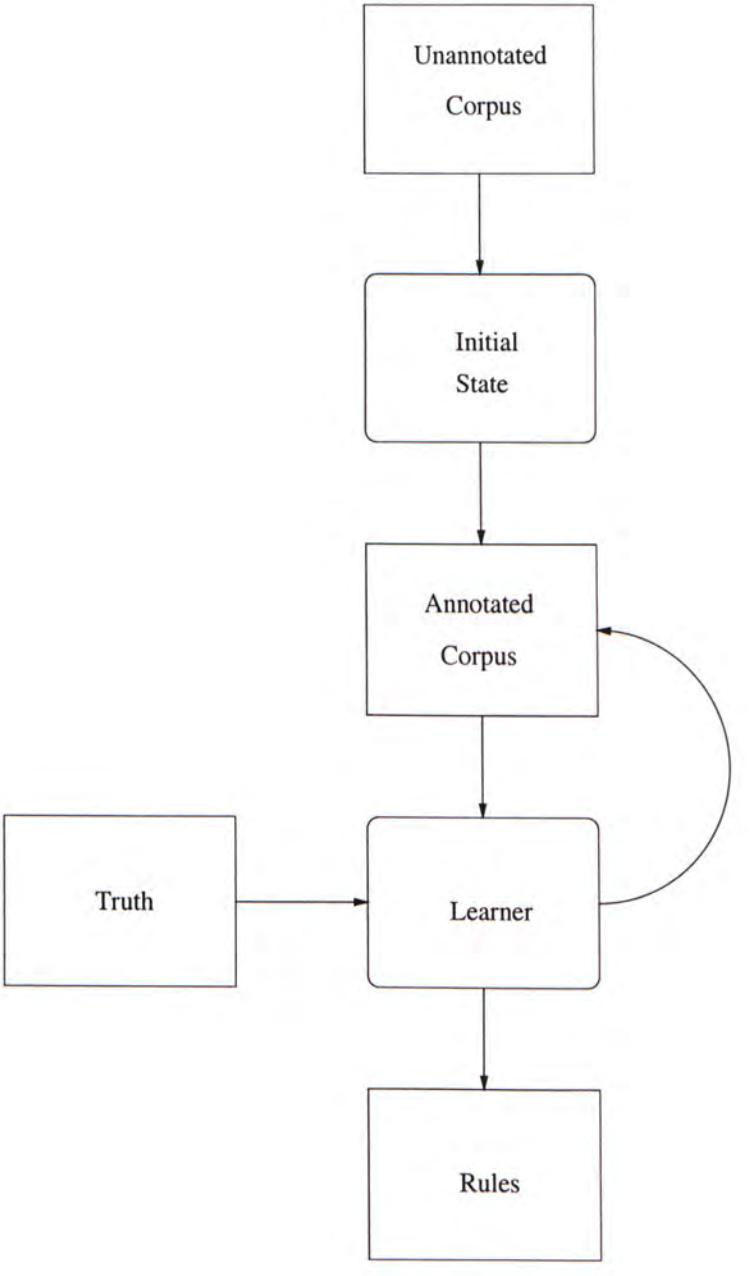

Figure 5.1: Framework of TEL

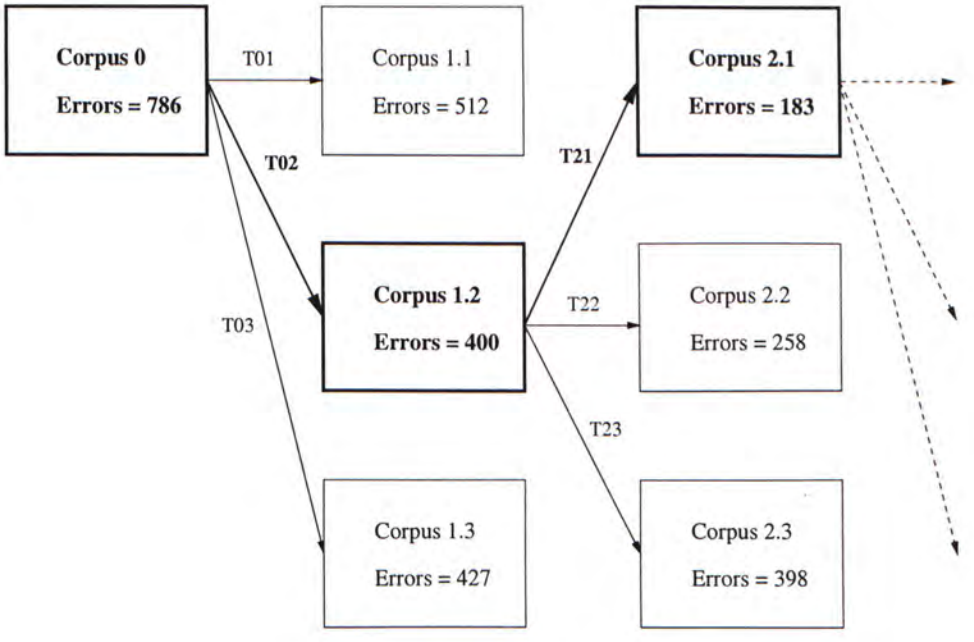

Figure 5.2: An Example of transformation rules learning process

of the learning process.

In Figure 5.2, the initial annotated Corpus 0 contains 786 errors. After examining the three possible transformations rules, transformation rule T02 gives the largest reduction of errors (i.e. decreased by 386). Therefore T02 is adopted in the transformation rule set. It is then applied to Corpus 0 which results in Corpus 1.2. This time, Corpus 1.2 contains 400 errors. T21 is found to give the largest reduction of errors (i.e. decreased by 217) and is added to the rule set. The learning process continues until there is no error reduction.

The testing process is shown in Figure 5.3. Corpus 0 is obtained by

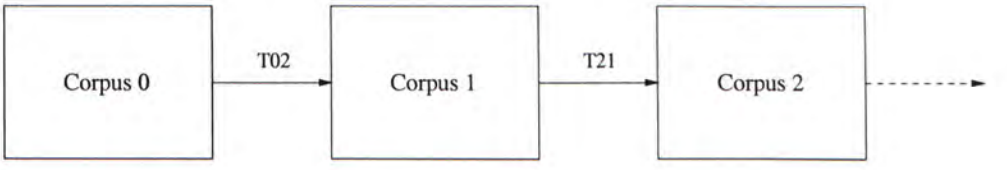

Figure 5.3: Applying transformation rules in testing

presenting an unannotated testing corpus to the initial annotator. The first learned rule T02 is then applied to Corpus 0 which results in Corpus 1. The other learned rules stored in the rule set will be applied to the annotated corpus one by one until the list of rules is exhausted.

5.1.1 Statistical Properties of TEL

In each iteration, a rule is learned to correct the most frequently appeared errors under the conditions specified by the transformation template. The probability of error occurrence is represented by the following equation:

$$P(error|condition) = \frac{freq(error|condition)}{freq(condition)}$$

Therefore, a rule can be understood as the probability of the occurrence of the error in that iteration. The earlier the rule is learned, the larger the probability that the error occur.

Figure 5.4 shows the relationship between the number of error reduction, i.e. the *gain*, and the number of iterations.[1] The gain is reduced after each

[1]The values shown in Figure 5.4 are examples only.

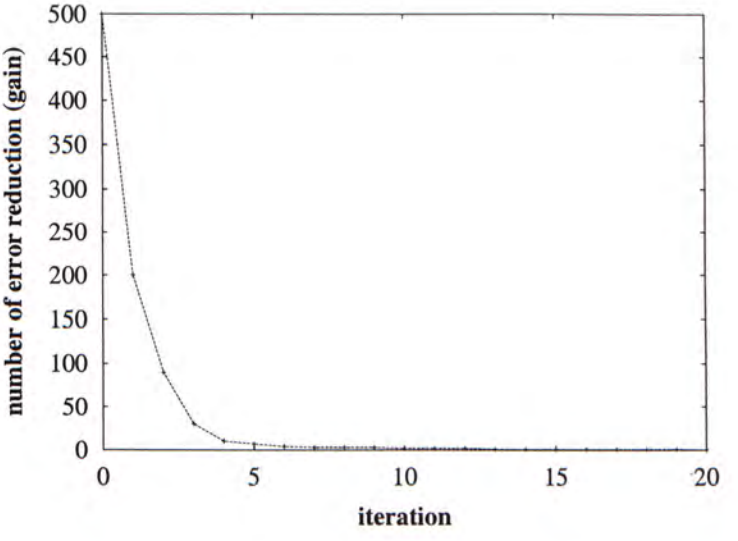

Figure 5.4: Relationship between number of error reduction and number of iteration

iteration. The reduction is large in early iterations and is slow down in subsequent transformations. The learning stop when the *gain* equals to zero or a threshold value.

5.1.2 Related Applications

Ramshaw and Marcus [29] applied Brill's transformation-based POS tagging model [11] on English minimal noun phrase (minNP) extraction. A chunk tag set $\{I, O, B\}$ is defined, where I and O mark a word which is inside a minNP or outside a minNP, respectively. The left most word of a minNP which immediately follows another minNP is marked with B. Generally speaking,

the chunk tag tells us whether a word is part of a minNP or not. With this representation, the minNP extraction problem will become a chunk tagging problem and Brill's POS tagger can be used.

In the initial state, each POS tag in the corpus is assigned the chunk tag that is most frequently associated with that POS tag in training. The features of the transformation templates include POS and lexical information. The transformation action is defined as changing from one chunk tag to another. The recall and precision of the system are 92.3% and 91.8% respectively.

Zhao and Huang [20] further modified Ramshaw's system [29] and applied it to Chinese minimal noun phrases extraction. The features of transformation include POS tag, semantic class, number of syllable and the grammar rule that the candidate minNP belongs. In the initial state, minNPs is identified with a set of minNP grammar rules. The grammar rules are actually the frequently appeared minNP pattern which are obtained from a training corpus. The output of the initial state is a set of minNPs which stores in a *candidate* list. The output is then compared with the correct answer, namely the *correct* list. If a noun phrase in the *candidate* list is found to be correct, it is moved to a $confirm$ list. Otherwise, it is removed from the *candidate* list. Similarly, incorrect noun phrases of the $confirm$ list is removed if they could not be found in the *correct* list. Thus, the transformation actions are defined as (i) moving a noun phrase from the *candidate* list to the $confirm$

list, (ii) removing a noun phrase from *candidate* list, (iii) removing a noun phrase from *candidate* list and, (iv) adding a noun phrase to $confirm$ list. After the learning process, a set of transformation rules and a $confirm$ list of minNP are obtained. The recall and precision in close test are 93.2% and 91.1% respectively, while the recall and precision in open test are 91.8% and 87.3% respectively.

These two systems are designed for minimal noun phrase extraction and the results are impressive. However, as we pointed out in Chapter 2, the structure of maximal noun phrase is more complicate than minimal noun phrase, therefore, we have to evaluate the performance of applying TEL algorithm to maximal noun phrase extraction.

5.2 Settings of Main Components

Our system is based on the framework described in Section 5.1. In this section, we describe in details the components of the system. This includes the initial state, the transformation templates, the evaluation function and the stopping threshold.

5.2.1 Initial State

Since our target is to improve the results given by the preliminary noun phrase extraction module with the TEL algorithm, therefore the module will be used as our initial state.

5.2.2 Transformation Actions

Two transformation actions are adopted in our system. They are removing and adding noun phrases. The objective of adding noun phrases is to improve recall for more correct noun phrases missed in the initial state could then be recognized. On the other hand, the objective of removing noun phrases is to improve precision as the wrong noun phrases extracted in the initial state could then be removed.

5.2.3 Triggering Features of Transformation Templates

Triggering features specify the condition for a transformation to take place. Wherever the triggering conditions are matched in the corpus, the corresponding transformation action is carried out.

Importance of feature selection

The selection of triggering features is essential to the success of the system. The triggering features must be able to identify most of the errors from the initial state. If the learned rules can only be applied to a few instances of the training corpus, it is less likely that it could be matched with any instances in the testing corpus and thus no transformations would be achieved.

Feature selection

Contextual information are commonly used in noun phrase extraction to determine the noun phrase boundary as some words are often used to indicate the beginning or ending of a noun phrase. For example, verbs such as "建立" (build) and "提出" (suggest) are often used in verb-object structures and they are usually followed by a noun. Therefore, these words can help us verify the beginning of a noun phrase.

Two commonly used contextual information are POS and semantic information. A POS tag describes the grammatical function of a word and is essential in representing the internal structure of a sentence. Therefore it plays a very important role in natural language processing. Many effective and reliable POS taggers were developed for different languages. Thus POS tags are chosen as the basis of the triggering features.

In Li's work [23], semantic information is used to resolve structural ambiguities of relative clauses and prepositional phrase. The Chinese semantic information is obtained from a lexicon called CILIN (同義詞詞林) [3]. It contains 65,536 word entries that were classified into 12 major, 95 medium and 1,428 minor semantic classes. However, CILIN has some drawbacks: (i) a large number of frequently used words are missing, (ii) there are no classes for special nouns such as name of place, name of person and name of an organization etc, and (iii) a word often belong to more than one class and there is no semantic tagger currently. Due to these drawbacks, semantic information is not adopted as a feature in our extraction system.

Complexity of the triggering environment

The complexity of the triggering environment describes the number of triggering features that we consider. For example, let the triggering environment are defined as the fist and last POS tag of a noun phrase, namely T_{b+0} and T_{e+0}. The possible triggering environment, which we have to consider are (i) T_{b+0}, (ii) T_{e+0} and (iii) T_{b+0} and T_{e+0}. The total number of possible triggering environment can be calculated by the following equation:

$$|env| = \sum_{r=1}^{n} {}_nC_r \tag{5.1}$$

where n is the number of triggering features.

For error-driven learning to be feasible, the complexity of triggering environment should not be too high because:

- higher complexity increases the probability of learning over-fitting rules,
- as shown in Equation 5.1, the number of possible triggering environments would increase exponentially as the complexity increases. Since the run-time of the learning algorithm is $O(|a| \times |env| \times |n|)$, where $|a|$ is the number of allowable transformation actions, $|env|$ is the number of possible triggering environments, and $|n|$ is the training corpus size, a large set of environments would make learning computationally infeasible.

In our investigation, the triggering environment are defined as:

$$T_{b-2} \rightarrow T_{b+2} \quad and \quad T_{e-2} \rightarrow T_{e+2}$$

T_{b-2} and T_{b+2} represent the POS tags of the second preceding word and the second successive word of the first word of a noun phrase. T_{e-2} and T_{e+2} represent the POS tags of the second preceding word and the 2nd successive word of the last word of a noun phrase. Within this range, all possible combinations of POS tag positions will be enumerated. An excerpt of the possible environments are shown in Appendix C

5.2.4 Evaluation of Rule

The performance of a candidate transformation rule is evaluated by applying it to the whole corpus and count the number of correct and incorrect transformations. We define the *gain* of a rule r as:

$$gain_r = C_r - E_r$$

where C_r and E_r are the number of correct and incorrect transformations respectively. The rule with highest *gain* is chosen. If more than one rule have an equal *gain*, the one with more correct transformations is chosen.

5.2.5 Stopping Threshold

Recall that the transformation learning continues until the *gain* is less than or equal to the stopping threshold. As pointed out by Brill [8], a higher threshold has the advantage of only learning transformations with higher probability of begin useful, thus lessening the amount of over-training and speeding up run time. But, a higher threshold could hurt performance by throwing away effective low frequency transformations. We varied the threshold value in the experiments and observed their effect on performance (see Section 5.3.2).

5.3 Experiments and Results

5.3.1 Setup and Procedure

In the following experiments, the corpora that introduced in Section 3.1.3 were used as training and testing data. We used 10-fold cross validation to verify our results. In the training stage, the training corpus was first presented to the preliminary noun phrase extraction module. The output was compared with the correctly annotated corpus and an ordered list of transformation rules were acquired.

In the testing stage, the testing corpus first underwent the preliminary noun phrase extraction module. The transformation rules obtained in training were then applied to the corpus one by one until the list of rules was exhausted. If there were ambiguities, i.e. if overlapping noun phrases exist, the longest noun phrase will be chosen because we are target on maximal noun phrase extraction.

5.3.2 Overall Performance

Figure 5.5 shows the overall performance[2] of our system in open test. From the graph, we observe that the precision decreases and the recall increases as the threshold increases. To determine the best threshold, we use *F-measure*

[2]All the figures and discussion given in this chapter are averaged over the 10-fold tests.

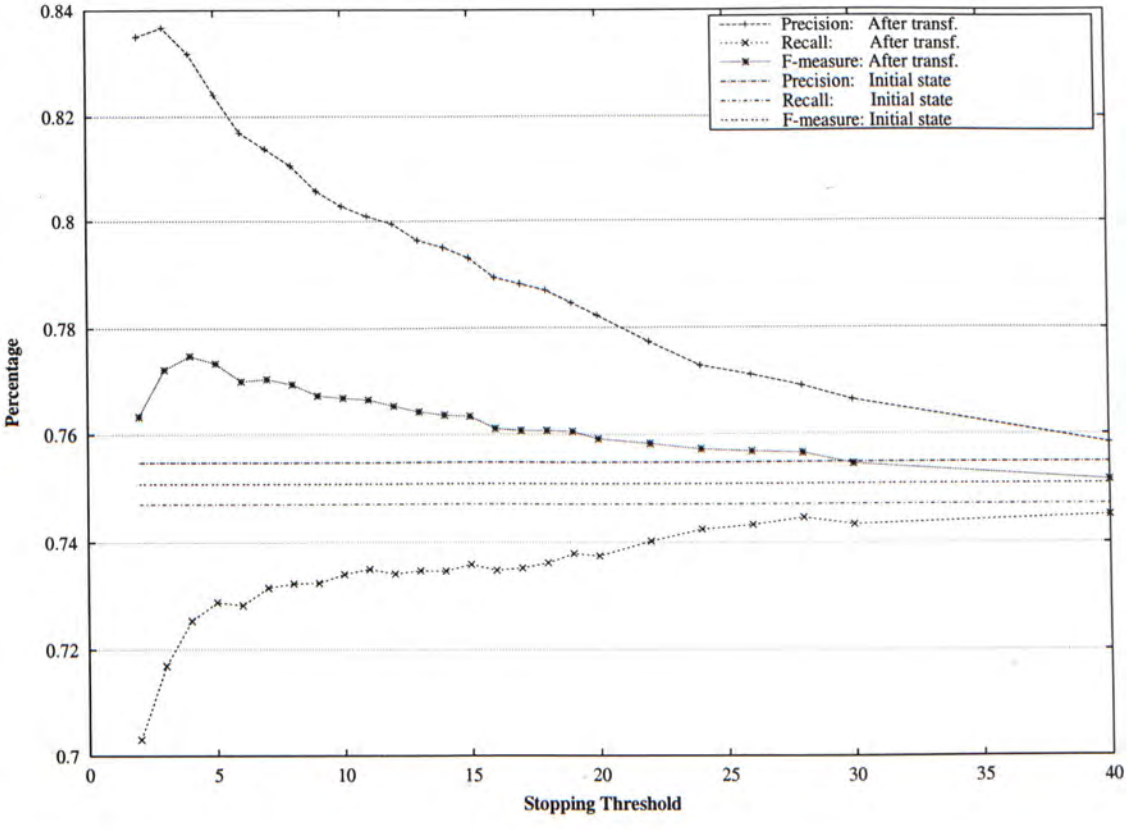

Figure 5.5: Overall performance in open test with different stopping threshold values

Stopping Th. = 4	Close test			Open test		
	Recall	Precision	F-mea.	Recall	Precision	F-mea.
Initial state	0.7949	0.7937	0.7943	0.7472	0.7549	0.7510
After transf.	0.8025	0.8964	0.8468	0.7253	0.8316	0.7748

Table 5.1: A comparison of the performance before and after transformations

to express the performance. The expression of *F-measure* is:

$$F_b = \frac{(b^2 + 1) \times precision \times recall}{b^2 \times precision + recall} \tag{5.2}$$

The value of b is set to 1 as we assume that the precision and recall are equally weighted. *F-measure* obtains the largest value when the threshold equals to 4.

The performance of the initial state is shown for comparison and it is plotted on the graph as the three horizontal lines. Both the precision and F-measure are found to be improved over all thresholds. However, there is a degradation in recall.

The results with threshold equals to 4 is summarized in Table 5.1. The precision has a significant improvement after the transformation in both close and open tests. However, the recall has only a little improvement in close test but a slight decrease in open test.

To analyze the results, we represent the performance in terms of the

Stopping Th. = 4	Close test		Open test	
	cor-NP	non-NP	cor-NP	non-NP
Initial state	8,384.4	2,179.3	875.5	284
After transf.	8,464.2	983.9	849.9	172.2

Table 5.2: Performance of the system in terms of the number of correct and wrong noun phrases

	Close test	Open test
Δ cor-NP	+79.8 (+0.95%)	-25.6 (-2.92%)
Δ non-NP	-1,195.4 (-54.85%)	-111.8 (-39.37%)

Table 5.3: Changes after transformations

number of correct and wrong noun phrases. The values are shown in Table 5.2 and the changes of them are shown in Table 5.3.

According to Table 5.3, 54.85% and 39.37% of incorrect noun phrases are removed in close and open tests, respectively. Therefore we have an significant improvement in precision. This results show that our system can successfully remove many incorrect noun phrases that extracted in the initial state. However, only 0.95% of correct noun phrases is inserted in close test. A number of correct noun phrases are removed in open test during transformations. This results a decrease in recall.

Rules	Num. of correct transf.	Num. of incorrect transf.	Num. of rules	Num. of un-executed rules
Removal	116.4	55.9	101.1	40.9
Insertion	42.4	24.0	56.2	28.2

Table 5.4: Statistics of transformation rules

5.3.3 Contribution of Rules

Table 5.4 shows the statistics of the learned rules. On average, 157.3 transformation rules are learned (with stopping threshold equals to 4) in the training process. Out of them, 101.1 are removal rules and 56.2 are insertion rules. The number of correct transformation represents how much errors were corrected and the number of incorrect transformations represents the number of errors made during transformations. Note that although a number of errors were made during transformations, they might be fixed by the subsequent transformation rules. The last column shows the number of un-executed rules.

Removal Rules

As shown in Table 5.3, 111.8 incorrect noun phrases were removed. An example is shown below to illustrate the operation of removal rules.

Correct: 據#p 統計#vg 在#pzai [傳統#ng 的#usde 指令#ng 系統#ng 計算機#ng 執行#vgn 的#usde 程序#ng] 中#f ，#xnull

Wrong: 據#p 統計#vg 在#pzai [傳統#ng 的#usde 指令#ng 系統#ng 計算機#ng] 執行#vgn 的#usde 程序#ng 中#f ，#xnull

The following transformation rule is learned and applied to correct the error:

- if $T_{e+2} = usde$, then remove the noun phrase

A close bracket is placed between "計算機#ng" and "執行#vgn" because "ng" and "vgn" are often recognized as the subject and the main verb of a sentence respectively. The probability of $P_R(ng, vgn) = 0.8154$ shows the reason why a right boundary is inserted between them. However, if we look one word forward, i.e. the word "的#usde", we know that "計算機#ng 執行#vgn" is serving as a modifier. Therefore the end of noun phrase should not be placed between "計算機#ng" and "執行#vgn" and thus it was removed.

Insertion Rules

Consider the example shown above. The correct noun phrase was inserted in the subsequent transformation by the following rule:

- if T_{b-1} = pzai & T_{b+0} = ng & T_{b+2} = ng & T_{e+1} = f & T_{e+2} = xnull, then insert a noun phrase

The pattern "在#pzai $\cdots$ 中#f" are commonly used as a mobile adverbial modifier in a sentence and a noun phrase is found between them [18]. The rule is learned basing on this property and therefore the noun phrase is recognized by the rule.

Such kind of noun phrase might not be extracted in the initial state because the boundary position is depends on the POS tag of the word behind "在#pzai" and in front of "中#f". For instance, the boundary probabilities between ng and f are $P_N(ng, f) = 0.5432$ and $P_R(ng, f) = 0.4568$. Therefore, it is less likely that a '] ' will be inserted between ng and f.

5.3.4 Remarks on Rules Learning

The examples shown above demonstrate some essential properties of our system:

1. Flexibility of feature space. Different location and size of feature space were enumerated and used in the experiment. The size of feature space

of the removal rule shown in the above example is 1 and the location of feature is the second successive word. For the insertion rule, the size of feature space is 5, where 3 of them are the surrounding tags of left boundary and the remaining are surrounding tags of the right boundary. With this flexibility, different cases of errors can be handled.

2. Conditional probabilities are represented as transformation rule instead of a large set of values. As mentioned in 5.1.1, transformation rule is learned to correct the errors that appeared most frequently in each iteration. Therefore, a transformation rule can be understood as the probability that a noun phrase should be remove/insert given the condition specified in the rule. The earlier the rule is learned, the higher the probability.

3. The learned transformation rules are comprehensive linguistic rules. Users can easily modify the rules according to their need.

5.3.5 Discussion on Recall Performance

Several reasons for the degradation of recall were found and listed as follow:

1. As shown in Table 5.4, the number of insertion rules learned is nearly half the number of removal rules. This lessen the chance of inserting correct noun phrases. Furthermore, the number of correctly in-

- T_{b-2} = xnull & T_{b+0} = rn & T_{b+1} = vgn & T_{e-2} = ng & T_{e-1} = usde & T_{e+0} = ng
- T_{b-2} = vgv & T_{b+0} = ng & T_{b+2} = ng & T_{e-2} = ng & T_{e-1} = usde & T_{e+1} = xnull
- T_{b-2} = xnull & T_{b-1} = vgn & T_{b+0} = ng & T_{b+1} = vg & T_{e-1} = ng & T_{e+1} = xnull
- T_{b-2} = p & T_{b-1} = vgn & T_{b+2} = ng & T_{e-2} = ng & T_{e-1} = usde & T_{e+0} = ng & T_{e+1} = xnull

Figure 5.6: Examples of un-executed insertion rules

serted noun phrases is less than the number of incorrectly removed noun phrases, thus results a decrease in the total number of correct noun phrases. Therefore the recall decreased.

2. Most of the insertion rules are learned in later transformations. The *gain* of those rules are low as the majority of errors were corrected in early transformations. The remaining errors are specific to the training corpus thus it is unlikely that the learned rules can be applied to the testing corpus. In fact, 28.2 insertion rules out of 56.2 were not executed, and most of them are learned in later transformations. Some of the triggering condition of un-executed insertion rules are shown in Figure 5.6. Note that their complexity is comparatively high.

3. Theoretically, a general rule is preferred to a specific rule as more instances can be matched and thus more errors can be corrected in each transformation. However, it is not true to the insertion rules. For example, given a candidate insertion rule

   - if T_{b-1} = vgn & T_{e+0} = ng, then insert a noun phrase,

   if it is used to identify noun phrase(s) in the sentence " 掌握#vgn [物理#ng 概念#ng 和#cpw 規律#ng 的#usde 物理#ng 意義#ng 和#cpw 適用#a 範圍#ng] 。#xnull", then six candidate noun phrases will be found because the sentence contains six "ng" which are all candidate noun phrase ending according to the rule. However, only the noun phrase shown above is correct. Therefore the *gain* of this rule equals to one minus six which is a negative value and the rule will not be accepted.

   On the other hand, for a rule with higher complexity, for example:

   - if T_{b-1} = vgn & T_{b+2} = cpw & T_{e-1} = a & T_{e+0} = ng, then insert a noun phrase

   then only one pair of boundaries match the specified condition and the correct noun phrase will be identified. This explains why the complexity of learned insertion rules are comparatively high.

5.4 Chapter Summary

In this chapter, we proposed a transformation-based model to correct the errors induced from our preliminary noun phrase extraction module. The output of the module is first compared with the correct corpus and a set of transformation rules are learned to correct the errors. The format of transformation rule is specified by a template, which consists of triggering conditions and transformation actions. The triggering conditions are defined as the surrounding POS tag of a noun phrase and the transformation actions are defined as insert and remove noun phrases.

In the open test, the average precision after transformation has a significant improvement from 75.49% to 85.16%. However, the recall decreased slightly from 74.72% to 72.53%. After the analysis of the statistics of the results, we found that most of insertion rules learned have a high complexity and they are over-fit to the training corpus. Therefore, many of them cannot be executed in testing and thus little correct noun phrases were inserted. This results a drop of recall.

Chapter 6

Conclusion

6.1 Summary

In this thesis, a corpus-based maximal Chinese noun phrase extraction system is proposed. The system consists of two stages: the first stage is designed for preliminary noun phrase extraction and the second for automatic error correction.

The preliminary noun phrase extraction operates in two steps: finding candidate boundaries and pairing the left and right boundaries to form noun phrases. A noun phrase annotated corpus is presented to the module for training and the boundary probabilities is learned. The probabilities are used for identifying the candidate boundaries. Once the candidate boundaries are identified, the correct boundaries for noun phrases are determined. Different

approaches to boundaries pairing are evaluated. The dynamic programming-based (DP) approach was adopted and later modified to enhance noun phrase extraction performance.

Through error analysis, we observe that the information considered by the DP approach is insufficient to identify some kind of noun phrases. In fact, the extraction performance can be improved by considering more contextual information. For this purpose, we have evaluated the feasibility of using conditional probability approach. However, this approach is very demanding in terms of training time and memory usage. Therefore, another approach called transformation-based error-driven learning (TEL) is proposed.

The objective of using TEL is to refine the output of the preliminary noun phrase extraction module automatically. In the learning stage of TEL, the output of the preliminary noun phrase extraction is compared with the true annotation. Through the comparison, TEL can learn the errors produced by the preliminary extraction module. Wherever there is an error, a transformation rule will be learned to correct it. The learned rules are then applied to the whole corpus to correct the errors specified by the rule. This procedure repeats until the number of error reduction is less than or equal to a stopping threshold. In the testing stage, a corpus is first annotated by the preliminary noun phrase extraction module. The transformation rules stored in the rule set is applied to the newly annotated corpus one by one

until the list of rules is exhausted.

The procedures of learning and correcting errors are fully automatic. The statistical information of the errors are represented as a set of simple and comprehensive transformation rules. Also, the computation time in testing is fast as it is proportional to the length of the testing corpus.

In open test, the extraction precision improved from 75.49% to 83.16%. This shows that our error correction module is effective in identifying and removing the problematic noun phrases.

6.2 Contributions

In this research, we proposed a fully automatic maximal Chinese noun phrase extraction system. Previous studies on rule-based noun phrase extraction system requires considerable effort in the manually design of linguistic or heuristic rules. However, the rules used in our system are learned automatically from the output of a statistical approach. Empirical results show that our system discover noun phrases effectively.

6.3 Future Work

Several directions of future research are worth mentioning:

1. Other than altering the triggering features, we may also try to add different transformation actions. Currently, the transformation actions are remove and insert noun phrase. However, experimental results show that the insert action is ineffective in adding correct noun phrases. In fact, we may identify more noun phrases by correcting existing errors. For example, (i) if a consecutive noun phrase is found to be split into two or more parts or, (ii) on the contrast, different functional units are mistakenly combined to form one noun, we can define a 'combine' and a 'split' action to correct the errors directly, respectively.

2. In our error correction module, POS tags is the only contextual information used as the triggering features for transformation. We could consider more information such as lexical and semantic information to assist in fixing errors. In fact, semantic information provides useful information in determining the noun phrase boundary position. It shows advantages in resolving structural ambiguities due to relative clause and preposition phrase. However, as pointed out in this thesis, semantic information is not adopted because there is no semantic tagger yet. Therefore, we could consider developing a semantic tagger.

3. The size of the corpus that we used in this research is small and thus we apply 10-fold cross validation to verify the performance of our system.

In fact, we can further improve the learning process by preparing a larger corpus. Furthermore, we can collect texts from different districts (e.g. Hong Kong and Taiwan) so that different styles of Chinese writing can be learned.

Bibliography

[1] 漢語詞性自動標注系統 - 技術報告. 清華大學計算機科學與技術系, 1992.

[2] 白明弘, 陳超然, 陳克健. 以語境判定未知詞詞類的方法. In *Proceedings of ROCLING XI*, 1998.

[3] 梅家駒，竺一鳴，高蘊琦，殷鴻翔. 同義詞詞林. 上海辭書出版社, 1984.

[4] 俞士汶. 關於現代漢語詞語的語法功能分類. 中國計算機報, 1994.

[5] D. Bourigault. Surface grammatical analysis for the extraction of terminological noun phrases. In *Proceedings of the Fifteenth International Conference on Computiional Linguistics. COLING-92, Vol.III*, pages 977–981, Nantes, France, 1992.

[6] Eric Brill. A simple rule-based part of speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*, 1992.

[7] Eric Brill. Automatic grammar induction and parsing free text: A transformation-based approach. In *Proceedings of the 31st Meeting of the Association of Computational Linguistics*, Columbus, Oh, 1993.

[8] Eric Brill. *A Corpus-based approach to language learning.* PhD thesis, University of Pennsylvania, 1993.

[9] Eric Brill. Transformation-based error-driven parsing. In *Proceedings of the Third International Workshop on Parsing Technologies*, Tilburg, the Netherlands, 1993.

[10] Eric Brill. Some advances in rule-based part of speech tagging. In *Proceedings of the Twelfth National Conference on Artifical Intelligence (AAAI-94)*, Seattle,Wa, 1994.

[11] Eric Brill. Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging. In *Computational Linguistics*, volume 21 no. 4, pages 543–565, 1995.

[12] Eric Brill. Unsupervised Learning of Disambiguation Rules for Part of Speech Tagging. In *Natural Language Processing Using Very Large Corpora Kluwer Academic Press.*, 1997.

[13] Eric Brill and Philip Resnik. A transformation-based approach to prepositional phrase attachment disambiguation. In *Proceedings of the Sixteenth International Conference on Computational Linguistics (COLING-1994)*, Kyoto, Japan, 1994.

[14] Claire Cardie and David Pierce. Error-driven pruning of treebank grammars for base noun phrase identification. In *Proceedings of COLING-ACL'98*, pages 218–224, 1998.

[15] K.H. Chen and H.H. Chen. Extracting noun phrases from large scale texts: A hybrid approach and its automatic evaluation. In *Proceedings of the 32nd Annual Meeting of ACL*, pages 234–241, New Mexico, USA, 1994.

[16] K. Church. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of Second Conference on Applied Natural Language Processing*, pages 136–143, Austin, Texas, USA, 1988.

[17] K. Church. Current practice in part of speech tagging and suggestions for the future. In *Sbornik praci: In Honor of Henry Kucera*. Michigan Slavic Studies, 1992.

[18] D.Z. Han. *Fifty patterns of modern Chinese*, page 225. The Chinese University Press, 1993.

[19] J.Hockenmaier and C.Brew. Error-Driven Learning of Chinese Word Segmentation. In *12th Pacific Conference on Language and Information*, 1998.

[20] Zhao Jun and Huang Changning. A study on Chinese baseNP recognition. In *Quantitative and Computational Studies on the Chinese Language*, pages 307–321, 1998.

[21] K.C.Chan. A dynamic programming-based approach for chinese noun phrase extraction. In *The First ACM Hong Kong Postgraduate Research Day*, pages 132–138, Hong Kong, October 1998.

[22] K.F.Wong, K.C.T.Chan, and C.H.Cheng. An investigation on transformation-based error-driven learning algorithm for chinese noun phrase extraction. In *2000 International Conference on Chinese Language Computing*, Chicago, USA, July 2000.

[23] W. Li. *Automatic Noun Phrase Extraction from Full Chinese Text*. PhD thesis, The Chinese University of Hong Kong, September 1997.

[24] W. Li, K.F. Wong, B.T. Low, A. Tse, and V. Lum. Chinese noun phrase extraction based on the statistical distribution information. In *Proceedings of 1996 International Conference on Chinese Computing (ICCC'96)*, pages 69–74, Singapore, June 1996.

[25] C Lyon and R Dickerson. A fast partial parse of natural language sentences using a connectionist method. In *7th Conference of the European Chapter of the ACL*, 1995.

[26] D. Magerman and M. Marcus. Parsing a natural language using mutual information statistics. In *Proceedings of the European ACL Conference*, pages 984–989, 1991.

[27] Lidia Mangu and Eric Brill. Automatic Rule Acquisition for Spelling Correction. In *ICML 97*, 1997.

[28] Helen Meng and C. W. Ip. An Analytical Study of Transformational Tagging for Chinese Text. In *Research on Computational Linguistics Conference (ROCLING)*, Taipei, Taiwan ROC, 1999.

[29] L. A. Ramshaw and M. P. Marcus. Text chunking using transformation-based learning. In *Proceedings of the Third Workshop on Very Large Corpus*, pages 82–94, Cambridge, Massachusetts, USA, 1995.

[30] Geoffrey Sampson. The SUSANNE corpus. *ICAME Journal*, 17:125–127, 1993.

[31] Geoffrey Sampson. *English for the Computer*. Oxford University Press, 1994.

[32] Giorgio Satta and Eric Brill. Efficient Transformation-Based Parsing. In *ACL 1996*, 1996.

[33] Anne Schiller. Multilingual finite-state noun phrase extraction. In *Proceedings of the ECAI-96 Workshop on Extended Finite State Models of Language*, 1996.

[34] Wojciech Skut and Thorsten Brants. Chunk Tagger, Statistical recognition of noun phrases. In *ESSLI-98 Workshop on Automated Acquistion of Syntax and Parsing*, Saarbrücken, Germany, 1998.

[35] Voutilainen and Atro. NPtool: a detector of english noun phrases. In *Proceedings of Workshop on Very Large Corpora: Academic and Industr ial Perspectives*, pages 48–57, 1993.

Appendix A

Chinese POS Tag Set

1. n (Noun 名詞)
    - nf (Surname 姓氏)
    - Proper noun 專有名詞
        - npf (Personal name 人名)
        - npu (Organization name 機構和組織名)
        - npr (Other proper nouns 其他專有名詞)
    - ng (Common noun 普通名詞)
    - nvg (Nominalized verb 動名詞)
2. t (Time word 時間詞)
3. s (Place word 處所詞)
4. f (Position word 方位詞)
5. v (Verb 動詞)
    - General verb 一般動詞
        - vg (非謂語或帶動詞補語的動詞)
        - vgo (Intransitive verb 不帶賓謂語動詞)
        - vgn (帶體詞性賓語)
        - vgv (帶動詞性賓語)
        - vga (帶形容詞賓語)
        - vgs (帶小句賓語)
        - vgd (帶雙賓語)
        - vgj (帶兼語賓語)

- va (助動詞)
- vc (補語動詞)
- vi (系動詞)
- vy ("是"動詞)
- vh ("有"動詞)
- vv ("來","去"連謂動詞)

6. a (adjective 形容詞)
7. z (state word 狀態詞)
8. b (distinguishing word 區別詞)
9. Numeral 數詞
    - mx (系數詞)
    - mw (位數詞)
    - mg (概數詞)
    - mf (分數詞)
    - mb (倍數詞)
    - cc (數量詞)
    - mh (數詞"半")
    - mo (數詞"零")
10. Measure word 量詞
    - Nominal measure word 名量詞
        - qni (個體量詞)
        - qnc (集合量詞)
        - qnk (種類量詞)
        - qng (名量詞"個")
        - qnm (度量詞)
        - qns (不定量詞)
        - qnv (容器量詞)
        - qnf (成形量詞)
        - qnt (臨量詞)
        - qnz (准量詞)

- Verbal measure word 動量詞
  - qvp (專用動量詞)
  - qvn (名動量詞)

11. Pronoun 代詞
    - Nominal pronoun 體詞性代詞
      - rnp (general personal pronoun 人稱代詞)
      - rnd (demonstrative pronoun 指示代詞)
      - rnq (universal quantifier "每")
      - rna (quantifier "全部")
    - rp (謂詞性代詞)
    - rd (副詞性代詞)
12. Preposition 介詞
    - p (一般介詞)
    - pba (介詞 "把","將")
    - pbei ("被","讓","叫")
    - pzai ("在")
13. d (Adverb 副詞)
14. Conjunctive 連詞
    - 主從連詞
      - cf (主從連詞前段,"因爲","雖然"···)
      - cbc (連接分句,詞語)
      - cbs (連接句子)
    - 并連連詞,"和","與","並且"···
      - cpw (連接詞語)
      - cpc (連接分句)
      - cps (連接句子)
15. Particle 助詞
    - 結構助詞
      - usde ("的")
      - uszh ("之")

- ussi (“似的”)
- usdi (“地”)
- usdf (“得”)
- ussu (“所”)
- ussb (“不”)

- Aspect particle 動態助詞
  - utl (“了”)
  - utz (“著”)
  - utg (“過”)
- Other particle 其他助詞
  - upb (“被”)
  - upg (“給”)

16. y (Modal word 語氣詞)
17. o (Onomatopoeia 象聲詞)
18. e (Interjection 嘆詞)
19. h (Prefix 前綴)
    - hm (數詞前綴)
    - hn (名詞前綴)
20. k (Suffix 後綴)
21. i (Idiom 成語)
22. j (Abbreviation 簡稱語,如:“四化”)
23. l (Habitually used word 習用語,如:“總而言之”)
24. Others 其他
    - xch (非漢字字符串,“%”,“$”,$\cdots$)
    - xfl (Equation 數學公式)
    - xnull (“，”, “。”, “；”, “：”, “？”, “！”)
    - $ (beginning of a sentence)

Appendix B

Algorithms of Boundary Pairing Models

B.1 Heuristic based Model

Procedure (input: S) /* a sequence of POS tags corresponding to the input sentence */

Begin Declaration

Current Right = the current right boundary position;

Llist = list of left candidates;

Rlist = list of right candidates;

End Declaration

Begin Procedure

step 0 : Current Right = beginning of S.

step 1 : *Llist* = { all left candidates between 2 consecutive right candidates, i.e. between Current Right and the next one };

step 2 : L-PMAX = $pos(left_{max})$;
/* the position of the highest probability left candidate in *Llist* */
Last Left = $last(Llist)$; /* position of the last candidate in *Llist* */

step 3 : *Rlist* = { all right candidates between Last Left and the very

next left boundary candidate after Last Left};

step 4 : R-PMAX = $pos(right_{max})$;

/* the position of the highest probability right candidate in *Rlist* */

Last Right = $last(Rlist)$; /* position of the last candidate in *Rlist* */

step 5 : the left candidate at the position L-PMAX

and the right candidate at the position R-PMAX are paired;

step 6 : Current Right = Last Right;

$Llist = Rlist = \phi$;

Repeat from step 1 until the end of S;

End Procedure

B.2 Dynamic Programming based Model

1. find the possible boundary candidates:

   input part-of-speech sequence $t_1, t_2, \ldots, t_n$;

   for $(i = 0; i < n; i++)$ {

   find $p_l(i)$ and $p_r(i)$ for any two consecutive tags t_i and t_{i+1};

   forms two candidate sets $\mathcal{L}$ and $\mathcal{R}$ based on a pre-defined

   threshold and assigns them all the relevant statistic information.

   }

2. find the best partner pair:

   initialization: $score(0) = 1$, $pair(0) = 0$, $pre(0) = 0$;

   for $(i = 0; i < n; i++)$ {

   $j = 0$;

   while $(j \leq i)$ {

$k = 0$;

while $(k \leq j)$ {

$(k^*, l^*) = argmax(score(k) \times p_l(j) \times p_r(i)$

$\times (1 - p_l(k)) \times (1 - p_r(j)) \times \sum_{l=k+1}^{j-1} p_n(l))$;

$score(i) = score(k^*) \times p_l(j^*) \times p_r(i)$

$\times (1 - p_l(k^*)) \times (1 - p_r(j^*)) \times \sum_{l=k^*+1}^{j^*-1} p_n(l)$;

$k++$;

};

$j++$;

};

$pair(i) = j^*, pre(i) = k^*$;

for the part-of speech sequence $t_j, t_{j+1}, \ldots, t_i$, find (k^*, j^*) in the lower level recursively until there are no candidates between t_j and t_i, i.e. $\sum_{l=j}^{i-1} p_n(l) = 0$;

$score(i^*) = \sum_{l=1}^{m} score(i_l^*)$ (l is the m^{th} level of noun phrase)

$i++$

};

for $(i = 1; i < n + 1; i++)$

$i^* = argmax(score(i) * \sum_{l=i+1}^{n-1} p_n(l))$;

for $(i = i^*; i > 0; i = pre(i))$ {

insert "]" in the position $pos(i)$;

insert "[" in the position $pos(pair(i))$;

}

Appendix C

Triggering Environments of Transformation Templates

The following is an excerpt of all triggering environments within the range:

$$T_{b-2} \rightarrow T_{b+2} \quad and \quad T_{e-2} \rightarrow T_{e+2}$$

1. T_{b-2}
2. T_{b-1}
3. T_{b+0}
4. T_{e+0}
5. T_{e+1}
6. $T_{b+0}\ T_{e+0}$
7. $T_{b+0}\ T_{e+1}$
8. $T_{b-1}\ T_{b+0}$
9. $T_{b-1}\ T_{e+0}$
10. $T_{b-1}\ T_{e+1}$
11. $T_{b-2}\ T_{b+0}$
12. $T_{b-2}\ T_{b-1}$
13. $T_{b-2}\ T_{e+0}$
14. $T_{b-2}\ T_{e+1}$
15. $T_{b+0}\ T_{e+1}$

16. $T_{b+0}\ T_{e+0}\ T_{e+1}$
17. $T_{b-1}\ T_{b+0}\ T_{e+0}$
18. $T_{b-1}\ T_{b+0}\ T_{e+1}$
19. $T_{b-1}\ T_{3+0}\ T_{e+1}$
20. $T_{b-2}\ T_{b+0}\ T_{e+0}$
21. $T_{b-2}\ T_{b+0}\ T_{e+1}$
22. $T_{b-2}\ T_{b-1}\ T_{b+0}$
23. $T_{b-2}\ T_{b-1}\ T_{e+0}$
24. $T_{b-2}\ T_{b-1}\ T_{e+1}$
25. $T_{b-2}\ T_{e+0}\ T_{e+1}$
26. $T_{b-1}\ T_{b+0}\ T_{e+0}\ T_{e+1}$
27. $T_{b-2}\ T_{b+0}\ T_{e+0}\ T_{e+1}$
28. $T_{b-2}\ T_{b-1}\ T_{b+0}\ T_{e+0}$
29. $T_{b-2}\ T_{b-1}\ T_{b+0}\ T_{e+1}$
30. $T_{b-2}\ T_{b-1}\ T_{e+0}\ T_{e+1}$
31. $T_{b-2}\ T_{b-1}\ T_{b+0}\ T_{e+0}\ T_{e+1}$