Word Segmentation in Sentence Analysis

Andi Wu, email: andiwu@microsoft.com Microsoft Research
Zixin Jiang, email: jiangz@microsoft.com Microsoft Research

Abstract

This paper presents a model of language processing where word segmentation is an integral part of sentence analysis. We show that the use of a parser can enable us to achieve the best ambiguity resolution in word segmentation. The lexical component of this model resolves most of the ambiguities, but the final disambiguation takes place in the parsing process. In this model, word segmentation is a by-product of sentence analysis, where the correct segmentation is represented by the leaves of a parse tree. We also show that the complexity usually associated with the use of a parser in segmentation can be reduced dramatically by using a dictionary that contains useful information on word segmentation. With the aid of such information, the sentence analysis process is reasonably fast and does not suffer from the problems other people have encountered. The model is implemented in NLPWin, the general-purpose language understanding system developed at Microsoft Research. A demo of the system is available.

Keywords: Word segmentation, sentence analysis, parsing, dictionary, ambiguity resolution.

Introduction

As a prerequisite for natural language processing in Chinese, automatic word segmentation by computer has received a great deal of attention both in the academic world and the corporate world. Many different versions of “word breakers” have been produced and most of them claim accuracy of 98% or above. While this might be good enough for some applications of language technology, it still fails to meet the requirement of natural language understanding where every *sentence* must be analyzed correctly. The syntactic analysis of a sentence requires perfect word segmentation. A single error in segmentation will cause a whole sentence to receive a wrong analysis or no analysis. Given a page of document with 500 words and 99% accuracy in word segmentation, the 1% error rate will result in 5 wrong segmentations which in turn may cause 5 sentences to be mis-analyzed. For the purpose of sentence analysis, therefore, the accuracy rate should be the percentage of *sentences* that are corrected segmented (Yeh and Lee 1991).

Errors in word segmentations occur mainly for two reasons: ambiguity and unlisted words. In this paper, we will concentrate on ambiguity resolution. The problem of unlisted words will be discussed in a separate paper, though the model presented here will also provide a good basis for the identification of unlisted words. In what follows, we will show how the use of a parser can enable us to achieve the best performance in ambiguity resolution. We will also show that the complexity or inefficiency usually associated with the use of a parser can be reduced dramatically if we use a dictionary that has useful information on word segmentation.

Why use a parser?

The idea of using a parser for word segmentation is by no means new (see Gan 1995 and Li 1997 as recent examples), though very few people have put it into serious implementation. The advantage of having a parser in ambiguity resolution is obvious: we can base our decisions on the "understanding" of the whole sentence rather than on some local contexts. While most segmentation ambiguities can be resolved locally at the lexical level, there are cases where we have to look at the structure of the sentence to make a decision. This is why the maximal-matching (MM) algorithm (e.g. Chen and Liu 1992) succeeds in most cases but fails completely in some cases. Statistical approaches (e.g. Sproat et al 1996) try to take a broader context into consideration by finding the best path through the whole sentence. This often helps, but the best path is not guaranteed to be the correct path. It can avoid some of the mistakes made by the MM algorithm, but it can also produce errors that would not occur with the MM algorithm. What is common to these approaches is that they try to make decisions too early without sufficient information. The MM approach lacks global information and statistical approaches lack structural information. The use of a parser, on the other hand, can provide us with both global information and structural information.

Consider the following two sentences.

(1) 什么时候我才能克服这个困难？
(2) 在这些企业中国有企业有十个。

Sentence (1) contains a local combinational ambiguity involving the character sequence 才能 which can be either 才能(Noun) or 才(Adv) + 能(Verb). The MM algorithm will mistakenly treat 才能 as a single word in this sentence, whether we use forward maximal matching or backward maximal matching. Statistical approaches mignt succeed in identifying the sequence as two words if 才能 happens to occur next to the certain words, but the success is far from guaranteed. If we parse the sentence, however, we will be able to find a successful parse **only if** 才能 is analyzed as 才(Adv) + 能(Verb).

Sentence (2) contains a local overlapping ambiguity that involves the character sequence 中国有 which can be either 中 + 国有 or 中国 + 有. Most existing word breakers try to resolve such ambiguity by comparing the probabilities of the overlapping pair and choosing the word that is more frequent. While this can lead to right decisions in many situations, it is clearly the wrong heuristic to use in this sentence. We see that the correct segmentation here is 中 + 国有 in spite of that fact that the frequency of 中国 is much higher than 国有. Again, the use of a parser will enable us to make the correct decision in this case, since the sentence cannot be successfully analyzed if the sentence is segmented as 在 + 这些 + 企业 + 中国 + 有+ 企业+ 有 + 十 + 个.

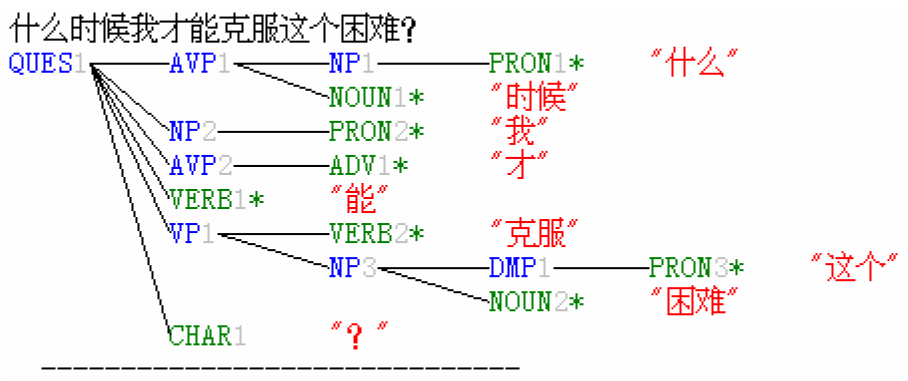
The ambiguities we have considered so far are all *local* ambiguities, which disappear once we look at the whole sentence. There are also *global* ambiguities, which do not disappear at the sentence level, thus resulting in two different readings of the sentence. Sentence (3) is such an example.

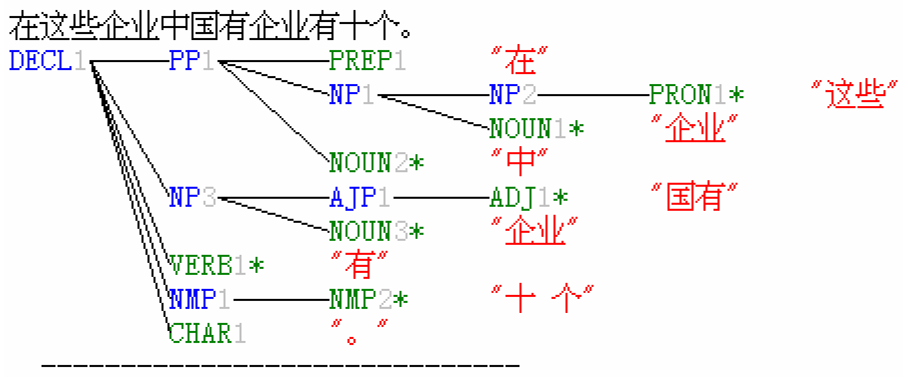
(3) 国防部组织部内的人做这一工作。

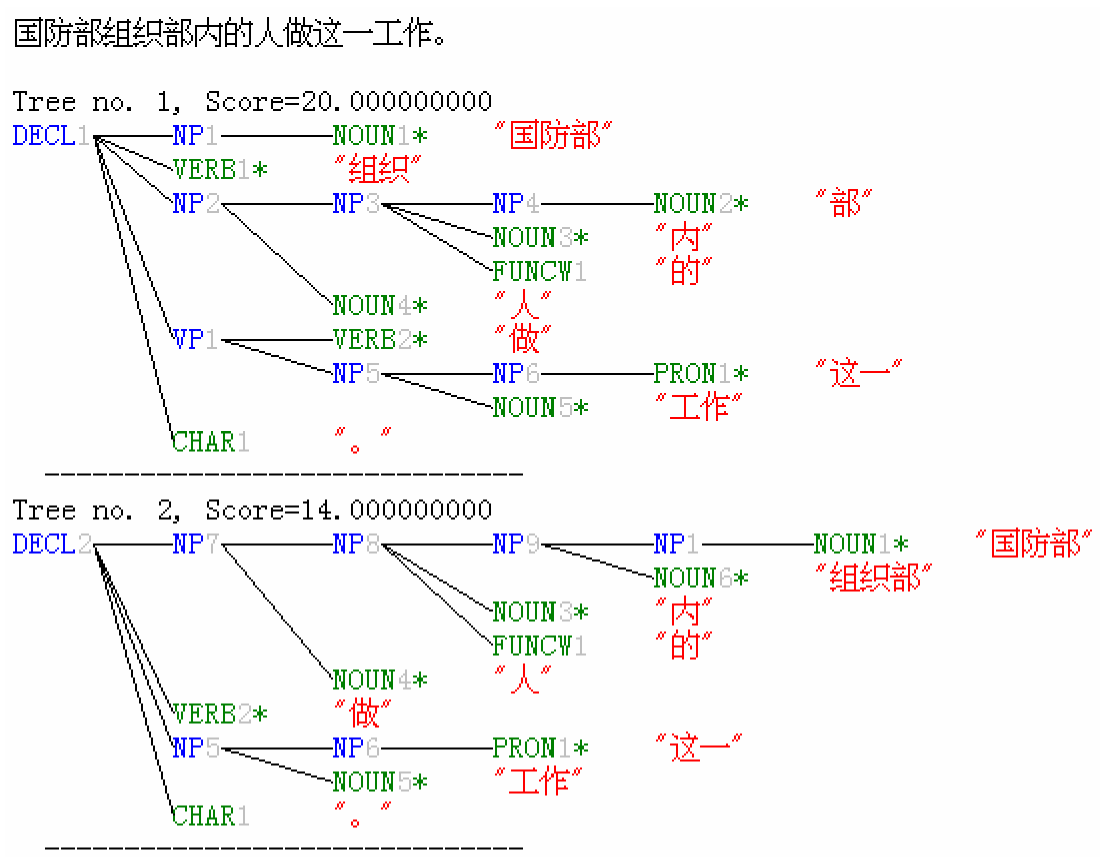
This sentence means "The Defense Department organizes the people in the department to do this job" if 组织部 is analyzed as two separate words: 组织 + 部 . When 组织部 is analyzed as one word, however, it means "The people in the organization department of the Defense Department do this job". While most existing word breakers can get one of those readings only, the use of a parser can give us both readings. We can then select the intended reading on the basis of discourse context.

The main argument against the use of a parser in word segmentation is that it is “inefficient” or “too expensive”. This is a valid argument if our goal is to build a stand-alone word breaker that does nothing but word segmentation and if we do not expect the accuracy to be perfect. Pattern matching definitely takes much less processing time than parsing. However, if our goal is to build a natural language understanding system where a parser is required in the first place, not to take advantage of the parser in word segmentation would be a mistake. In such a system, word segmentation is not the final result, but a by-product of sentence analysis. Since the system we are developing at Microsoft Research (NLPwin) is a general-purpose language understanding system and we already have an efficient parser in the system, we choose to make word segmentation an integral part of the system where some segmentation ambiguities are resolved in the parsing process.

The following are the parse trees of (1), (2) and (3) produced by NLPwin. The correct segmentation is represented by the leaves of the parse trees. Notice that Sentence (3) has two trees, each corresponding to one of the readings.







```
国防部組织部内的人做这一工作。

Tree no. 1, Score=20.000000000
DECL1──NP1────NOUN1*    "国防部"
     ──VERB1*  "组织"
     ──NP2────NP3────NP4────NOUN2*    "部"
                  ──NOUN3*  "内"
                  ──FUNCW1  "的"
          ──NOUN4*  "人"
     ──VP1────VERB2*  "做"
          ──NP5────NP6────PRON1*    "这一"
                  ──NOUN5*  "工作"
     ──CHAR1      "。"
  ------------------------------
Tree no. 2, Score=14.000000000
DECL2──NP7────NP8────NP9────NP1────NOUN1*    "国防部"
                        ──NOUN6*    "组织部"
                  ──NOUN3*  "内"
                  ──FUNCW1  "的"
          ──NOUN4*  "人"
     ──VERB2*  "做"
     ──NP5────NP6────PRON1*    "这一"
          ──NOUN5*  "工作"
     ──CHAR1      "。"
  ------------------------------
```

When to use the parser?

The fact that we are using a parser does not mean that *all* segmentation ambiguities are resolved in the parsing process. As most people have discovered, many disambiguation decisions *can* be made at the lexical level. In such cases, leaving the ambiguities to the parser will not make sense. The complexity of parsing is exponential and any additional ambiguity can contribute to its explosion. This is the very reason why most people have considered the use of a parser infeasible. To make this approach feasible, we must try to disambiguate as much as possible at the lexical level, leaving to the parser only those ambiguities whose resolution requires syntactic information.

To avoid missing any possible analysis of a sentence, our system must look up every possible word in the sentence. This will give us more words than we need. Take sentence (1) again as an example. The initial lookup will recognize the following words from the dictionary.

(4)

```
什么 PRON
    时候 NOUN
    时 NOUN
       候 VERB
```

```
         我 PRON
          才能 NOUN
          才 NOUN
          才 ADV
            能 VERB
              克服 VERB
              克 NOUN
                服 VERB
                  这个 PRON
                  这 PRON
                       个 NOUN
                       困难 ADJ
                       困难 NOUN
                       困 VERB
                       困 ADJ
                         难 ADJ
```

If we were to pass all these words to the parser, the parsing chart would be filled with useless sub-trees and the analysis would be extremely inefficient. It is obvious that some of the words in (4) cannot possibly be legitimate words in this sentence. For example, although 克 is a word (as in 五百克豆油)and 服 is also a word (as in 他刚服了药) , they cease to be independent words when 克 is immediately followed by 服, forming a single word 克服. Similarly, 困 and 难 are not words in 困难, and 时 and 候 are not words in 时候. The resolution of such combinational ambiguities requires lexical information only. Therefore, our system eliminates those single character words before parsing begins. The words that the parser sees are only the following:

(5)

```
什么 PRON
    时候 NOUN
         我 PRON
          才能 NOUN
          才 NOUN
          才 ADV
            能 VERB
              克服 VERB
                  这个 PRON
                    困难 ADJ
                    困难 NOUN
```

Now the only combinational ambiguity that has to be resolved in the parsing process is 才能 vs. 才 + 能. This way the additional complexity that word segmentation contributes to parsing is reduced to the minimum.

When to ignore a word?

The question that arises naturally at this point is how we decide when to ignore the single character words contained in a multiple character word. Why do we ignore 困 and 难 in 困难 while keep 才 and 能 in 才能? In our system, such decisions are based upon the lexical information that is stored in the dictionary entry of each word. For each of the multiple character word in our dictionary, there is a binary-valued attribute called "*Atomic*". A word can have its component words ignored if the value of *Atomic* is 1 but must keep its component words if the value is 0. Apparently, the value of this attribute is 1 for 困难 and 0 for 才能. In the actual segmentation process, we use an augmented version of the MM algorithm. The main difference between this version and the standard version is this. After finding a word ***W*** from Position ***i*** to Position ***j*** in the sentence, the standard MM algorithm will move the pointer to ***j+1*** and start matching from that position. In our version, however, this happens only if *Atomic* is set to 1 in ***W***. Otherwise, we will match the strings from Position ***i*** to Position ***j-1, j-2,*** etc. until we find a word. This ensures that the sub-words contained in a non-atomic word will all be recognized. To give a more intuitive description of the difference, we can say that the standard MM algorithm behaves as if *every* multiple character word has *Atomic* set to 1 while our system does things differently depending upon the value of *Atomic*. As a result, we are able to resolve the ambiguities that standard MM can correctly resolve and leave to the parser those ambiguities which standard MM is unable to resolve correctly.

The operation described above is complicated by the fact that two +*Atomic* words can overlap. Take (6) as an example.

(6) 这块肉的确切得好。

Both 的确 and 确切 are marked as +*Atomic* in our dictionary. However, there will be a problem if we remove all the single character words covered by these two words. When 的, 确 and 切 are gone, we will miss the word 切 if 的确 is the correct word in the sentence and miss the word 的 if 确切 is the correct word. In either case we will not be able to parse the sentence, because none of the "paths" through the sentence is left unbroken. To prevent this from happening, we require that all the single character words in a multiple character word be retained regardless of the value of *Atomic* except the word(s) covered by the overlapping part. In the case of 的确切, we will keep 的 and 切 but remove 确 which is in the intersection of 的确 and 确切. The resulting word lattice of (6) will therefore be the following.

(7)
```
这 PRON
  块 NOUN
    肉 NOUN
      的确 ADV
      的 FUNCW
        确切 ADJ
        切 VERB
          得 VERB
          得 FUNCW
            好 ADJ
            好 ADV
```

Postpone it if not ignored

So far we have discussed the elimination of certain words in cases of combinational ambiguity. We can also ignore certain words in cases of overlapping ambiguity. Consider the following sentence.

(8) 中国以新的姿态出现在世界的东方。

There are three instances of overlapping ambiguity here: 出现 vs. 现在, 现在 vs. 在世, and 在世 vs. 世界. We can see that 现在 and 在世 are not words in this sentence and it is desirable not to make them visible to the parser. When we take a closer look at these two words, we find that they represent two very different cases. In the case of 在世, we can make a disambiguation decision on a very local basis: it cannot be a word here because it is immediately followed by 界. Given the string 在世界 in a sentence, it is almost certain that the words are 在 and 世界 while the possibility 在世 + 界 is almost zero. We can therefore safely ignore 在世 without knowing the rest of the sentence. The situation with 现在 is totally different, however. Given the string 出现在, 出现 and 现在 have equal likelihood. Although 现在 is not a word in Sentence (8), it is certainly a word in Sentence (9), in spite of the fact that it also appears next to 出.

(9) 你绝不能说出现在的方案。

The ambiguity involved in 出现在 is thus not locally resolvable. We must look at the whole sentence to make a decision.

In our system, cases like 在世界 and cases like 出现在 are handled differently. The ambiguity in 在世界 is resolved locally at the lexical level. We simply eliminate 在世 when it is followed by 界. In the case of 出现在, we have to keep both possibilities and let the ambiguity be resolved in parsing. However, statistical data tells us that the probability of 出现 + 在 is higher than that of 出 + 现在. A smart parser should consider 出现 + 在 first. The possibility of 出 + 现在 should not be considered unless no successful analysis can be found using 出现 + 在. To achieve this effect, we assign high probability to 出现 and low probability to 现在 in 出现在, since our parser considers words with higher probabilities first.

Now the question is how we know when to ignore or assign low probability to a word in an overlapping pair. This information is again stored in the dictionary. Many multiple character words in the dictionary have the following lists in their entry.

- The LeftCond1 list. The word in this entry will be ignored if it is immediately preceded by one of the characters in this list in a sentence.
- The RightCond1 list. The word in this entry will be ignored if it is immediately followed by one of the characters in this list in a sentence.
- The LeftCond2 list. The word in this entry will be assigned low probability if it is immediately preceded by one of the characters in this list in a sentence.
- The RightCond2 list. The word in this entry will be assigned low probability if it is immediately followed by one of the characters in this list in a sentence.

For instance, the character 界 is in the RightCond1 list of 在世, and the character 出 is in the LeftCond2 list of 现在. As a result, 在世 will be removed in (8) and 现在 will be set to a low probability in (8). The words that the parser can see is (9), where 现在 is shaded to indicate that it has low probability and therefore its use in the parsing process will be postponed.

(9)

```
中国 NOUN
    以 PREP
       新 ADJ
       新 ADV
         的 FUNCW
           姿态 NOUN
                出现 VERB
                出 VERB
                  现在 NOUN
                  现 ADV
                     在 PREP
                       世界 NOUN
                           的 FUNCW
                             东方 NOUN
```

Coverage and performance

All the strategies described above have been implemented in NLPWin, the general natural language processing system of Microsoft. The Chinese grammar already covers most of the structures and we are beginning to parse unrestricted texts.

We have a dictionary of over 80,000 words, not including derived words, compounds and proper names, which are built on-line in our system. Almost all these words have been marked (automatically or by hand) for the attributes of *Atomic, LeftCond1, RightCond1, LeftCond2 and RightCond2*.

To find out how much we have improved over other systems, we collected about 100 sentences that are claimed to be difficult for some systems. Many of those sentences come from papers on word segmentation and they are usually the problematic cases. A sample of these sentences is given in the appendix. When we processed them with our system (without special tuning for those sentences), 85% of them received a good parse (which also means correct word segmentation, of course). For cases where we failed to find a successful parse, about half of them involve a wrong segmentation. If we use the usual measure for word segmentation, i.e. the number of correctly segmented words divided by the total number of words, the accuracy is well above 99%.

The system is fairly efficient considering the fact that it is doing full dictionary lookup and full parsing. On a Pentium 200 PC, it is able to analyze 20 sentences per second, the average length of the sentences being 30 characters. For shorter sentences like those in the appendix, the speed is about 45 sentences per second.

A demo of the system will accompany the presentation of this paper.

More advantages with the parser

When we make word segmentation part of sentence analysis, we can solve problems other than the ones discussed so far. As we know, the most difficult task in Chinese word segmentation is the identification of unlisted words. We can use various heuristics at the lexical level to guess, for example, which character string forms a name. These guesses are not perfect and we are bound to make mistakes if our decisions are based on lexical information only. In our system, the final decision is made in the parsing process. The lexical component is only responsible for making good *guesses*. In other words, the lexical component only has to propose candidates that meet the *necessary* conditions for being a personal name, a place name, a newly coined word, etc. The sufficient conditions are provided by the parser. Consequently, we are able to correctly recognize most of the unlisted words. The details of this work will be presented in a separate paper.

In conclusion, the use of a parser can put us in a position to achieve a higher accuracy in word segmentation than approaches that have access to lexical information only. The parser can be fairly efficient if we can use lexical information to prune the search space before parsing begins, though the final decision is made by the parser.

Appendix

他有各种才能。
什么时候我才能克服这个困难？(Sproat et al 1996)
他将来上海。(Yao et al 1990)
将来的上海会有严重污染。(Yao et al 1990)
这名记者会说国语。(Yeh and Lee 1991)
记者会将于五点钟开始。(Yeh and Lee 1991)
这匹马路上病了。(Sproat et al 1996)
我马上就来。(Yao et al 1990)
他从马上下来。(Yao et al 1990)
这些企业中国有企业有十个。
物理学起来很难。(Yao et al 1990)
物理学是一门基础科学。(Yao et al 1990)
研究生一般年龄较大。(Yao et al 1990)
他在研究生命起源。(Yao et al 1990)
这块肉的确切得好。(Yao et al 1990)
他的确切地址在这儿。(Yao et al 1990)
他们经常州去上海。
他们上台北去了。
她本人生了三个孩子。(Gan et al 1996)
他在北京住了三十来年.
我们要学生活得有意义。(Gan et al 1996)
这位职员工作的压力很大。(Gan et al 1996)
国防部组织部内的人做这一工作。(Wu & Tseng 1993)
美国会采取这种政策。(Wu & Tseng 1993)
中国已开发和尚未开发的资源都很多。(Sproat et al 1996)
记叙文本来应以叙事为主。
日文章鱼怎么说？(Sproat et al 1996)
中国以新的姿态出现在世界的东方. (Li 1997)
我们要发展中国家用电器。
发展中国家的电器很便宜。
我们有机会见面。(Bai 1995)

你应该考虑到来年的收成。
你应该努力学会计划时间.
每八个人中就有一人死于怀孕或分娩。
(Guo 1996?)

它的模范行为让群众看到了共产党人为理想而献身的精神。(Guo 1996?)

References

Bai, Shuanhu. 1995. An integrated model of Chinese word segmentation and part of speech tagging. In Liwe Chen and Qi Yuan, editors, *Advances and Applications on Computational Linguistics*. Tsinghua University Press, pages 56-61.

Chen, Keh-Jiann and Shing-Huan Liu. 1992. Word identification for Mandarin Chinese sentences. In *Proceedings of 14th International Conference on Computational Linguistics (COLING'92)*, pages 101-107. Nantes, France.

Gan, Kok-Wee. 1995. *Integrating Word Boundary Disambiguation with Sentence Understanding*. Ph.D. dissertation, Department of Computer Science and Information Systems, National University of Singapore.

Gan, Kok-Wee, Martha Palmer, and Kim-Teng Lua. 1996. A statistically emergent approach for language processing: appliation to Modeling Context Effects in Ambiguous Chinese Word Boundary Perception. *Computational Linguistics*, 22(4): 531-553.

Guo, Jin. 1996? A comparative Experimental Study on English and Chinese Word Boundary ambiguity.

Li, Wei. 1997. *CPCG: A Lexicalized Chinese Unification Grammar and Its Application*. Ph.D. dissertation in progress, Department of Linguistics, Simon Fraser University, Canada.

Sproat, Richard, Chilin Shih, William Gale and Nancy Chang 1996. A Stochastic Finite-State Word-Segmentation Algorithm for Chinese. *Computational Linguistics*, Volume 22, Number 3.

Wu, Zimin and Gwyneth Tseng. 1993. Chinese text segmentation for text retrieval: Achievements and Problems. Jounal of the Americal Society for Information Science: 44(9):532-542.

Yao, Tian-Shun, Gui-Ping Zhang, and Ying-Ming Wu. 1990. A rule-based Chinese automatic segmentation system. *Journal of Chinese Information Processing* 4(1):37-43.

Yeh, Ching-Long and His-Jian Lee. 1991. Rule-based word identification for Mandarin Chinese Sentences – a unification approach. In *Computer Processing of Chinese & Oriental Languages*, 5(2):97-118.