USeg: A Retargetable Word Segmentation Procedure for Information Retrieval

Jay M. Ponte and W. Bruce Croft
Computer Science Department
Amherst, MA 01003-4610, USA

Abstract

Many languages, such as Chinese, are written without interword delimiters. For these languages, a segmenter is required as a pre-processing step for information retrieval systems. We describe USeg, a platform for word segmentation designed to fulfill the requirments imposed by the information retrieval task. USeg is based on an underlying probabalistic automaton which serves as a simple language model. A description of the proposed model(s), implementation issues for these models and experimental results are presented. The experiments show that a fairly simple underlying model can produce reasonable segmentation results, can do so quickly enough to be useful for indexing in an information retrieval system and can be re-targeted to new languages without a great deal of human effort.

1 Introduction

The word segmentation problem consists of finding the word boundaries in text where they are not marked (by whitespace for example). Many languages such as Chinese and Japanese are written this way and often a sequence of characters can be grouped in several ways, making segmentation a non-trivial problem. Figure 1 shows an example of Chinese segmentation. The three characters individually mean "lower", "middle" and "country".

The segmentation on the left, meaning "lower-middle country" is probably not as good as the one on the right, which cooresponds to "lower China" .

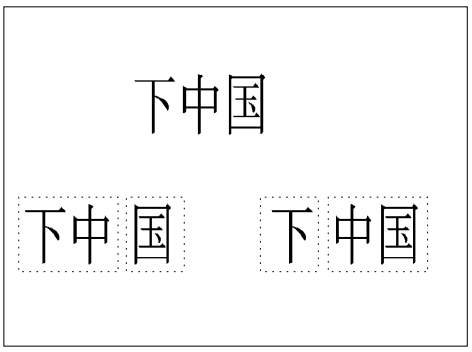

Figure 1: An Example of Chinese Segmentation.

USeg was designed to work in the context of the INQUERY information retrieval system [3] . In order for it to be a useful tool it needs to satisfy the following requirements:

- Retargetability – The INQUERY system is designed to be retargetable to different languages. The segmenter must also be retargetable to different languages with minimal effort and without the need for expensive resources such as large amounts of hand segmented text. The original segmenter was written for Chinese and then was retargeted to Japanese and also to (corrupted) English text to test retargetability of the module.
- Speed – INQUERY's indexing procedure works at a rate of hundreds of megabytes

per hour. A segmenter that cannot run at similar speeds is not useful.

- Accuracy – Clearly the segmenter should be as accurate as possible, but experiments need to be done to determine the exact effects of segmenter accuracy on retrieval. For retrieval purposes, it remains to be seen whether certain types of errors are more tolerable than others or whether absolute accuracy is the quantity to maximize.

This paper contains some preliminary results comparing USeg's Chinese output to hand segmented Chinese text and some results using English text. A second measure of performance would be a comparison of USeg to other segmenters in the context of information retrieval by measuring the retrieval effectiveness using the standard metrics of recall and precision. This is left for future work.

In order to fulfill the above requirements, the underlying model for USeg is a probabilistic automaton. This approach is easily retargetable compared, for example, to a language specific rule based system. In addition, a good implementation of this model can satisfy the speed requirements. Finally, the parameters of the model can be adjusted from training data in order to improve accuracy. Two different types of probabilistic automata were used for the underlying model. The first is a simple word based model suggested by Barnett [9] and the second is a word bigram model which can be thought of as a Markov model [1]. The resources required to build a segmenter for a new language are a lexicon and some unsegmented text for training. Some experiments that did not require a lexicon were also performed.

2 Previous Work

For an overview of Chinese segmentation for information retrieval see [4]. Sproat *et al* [8] describe a Chinese word segmentation algorithm based on probabilistic automata. Their approach includes special recognizers for Chinese names and transliterated foreign names and a component for morphologically derived words. Our approach is to develop the above components as post processors to the segmenter. This leaves the segmenter as a more general module and allows it to be used alone when such recognizers are not needed.

Barnett [9] uses a word based model for segmenting Japanese. The model contains word frequencies and the input is segmented by adding up the frequencies of possible words and then subtracting out a per word cost. Additional complexity arises from the inflectional endings. In order to consider a candidate word, morphological processing is done to find every possible grouping of characters that form valid words. This complication does not arise in Chinese to the same degree.

Chang *et al* [6] describe a method of Chinese lexical acquisition using a small seed corpus, a word based segmenter, and a two class classifier for words. N-grams of size 2, 3, and 4, were used as candidates in the initial lexicon. The word probabilities were updated by training the word based segmenter on a small segmented seed corpus. The two class classifier used n-gram frequency, mutual information and left and right entropy as features in a linear function. The weights for each feature were trained on the seed corpus.

A Chinese segmenter was developed at New Mexico State University (NMSU) as part of the "Norm" project [10]. For each character in the input string, a list of candidate words is considered. The words are compared to the input string and the ones that match are aligned to determine the segmentation. This segmenter was used to produce a file which was then hand corrected and used as a test file in the experiments for this paper. A side by side comparison of this segmenter with USeg was also done for both speed and accuracy.

3 Models for Segmentation

As mentioned earlier, experiments were done using two different underlying models which can be characterized as word based and bigram based. Both models use dynamic programming but they maximize different functions. The two models are now presented in more detail.

4 The Word Based Model

4.1 Initialization

The word based model is initialized by building a suffix tree of words in the lexicon. A root node is associated with each word initial character. All suffixes (in the string-matching sense, not the linguistic sense) are represented as subtrees. For example, given a trivial lexicon consisting of three words, "A", "AB" and "B", A three node graph will be built as in figure 2.

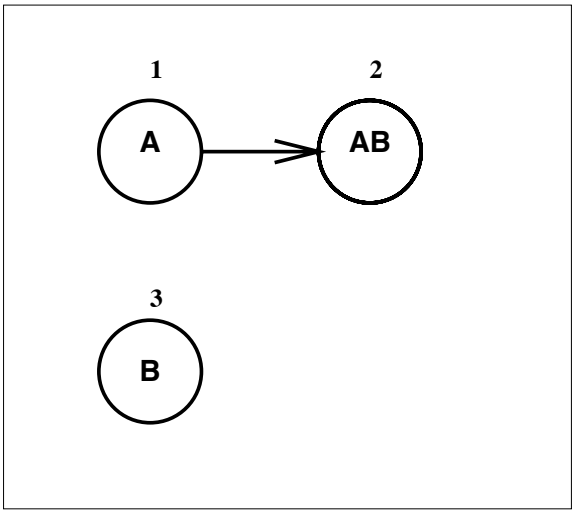

Figure 2: A simple suffix tree.

Node 1 is a root node for the character "A", an accept state for the single character word "A", and has node 2, the accept state for "AB", as a one node sub-tree. Node 3 is the start and accept state for the word "B". Notice that each node encodes the entire sub-string of the current word. Structures of this type can be used to perform string matching on a large number of strings at once.

Each word in the lexicon is assigned a probability estimate based on a longest match (greedy) segmentation of the training text. The result is a very simple language model expressed as a probabilistic automaton.

4.2 Training and Segmenting

Once the model has been initialized, the parameter estimates can be refined. As mentioned, the initial estimates are obtained by doing a greedy segmentation of the training data. A single pass is made over the training data and the longest match is used for segmentation. The words are counted along the way to obtain the initial probability estimates.

Using the simple example from above and the following string as the single training sequence, "ABABAABB", initial probabilities can be estimated as follows. The training sequence is ambiguous with respect to the model but the longest match segmentation is the state sequence 12121123 corresponding to a segmentation that looks like this: "AB AB A AB B". From this segmentation, the words "A", "AB" and "B", will be assigned estimated probabilities of 0.2, 0.6 and 0.2 respectively.

The initial model can then be used to segment the training data via dynamic programming. Again, a single pass is made over the data, but this time two tables are filled in. The first table will contain the probabilities of each possible segmentation. When there is more than one path leading to a node the one with the highest probability is used. The second table contains the predecessor of each node used to fill in the probability table. When the end of the sequence has been reached, this table is used to backtrack over the states in order to get the best path through the model. When the text has been segmented in this fashion, the probability estimates can be updated by keeping a count variable for each word in the lexicon.

5 The Bigram Model

5.1 Initialization

The bigram model is slightly more complex. A state graph is built with a sequence of nodes for each word in the lexicon, one node per character. Next, a single pass is made over the training data to obtain initial probability estimates. There are two sets of probabilities to be estimated. The initial state distribution is estimated by counting the number of times each word starts a sequence. In the case of ambiguities, every word that could be completed is given credit. Then, each possible path is taken through the graph and possible interword edges are added/updated as they are encountered. The resulting model has estimates for

the first word in each sequence and estimates for the conditional probabilities of seeing word $n+1$ given that word n was just seen. Note that the "initial" model is already "trained" in that non-occurring edges are never added and the initial estimates are obtained from the training data.

5.2 Training and Segmenting

The algorithms for training the bigram model and segmenting with it are similar. For ease of discussion, they are presented in reverse order, segmenting first followed by training.

Given the automaton representing word bigrams, segmenting some free text means finding the best path through the model. The Viterbi algorithm accomplishes this [1]. In order to uncover the maximum likelihood state sequence for a given output sequence, all possible paths through the model must be considered along with their probabilities. Clearly, the number of paths is exponential in the length of the sequence. However, since the subsequences overlap, this problem lends itself to solution by dynamic programming. A table is filled in such that at each time-step, the probability of a given state will be the maximum of the product of path probabilities from the previous step and the transition probability from each of the states at the (current) end of the paths. At the same time a backtracking table can be filled in to "remember" the path. The complexity of this algorithm will then be linear in the length of the input sequence and linear in the number of edges in the model or, in general, linear in the square of the number of states. However, for this problem, the number of edges to be considered at each time step is a function of the current state and the next character so the number of edges that actually need to be considered is very small compared to the more general case.

The model is trained using the Baum-Welch algorithm, a special case of Expectation Maximization [1]. Like the Viterbi algorithm, it uses dynamic programming. In this case, two tables are filled in, one table of probabilities propagated forward in time and a second set propagated backward. Instead of using the maximum likelihood path, the sum of all paths is computed at each step. So, where the Viterbi algorithm calculates the probability of the maximum likelihood path for a sequence given the model, the Baum-Welch algorithm calculates the probability of the sequence (over all possible paths) given the model. Since this is done in both directions, the two tables contain the probabilities of getting to each point in the sequence and the probabilities of finishing the rest of it for each state at each time-step. The edge probabilities are updated incrementally based on the probability of traversal until the model converges to a local maximum [1].

6 Implementation Issues

6.1 Very Sparsely Populated Matrices

The transition matrices are very sparsely populated since the model graphs are connected in a very constrained fashion. In addition, the outputs for each state are limited, so at each time step, there will be very few potential transitions. The algorithms are implemented to take advantage of these facts.

6.2 Suffix Tree Implemented as a Hash Table

When each character is read, the only edges that need to be considered are those that come from the present node(s) and which go to a node with the new character. Essentially, the structure needed for this task is a suffix tree. Each node should have all of its edges available by the character of the next node. However, the alphabet size is large and there is considerable variance in the number of edges out from each node. Because of this, a single hash table is used to look up edges by current node and next character, effectively implementing a suffix tree via hash lookups. For the Baum-Welch algorithm, a second hash table of pointers hashed by node and previous character is used for the backward pass. The probability and backtracking tables will also be very sparsely populated, so they are implemented as hash tables to conserve memory.

For the word based model, no interword edges are explicitly represented. The model graph is considered to be fully connected. Upon reaching a word end, the start node of words beginning with the next character will be added

to the list of candidate nodes.

6.3 Sorting

When the potential edges for a step have been found, they need to be sorted in order of the second node (the 'to' node) in order to calculate the table entries. Since there are generally very few edges (under ten), a simple in line shell sort is used to avoid the overhead of a more complicated sort.

6.4 Scaling

As mentioned in [1], there is a potential problem in the probability calculations. Since at each step, numbers significantly less than one are being multiplied together, the values will approach zero at an exponential rate, quickly exceeding the precision of the machine. There are two methods of getting around this problem. As mentioned, the calculations for the Viterbi algorithm proceed in the following way. At each time-step, the value for each possible node is obtained by multiplying the probability of the path of maximum value from the previous time-step by the values of the edges into the node being calculated. Since there is no summing, this calculation lends itself to logarithmic computation. Rather than using the edge probabilities, the log probabilities are used, so that at each step it is a simple matter of adding the logarithms. This replaces floating point multiplies with floating point adds and solves the problem without doing any extra calculations.

The forward and backward calculations for the Baum-Welch algorithm are a little different. In this case, the sum of the probabilities over all paths into a node is used. Since we are summing, we will use a scaling factor rather than using logarithmic computation. At each time step, all of the values are increased by a scaling factor which is calculated as the sum of all of the values added at that time step. In the re-estimation phase, these scaling factors will cancel out yielding the correct results.

7 Experiments in Chinese

7.1 Available Text

Two large collections of unsegmented text were available for training. The first set consisted of 8.6 MB of text from the "Xinhua" newswire. The second set was 132 MB total, 105 MB of which was made up of of Chinese characters. This text was a superset of the first with the additional text coming from the "People's Daily News".

A third set, consisting of 61 KB of text segmented using the NMSU segmenter and corrected by hand, was used for testing. The output file was compared to the hand corrected file using the metrics of recall and precision. Recall is measured to be the percentage of words in the hand corrected file that were correctly identified by the segmenter. Precision is the percentage of words identified by the segmenter that occurred in the hand corrected file (in the corresponding position). These metrics provide a reasonable indication of performance, however they do not measure the quality of segmentation with respect to retrieval performance nor do they account for the fact that even native speakers of Chinese do not always agree on the correct segmentation [8].

Segmenter	Recall	Precison
Word Based	87.80	84.40
Perfect Lexicon	93.63	95.87
Bigram	86.77	80.64
NMSU	87.53	80.78

Table 1: Recall and precision results.

As mentioned earlier the models are initialized from a Chinese lexicon. Two different Chinese lexicons were tried. The first was the lexicon from the NMSU segmenter consisting of 42197 words [10]. This is a small to medium lexicon made up of common words, proper nouns and idioms. The second consisted of all and only the words that occurred in the test data. This was done to measure the contribution of the lexicon to performance since it is effectively a perfect lexicon.

As shown in Table 1, the bigram segmenter performs about as well as the NMSU segmenter. The word based segmenter is approximately the same in terms of recall and 3.6% better in terms of precision. This result was little bit surprising at first since one would expect bigrams to be a better language model. The short answer to why the word model works better than the

bigram model is that the bigram model has too many parameters to estimate from the available training data. The bigram model contains 85,882 interword edges out of a possible 1.8×10^9. One might expect that there would be many important edges that did not occur in the training data at all and that is exactly what happened. Comparing the output of the two models on the test data, there are 216 differences. All of these differences are caused by the same problem, the bigram model puts in a break prematurely.

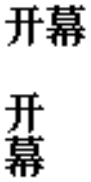

Figure 3: Example of an Error Made by the Bigram Model.

In figure 3, the word model produced the correct segmentation on top while the the bigram model has broken these characters into two words. The reason for the break is that the bigram model does not have an edge from the previous word into the start state of the correct word because no character sequence corresponding to that edge occurs in the training data. As a result the bigram segmenter never considers the correct word. One way around this problem is to smooth the probability distribution using a small default weight for the non-occuring edges. This has the effect of fully connecting the bigram model. This workaround fixes the problem in figure 3 and while the total net result is slightly better than the original bigram model, it still is not as good as the word model. Since we are guessing at this point, the weight of the non-occuring edges can also be set randomly. The result, again, is somewhere in between the original bigram model and the word model. When there is no estimate for an edge, we can guess and sometimes get lucky, giving us better performance than setting the non-occurring edges to zero. Any edge that is not estimated reliably is a potential source of errors for the bigram model and, with the huge number of possible edges, this problem outweighs the benefits we might expect from the finer grained representation.

One way to obtain a better language model without the huge number of parameters is to place words into equivalence classes (part of speech tags being the obvious choice), but the performance of the word model using the ideal lexicon suggests that this should not be the top priority.

Even if a better model could be estimated reliably, the lexicon was, by far, the most important factor. Using a near perfect lexicon consisting of all and only the words that occurred in the hand corrected test set, the recall goes up to 93.63% and the precision to 95.87%. This is clearly cheating and in a real application, such a lexicon will certainly not be available. The point of the experiment was to see if the segmentation errors were due to the two major inherent limitations of the model. The first limitation is that the probability estimates are far from perfect. This is a much bigger problem with the bigram model. The second is that these simple automata are not perfect language models. The word model, in particular, is a very poor way to model a natural language. The "perfect lexicon" experiment shows that the limitations of the models were not as critical as the limitations of the lexicon. This suggests that automatically acquiring a good lexicon for the word based segmenter would be more important than using a more complex model. An additional reason to acquire the lexicon from the training text is that many modern words, such as words used for concepts in science and technology, are very different in Chinese as spoken in Mainland China and in Taiwan [7].

7.2 Automatic Lexical Acquisition From Text

The lexical acquisition process was done in a manner similar to [6]. Statistics were collected for n-grams of size 2, 3, and 4 since most Chinese words are of those lengths. The frequency of occurrence, mutual information and entropy were the statistics used [6].

Informally, an n-gram is assumed to be a word if it:

- Occurs frequently

- Has a high degree of mutual information – i.e. The characters occur together more often than would be expected by chance.
- Has a high degree of entropy with its surroundings – i.e. If the characters that surround the n-gram tend to have a high degree of randomness.

These three measures were combined in a linear classifier function.

The results, 60.55% recall and 60.21% precision, show that the automatically acquired dictionary does not seem to work as well as the manually constructed one. Nevertheless, it is encouraging since many of the selected n-grams that were not words turned out to be either subsequences of words or frequently occurring word pairs [7]. While such non-words are not desirable for many natural language tasks, it is possible that they would work well for the purpose of information retrieval since they have good statistical properties.

7.3 Segmentation Speed

Timing figures were collected for USeg and the NMSU segmenter for comparison purposes. Both segmenters were run on a DEC AlphaStation 250 with no competing processes and with the data on a local disk.

Segmenter	Time
USeg	42 sec.
NMSU	6 min. 46 sec.

Table 2: Timing figures for 8.6 MB Xinhua collection.

As shown in table 2, USeg is faster than the NMSU segmenter by an order of magnitude on the 8.6 MB Xinhua collection. More importantly, USeg operates at reasonable speeds for indexing in an information retrieval environment. An additional timing run for USeg was done on the larger 132 MB collection and the result was 8 min. 16 sec., a linear increase in execution time with respect to collection size.

7.4 Error Analysis

Figure 4 shows some common errors. The top segmentation in each example is from the hand-corrected file and the bottom is from Useg.

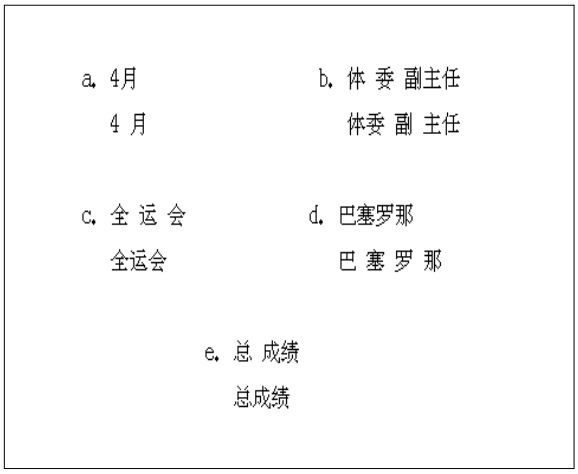

Figure 4: Examples of segmentation differences.

Figure 4-a is a simple problem, the top segmentation is correct and means the month of April. This is a trivial error to fix within the segmenter and could also be handled as part of a special purpose date recognizer. In figure 4-b, the output from USeg is better than the "correct" version. The first two characters mean "body" and "committee" respectively and together they refer to a specific organization, a ministry of fitness. The last three characters mean "assistant director" and can be segmented either way. Figure 4-c is another win for USeg, the three characters form a proper noun that refers to a national sporting event. In figure 4-d, the opposite happened, these four characters together are a transliteration of "Barcelona." USeg failed to recognize this because it did not occur in the lexicon. To correct this type of error, a special recognizer for transliterated foreign names could be used. Without such a recognizer, this strategy of breaking up unsegmentable sub-sequences is reasonable for retrieval purposes since it provides full coverage of the data. The example in figure 4-e could be segmented either way. The three characters mean "total score."

8 Preliminary Work in Japanese

Segmenting Japanese seems to be a harder problem due to the inflectional endings. Verbs and adjectives are given inflectional endings to indicate their function in the sentence as compared to an un-inflected language where much of this information is deduced from the syntax. The current Japanese version of INQUERY uses the JUMAN morphological analyzer for segmentation. JUMAN employs sophisticated morphological analysis to handle the inflectional endings [5]. In principle, the underlying model of USeg is general enough to incorporate morphological rules, but it would require considerable human effort. Instead, our approach was to use the lexical acquisition procedure from the Chinese experiments. The procedure was the same, but a larger window size was used to account for the longer average word length in Japanese. The resulting segmenter does not produce a valid segmentation from a natural language perspective, but it will segment text into pseudo-words with good statistical properties. Our intention is to test the performance of the segmenter for information retrieval purposes by doing a set of experiments that compare retrieval performance using INQUERY on the Japanese Tipster data with USeg and with the JUMAN segmenter. Due to time constraints, this has been left for future work, however, it seems that the model can be retargeted easily enough. The proof will be in the retrieval performance.

9 Experiments in English

Recently, the question was posed to us whether similar segmentation methods could be used for English text that had been corrupted with respect to whitespace. This may also be useful for automated transcription of Morse code or a similar task. The results are interesting because they show the effects of a large amount of segmented training data. For English, only the word based model was used. The word probabilities were estimated from 1 Gigabyte of data from the Tipster collection [3]. The lexicon consisted of 558,238 words obtained by "dumping" INQUERY's inverted index file. This produces a list of all of the words in the collection along with frequency information. The list was filtered by throwing away any word that occurred less than one hundred times. The probability estimates for those that were left were calculated from the term frequency scores.

This lexicon is much larger than the lexicon used in the Chinese experiments, however, the probability estimates are very good due to the volume of data and the fact that it is correctly segmented. The segmenter was tested on a 500 KB of text from the Wall Street Journal. Recall for this task was 93.56% and precision was 90.03%. Even though the lexicon was very large, the availability of good probability estimates allows the word based segmenter to perform quite well.

---

the unit of New York-based Loews Corp that makes Kent cigarettes stopped using crocidolite in its Micronite cigarette filters in 1956.

theunitofNewYork-basedLoewsCorpthatmakesKentcigarettesstoppedusingcrocidoliteinitsMicronitecigarettefiltersin1956.

the unit of New York-based Loews Corp that makes Kent cigarettes stopped using c roc id o lite inits Micron it e cigarette filters in 1956.

Figure 5: An example using English text.

Figure 5 shows a sentence fragment from the original text at the top, the same text with whitespace removed in the center, and the same text again, segmented with USeg, at the bottom (linebreaks in center were inserted to make the text fit). The words "crocidolite", a form of asbestos and "Micronite", a type of cigarette filter did not occur in the training text. The non-word "inits" was present in the original text as were many combinations of short words such as "ofa" and "ofthe". Most of these non-words were removed in the filtering step. For example, "ofa" occurred 67 times and "ofthe" occurred 60 times while "inits" occurred 172 times and so was not filtered out. The present cutoff value of one hundred was chosen in an admittedly ad hoc fashion for this experiment, but since ample training data is available for this task, the cutoff value could be set in a more principled manner

to optimize performance.

10 Conclusions and Future Work

The basic methodology and implementation of USeg fulfill the requirements as a component of an information retrieval system. The segmenter runs at reasonable speeds for text indexing and is retargetable with minimal effort. The accuracy of the Chinese version is comparable to a competing segmenter.

The Chinese experiments showed that a simpler word based model worked better than a more complex bigram model. The major reason for this is that the much larger parameter space of the bigram model makes it difficult to get reliable estimates even from a fairly large training corpus. One way around this would be to place the words into equivalence classes such as part of speech in order to make use of a better language model without the huge number of parameters. However, the Chinese experiments also showed that even more important than the underlying language model is the lexicon. So far, attempts to acquire a lexicon from unsegmented text have not yielded very good segmentation results but it remains to be seen whether this approach would be useful in the context of information retrieval because non-words might still be reasonable document features if they have good statistical properties.

The preliminary Japanese experiments are encouraging because of the ease of retargeting the segmenter but not much more can be said at this point. In the near future, when comparisons of the Japanese (and Chinese) segmenter(s) can be done in the context of retrieval, a good indicator of performance will be available. The English experiments show that if a large amount of segmented text is available, a lexicon along with good parameter estimates are easy to obtain and a simple segmentation model can work quite well.

In the future, the effect of the segmenter on retrieval performance needs to be investigated. Depending on the results of those experiments, some refinement of the lexical acquisition technique may need to be done. The next method to try will be to start with a reasonable lexicon, and analyze only the portions of text where the segmenter fails.

Acknowledgments

Thanks to Hideo Fujii for supplying native speaker intuitions for the Japanese experiment. Thanks to Chengfeng Han for additional help with Japanese. Special thanks to Jinxi Xu for doing likewise for the Chinese experiments and for his patience. Thanks to James Callan for reviewing an early draft of this paper and for many helpful discussions. Thanks to Jim Barnett for his very valuable suggestions. Thanks to Mike Crystal for providing the hand corrected text. Finally, special thanks to the Chinese text processing group at the New Mexico State University Computing Research Lab for the use of their segmenter.

References

[1] Rabiner, L.R. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE* vol. 77, no. 2, Feb. 1989.

[2] King, S.F. *Syntactic Pattern Recognition and Applications* Prentice Hall, 1982

[3] Broglio, J., Callan, J. P., and Croft, W. B. INQUERY System Overview *Proceedings of the TIPSTER Text Program (Phase I)* San Francisco, CA. Morgan Kaufman, 47-67.

[4] Wu, Z., Tseng, G. Chinese Text Segmentation for Text Retrieval Achievements and Problems. JASIS, Oct, 1993.

[5] Matsumoto, Y., Kurokashi, S., Myoki, Y., User's Guide for the JUMAN System – A User-Extensible Morphological Analyzer for Japanese. Nagao Lab, Kyoto University, 1991.

[6] Chang, J.S. , Lin Y. C. and Su, K. Y. Automatic Construction of a Chinese Electronic Dictionary. *Proceedings of the Third Workshop on Very Large Corpora* June 1995.

[7] Xu, Jinxi. Personal Communication.

[8] Sproat, R. Shih, C. , Gail, W. and Chang, N. A Stochastic Finite State Word Segmentation Algorithm for Chinese.

[9] Barnett, J. Personal Communication.

[10] Jin, Wanying. A Case Study: Chinese Segmentation and its Disambiguation. NMSU CRL Memo Number MCCS-92-237, 1992.