Chinese text segmentation with MBDP-1: Making the most of training corpora

Michael R. Brent and **Xiaopeng Tao**
Department of Computer Science
Campus Box 1045
Washington University
St. Louis, MO 63130-4899
{xptao, brent}@cs.wustl.edu

Abstract

This paper describes a system for segmenting Chinese text into words using the MBDP-1 algorithm. MBDP-1 is a knowledge-free segmentation algorithm that bootstraps its own lexicon, which starts out empty. Experiments on Chinese and English corpora show that MBDP-1 reliably outperforms the best previous algorithm when the available hand-segmented training corpus is small. As the size of the hand-segmented training corpus grows, the performance of MBDP-1 converges toward that of the best previous algorithm. The fact that MBDP-1 can be used with a small corpus is expected to be useful not only for the rare event of adapting to a new language, but also for the common event of adapting to a new genre within the same language.

1 Introduction

Many languages, including Chinese and Japanese, are written without spaces or other delimiters between the words. A word segmentation algorithm is therefore required as a front end for any language procesing system that relies on a word-based representation. Most systems for parsing, indexing, document retrieval, spell checking, and grammar checking fall into this category. As a result, the text segmentation problem has received considerable attention, particularly for Chinese. A variety of methods have been investigated. One approach is based on looking up strings in a pre-existing dictionary and using the longest match (Cheng et al., 1999). A second approach is based on bigram frequencies or more general substring frequencies (Dai et al., 1999; Ponte and Croft, 1996; Teahan et al., 2000). A third approach uses transformation-based learning (Hockenmaier and Brew, 1998; Palmer, 1997). A review of earlier work can be found in (Wu and Tseng, 1993). All of these methods require either a pre-existing dictionary or else a supervised training regimen using a manually segmented corpus.

Dictionary-based methods have the well-known advantages and disadvantages of all knowledge-intensive approaches. Manually curated linguistic knowledge tends to be more accurate than what can be gleaned by an adaptive learning system, especially when handling relatively rare cases. On the other hand, it tends to give insufficient weight to the common cases and suffers from a lack of adaptability to new languages, genres, and applications. Since this paper is concerned primarily with adaptability, we will focus on the best available methods that do not require a pre-existing dictionary.

PPM is an adaptive text compression algorithm that has been applied to the segmentation problem (Teahan et al., 1998; Teahan et al., 2000). Many text compression algorithms, including PPM, work by estimating a probability distribution on the next symbol in a text given the previous context. Distributions estimated from a text that includes word-boundary delimiters assign a probability to word-boundary delimiters in each context. If an unsegmented text is viewed as having hidden word-boundary delimiters, a Viterbi-style algorithm can be used to find the most probable locations of the hidden delimiters, according to the estimated probability model. Teahan and colleagues have done this with the PPM probability model. The result appears to be the best available algorithm for segmentation of large corpora of written text, both in English and in Chinese. PPM requires supervised training using a manually segmented corpus.

The MBDP-1 algorithm (Brent, 1999a) was developed as a model of how children segment speech in the course of learning their native languages. Because speech contains no known acoustic marking of word boundaries, children must segment the utterances they hear in order to learn the

words of their language. Further, children start out without knowing any words and without access to a presegmented speech sample of the sort that would be required for supervised training. Thus, MBDP-1 — the algorithm underlying an abstract cognitive model known as INCDROP, for INCremental Distributional Regularity OPtimization (Brent, 1999b; Dahan and Brent, 1999) — requires neither a dictionary nor a segmented training corpus. It bootsraps its own dictionary, which is initially empty, using a probability model and Viterbi-style optimization algorithm. In this paper, we show that MBDP-1 is useful for text segmentation in both English and Chinese.

The remainder of the paper is organized as follows. The next section describes the probability model underlying INCDROP and the objective function that results from the model. Section 3 describes the optimization algorithm that MBDP-1 uses to find the most probable segmentation, according to the model. Section 4 reports experiments in which MBDP-1 is compared to PPM (Teahan et al., 2000) using the PH corpus of Chinese newspaper text and the English portion of the Hansard corpus. Finally, Section 5 considers the broader implications of this work.

2 Generative Probability Model

This section introduces a language-independent prior probability distribution on all possible segmented texts. Given an unsegmented text T, this prior distribution defines a conditional distribution on all segmented texts that yield T after word-boundary deletion. The conditional distribution determines the most probable segmentation of T, according to the model. The prior distribution is derived from a five-step model for the generation of texts. The steps are presented below, along with their probability distributions. This section describes a mathematical model, not an algorithm that is intended to be run.

In the following generative model, let Σ be the alphabet or character set of the text to be segmented, and let # and \$ be reserved symbols that are not in Σ. These symbols will be used to represent word boundaries and sentence boundaries, respectively.[1] After describing each step of the generative procedure, we provide a sample result that could be produced from that step. Combining all five sample results yields a sample segmented text that can be generated by this model.

1. Choose the number of distinct word types in the text, n, according to the distribution:

$$\Pr(n) = \frac{6}{\pi^2}\left(\frac{1}{n}\right)^2 \quad (1)$$

The inverse-squared distribution on the positive integers was chosen because it is a simple, smooth distribution that is relatively flat, representing a relatively unbiased prior, yet its sum converges (unlike the sum of $1/x$). The $6/\pi^2$ term normalizes the sum to one.

Sample result: $n = 6$.

2. Let $\Sigma' = \Sigma \cup \{\#\}$ be the character set together with the word-boundary marker, and let $\{p_1 \ldots p_{|\Sigma'|}\}$ be a probability distribution on Σ'. For each i from 1 to n, generate word type i by choosing characters at random, according to the probabilities $\{p_1 \ldots p_{|\Sigma'|}\}$, until the word boundary character # is chosen. If the word boundary character is chosen first, discard it and choose again until a non-boundary character is chosen, ensuring that the word has at least one such character. Call the resulting word type W_i, and let $L = \{W_1 \ldots W_n\}$ be the resulting lexicon. For notational convenience, let $W_0 =$ \$, the reserved sentence-boundary marker. To compute the probability of a given lexicon, we estimate probabilities of the characters using add-one smoothing:[2]

$$\hat{p}_l = \frac{c_l}{S}$$

where c_l is one greater than the frequency count of letter l in the currently hypothesized segmentation, and $S = \sum_l c_l$. If L is a set of word types that are all chosen independently, then

$$\Pr(L \mid n) = n!\left(\frac{1}{1-\frac{c_\#}{S}}\right)^n \prod_{l\in\Sigma'}\left(\frac{c_l}{S}\right)^{c_l} \quad (2)$$

The product at the right of (2) represents the probability of selecting the *sequence* of word types $W_1 \ldots W_n$, in that particular order, when multiple word boundaries (#'s) are permitted to occur in sequence. The center term on the right hand side of (2) results from imposing the constraint a word cannot begin with #. The $n!$ term reflects the fact that L is an unorderd set, so any permutation of the n words is permissible.

[1] While word boundaries are not marked in the text to be segmented, sentence boundaries are.

[2] As described below, we treated each byte of each Chinese character as a separate character, so the alphabet for MBDP-1 has just 2^8 characters.

Sample result:

$W_1 = do\#$ $W_2 = the\#$ $W_3 = kb\#$
$W_4 = like\#$ $W_5 = see\#$ $W_6 = mbo\#$
$W_0 = \$$

3. For each i from 1 to n, choose $f(i)$, the frequency of word W_i in the text, according to the inverse-squared distribution on the positive integers:

$$\Pr(f(i) = k) = \frac{6}{\pi^2}\left(\frac{1}{k}\right)^2 \qquad (3)$$

Sample result:

$f(1) = 2$ $f(2) = 4$ $f(3) = 2$ $f(4) = 1$
$f(5) = 2$ $f(6) = 2$ $f(0) = 2$

4. Let $m = \sum f(i)$ be the total number of word tokens. Choose an ordering function $s : \{1, \ldots, m\} \rightarrow \{1, \ldots, n\}$ that maps each position in the text to be generated to the index of the word that will appear in that position. Thus, word type $W_{s(j)}$ appears as the jth word in the text. Note that s is constrained to map exactly $f(i)$ positions to word type W_i. Choose s according to the uniform distribution on the distinct orderings (there are only finitely many, given n, L, and f):

$$\Pr(s|n, L, f) = \frac{\prod f(i)!}{(\sum f(i))!} \qquad (4)$$

Sample Result:

$s(1) = 1$ $s(2) = 3$ $s(3) = 5$ $s(4) = 2$
$s(5) = 6$ $s(6) = 0$ $s(7) = 5$ $s(8) = 2$
$s(9) = 2$ $s(10) = 0$ $s(11) = 1$ $s(12) = 3$
$s(13) = 4$ $s(14) = 2$ $s(15) = 6$

Define w_j to be $W_{s(j)}$, the jth word in the text. Define $\mathbf{w}_k = w_1 \ldots w_k$, the concatenation of the first k words of the text. Note that $\mathbf{w}_k$ is a text in which word boundaries are marked by #.
Example:

$$w_1 = W_{s(1)} = W_1 = do\#$$
$$w_2 = W_{s(2)} = W_3 = kb\#$$
$$w_3 = W_{s(3)} = W_5 = see\#$$
$$\vdots$$

Example:

$$\mathbf{w}_2 = do\#kb\#$$

5. Delete the #'s from $\mathbf{w}_m$ and output the result. The output is a text in which the word boundaries are not marked, like the texts to be segmented. This is a deterministic process, so the unique possible outcome has probability 1.0.
Sample output:

dokbseethembo$seethethe$dokblikethembo

The probability with which steps 1-4 generate a particular segmented text $\mathbf{w}_m$ is simply the product of equations (1)-(4). The conditional probability that steps 1-4 generated a segmented text $\mathbf{w}_m$, given the unsegmented text T resulting from step 5, is proportional to the marginal probability if T can be obtained by deleting the word boundaries from $\mathbf{w}_m$; otherwise it is zero.

The probability of $\mathbf{w}_m$ resulting from steps 1-4 can be factored by defining R (for *relative probability*) as follows:[3]

$$R(\mathbf{w}_k) = \frac{\Pr(\mathbf{w}_k)}{\Pr(\mathbf{w}_{k-1})},$$

where $\Pr(\mathbf{w}_0)$ is defined to be 1. Now

$$\begin{aligned}\Pr(\mathbf{w}_k) &= R(\mathbf{w}_k)\Pr(\mathbf{w}_{k-1}) \\ &= \prod_{i=1}^{k} R(\mathbf{w}_i) \qquad (5)\end{aligned}$$

Hereafter, we focus on $R(\mathbf{w}_i)$.

If w_k has occurred previously ($w_k \in \{w_1, \ldots, w_{k-1}\}$) we say w_k is a *familiar word*; otherwise, we say it is a *novel word*. Brent (1999a) showed that if w_k is a familiar word that occurs $f(w_k)$ times in $\mathbf{w}_k$ then

$$R(\mathbf{w}_k) = \frac{f(w_k)}{k}\left(\frac{f(w_k) - 1}{f(w_k)}\right)^2 \qquad (6)$$

Note that the final occurrence in position k is included in $f(w_k)$, so $f(w_k) \geq 2$ for a familiar word and thus (6) is never zero. The first term on the right hand side of (6) is the relative frequency of the word so far, with one added to both the numerator and the denominator (since the occurrence in position k is included). This term is similar to the familiar maximum likelihood estimate for a parameter of a multinomial distribution on words. The second term can be thought of as an adjustment for the fact that the observed

[3] Algebraically, the relative probability can be treated as the conditional probability of the kth word given the first $k-1$ words, but the semantics is different.

relative frequency of a word tends to overestimate its asymptotic relative frequency, especially when it has occurred only a few times. (Intuitively, if a word has occurred only once, there are probably a lot of other words that are just as frequent in the long run but happen not to have been observed in the available sample.) However, (6) was derived directly from the probability model, without any consideration of relative frequency or adjustments to it.

If w_k is a novel word whose character sequence is $a_1 \ldots a_q$ then

$$R(\mathbf{w}_k) = \frac{6}{\pi^2} \cdot \frac{n}{k} \cdot \left(\frac{n-1}{n}\right)^2 \cdot \frac{p_{a_1} \ldots p_{a_q}}{1 - p_{\#}}, \quad (7)$$

where n is the number of distinct word types in $\mathbf{w}_k$ and $p_{a_1} \ldots p_{a_q}$ are the estimated probabilities of the characters in w_k. The second term of (7) — n/k — is the type-to-token ratio, or the relative frequency with which new words have occurred in the past. It makes sense that the probability of novel words would be higher if they have occurred frequently in the past than if they have occurred rarely in the past. The third term can be thought of as an adjustment factor to the second term, reflecting the fact that the relative frequency with which novel words have occurred in the past will tend to overestimate the asymptotic relative frequency when the sample is small.[4] The last term, which generally dominates, corresponds to the probability that a particular novel word will happen to be spelled $a_1 \ldots a_q$. This term favors novel words constructed out of common characters over novel words constructed out of rare characters and favors short words over long words, all other things being equal.

The most likely segmentation of a text, according to the model, is the one that maximizes (5), the product of the relative probabilities.

3 Optimization Algorithm

The MBDP-1 algorithm segments one input sentence at a time, freezing the segmentation of each sentence before the next sentence is read in. Each sentence is segmented so as to maximize the product of the relative probabilities of words in the segmentation. The relative probabilities are computed using (6) and (7), assuming that the all previous sentences in the corpus were segmented correctly. In addition, the relative probability of each word in a sentence is based only on the words in the previously segmented sentences, not on other words in the same sentence. Put differently, the relative probability of each word is calculated as though it were the first word in the sentence. Given these assumptions, MBDP-1 finds the optimal segmentation of a sentence using a Viterbi-style dynamic programming algorithm that computes the relative probability of every sub-string of the sentence. This algorithm can be visualized as a graph structure that we call a trellis, by analogy to the trellis used in Viterbi decoding of HMMs. This is illustrated in Figure 1 for the input "sentence" consisting of the single word *only*. Nodes represent the potential word boundaries between adjacent input characters, an arc (or straight line segment) from node j to node k represents the potential word between nodes j and k, and a path from one end of the trellis to the other represents a potential segmentation.

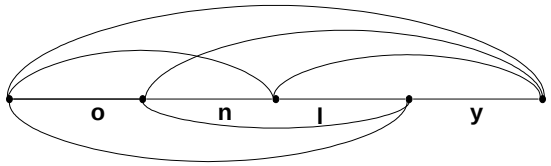

Figure 1: The segmentation trellis for the "sentence" consisting of the single word *only*. Arcs between adjacent nodes are drawn as straight line segments.

For each node k, from the left edge of the sentence to the right, MBDP-1 computes the relative probability of each impinging arc that starts to the left of k, under the assumptions described above. It multiplies this arc probability by the product of relative probabilities along the best path to the node where the arc starts. Finally, it takes the maximum over all impinging arcs and stores the result as the product of relative probabilities along the best path to node k.

$$\text{BestPathProb}[k] = \max_{j=0}^{k-1}(\text{BestPathProb}[j] * \text{RelProb}[j,k])$$

where BestPathProb[0] is set to 1. Only arcs within the current sentence are considered, since words cannot span sentence boundaries. At each node, pointers to the previous node along the best path are stored; tracing back through these arcs yields the best path.

It is important to emphasize that, while the search for a good segmentation is limited to one sentence at a time, the probability calculation is based on the entire corpus processed so far. That is, the determination of whether a word is novel

[4] Equation (7) is applicable only after the first sentence has been segmented, guaranteeing $n \geq 2$. For the first sentence, (1)-(4) must be used directly.

or familiar, its frequency so far, the total number of words so far, and the estimated probabilities of all the letters, are based on the best segmentations found for all previous sentences in the corpus. Formally, if $\mathbf{w}_i$ is the sequence of words resulting from the segmentation of all previous sentences in the corpus and $\sigma_{j,k}$ is the string that lies between nodes j and k in the current sentence, then

$$\text{RelProb}[j,k] = R(\mathbf{w}_i \circ \sigma_{j,k}),$$

where $\circ$ is the concatenation operator and R is the function defined by (6) and (7).

Normally, MBDP-1 is initialized to a naive state in which all words are novel and all letters have observed frequency 0. However, using MBDP-1 with a manually segmented training corpus could not be simpler — the training corpus is simply appended to the test corpus as a prefix, with its segmentation frozen in advance. Thus, when the first test sentence is processed, the training corpus plays the role of $\mathbf{w}_i$ in the last equation.

4 Experiment

Based on published results, MBDP-1 appears to be the most effective known algorithm for segmentation of phonemically transcribed spontaneous speech by mothers to young children (Brent, 1999a; Brent, 1999b). However, it was designed as a computational model of how children segment speech in the course of acquiring their native language, an application that does not permit the use of manually segmented training texts. Further, written text has very different characteristics from spontaneous child-directed speech, even in an alphabetic writing system. Text written in Chinese characters is even more remote from the corpora on which MBDP-1 had been tested. Thus, we had no idea how MBDP-1 would perform on Chinese text segmentation after being trained on a manually segmented text. In the following experiment, we applied MBDP-1 to the segmentation of (1) the PH Corpus of Chinese newspaper text, and (2) Part A of the English portion of the Hansard corpus, which is a sample of the official proceedings of the Canadian parliament. We also tested PPM, which is apparently the best known algorithm for this problem (Teahan et al., 2000), on the same corpora.

4.1 Input

The first corpus we used is Guo Jin's Mandarin Chinese PH corpus, containing more than one million words of newspaper stories from the Xinhua news agency of PR China written between January, 1990 and March, 1991. This manually segmented corpus is represented in the standard GB coding scheme, which uses two bytes for each Chinese character. Following the procedure used by Teahan et al. (2000), we treated each byte as a separate input symbol for both MBDP-1 and PPM. Thus, it is possible for either algorithm to insert a word boundary between the two bytes of a Chinese character.

We chose the first one million words of PH as a training corpus and the following 16,000 words as a test corpus. The two algorithms were trained on subsets of the overall training corpus whose sizes varied from 2^2 up to 2^{20} words, not counting punctuation. The test corpus was divided into 16 samples of 1,000 words each so we could assess the variance in performance across samples of a given genre. The test corpus was preprocessed to remove all spaces and create separate "sentences" at punctuation marks, which are used in Chinese approximately as in English.

The second corpus we used was Part A of the English portion of the Hansard corpus, which contains a sample of the proceedings of the Canadian parliament. We extracted training and testing samples exactly as for the PH corpus, but since the Hansard is larger we were able to investigate training samples of up to 2^{22} words.

4.2 Results

We evaluated the results by applying the standard measures *precision* and *recall* to pairs of consecutive word boundaries (not to individual word boundaries). To compute these measures, each character of the automatic segmentation is aligned with the corresponding character of the standard segmentation. Each word in the automatic segmentation is labeled a true positive if it lines up exactly with a word in the standard segmentation– that is, both boundaries match. Each word in the automatic segmentation that does not align exactly with a word in the standard segmentation is labeled a false positive. Each word in the standard segmentation that does not align exactly with a word in the automatic segmentation is labeled a false negative. For example, if the test sentence is "ABCAEDF", the standard segmentation is "A BC AED F", and the automatic segmentation is "A BC A ED F", then the true positives are 'A', 'BC', and 'F'. The false positives are 'A' (the second one) and 'ED'. 'AED' is a false negative.

Using these terms, we define precision and recall as follows:

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \quad (8)$$

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \quad (9)$$

Precision is the proportion of the machine-segmented words that are right. Recall is the proportion of words in the standard segmentation words that are identified by the algorithm. These two measures can diverge, and good performance is achieved only when both are high.

Figure 2 shows the results on the Chinese PH corpus, as a function of the logarithm of the training corpus size. The left panel shows precision and the right shows recall. Data points for MBDP-1 are disks, data points for PPM are triangles, and error bars span two standard errors of the mean. The results show that MBDP-1 has better recall than PPM on the PH corpus until the training corpus reaches 2^{20} words; at 2^{20} words the two algorithms are statistically indistinguishable.[5] MBDP-1's precision is significantly better than PPM when the training size is 2^{10} words or less. After that the two become statistically indistinguishable.

Interestingly, PPM occasionally inserts a word boundary between the two bytes of a Chinese character, whereas MBDP-1, although it could segment between bytes, never does.

Figure 3 shows the results on the English Hansard corpus, as a function of the logarithm of the training corpus size. The format is the same as in Figure 2. The results show that MBDP-1 has reliably better recall than PPM on the Hansard corpus, except when the training corpus size is between 2^{12} words and 2^{16} words, where the two algorithms are statisticaly indistinguishable. MBDP-1 has better precision, too, for both large and small training corpora. The two algorithms are tied at training sizes of 2^{10} and 2^{16}, and PPM has greater precision between those points. At very large training corpus sizes both algorithms perform extremely well.[6]

[5]For the smallest training corpus sizes MBDP-1 will learn as it tries to segment the test corpus, so performance should be better than shown here on larger test corpora. For the training corpora above 2^{16} words performance is already so good that this effect is negligeable.

[6]PPM has one free parameter, the order of the model. All experiments in this paper use order 3, which Teahan et al. (2000) describe as giving the best results overall. Although Teahan et al. do not discuss adaptation of the order, we have implmented an adaptation scheme and found that it generally chooses order 5 for large English corpora. This improves the results for large corpora by about two percentage points, making them statistically indistinguishable from the results of MBDP-1.

4.3 Discussion of the Experiment

In this discussion we focus on the results in Chinese, since text segmentation has no real application in English. Considering precision and recall together, it appears that MBDP-1 performs much better when the available training corpus is smaller than 2^{12} words, somewhat better when the training corpus is between 2^{12} words and 2^{18} words, and indistinguishably when the training corpus is 2^{20} words.

There is no simple, complete explanation for the fact that MBDP-1 outperforms PPM with small training corpora. However, it is worth keeping in mind that MBDP-1 is able to hypothesize the existence of words even when it has no training corpus, recognize them when they occur in later utterances, and segment them out. It does this, in part, by making very good use of sentence boundaries and other punctuation. Initially, when it has little or no experience, MBDP-1 tends to treat entire sentences (or other punctuation-bound phrases) as single words, storing them in a list of familiar words. Occasionally, phrases do in fact consist of only one or a few words. Such short phrases are likely to occur again embedded in longer phrases. When they do, they tend to be segmented out, leaving the remaining contiguous segments of the phrase to be treated as though they were separate phrases. This leads to the isolation and storage of more words. For example, the one-word sentence *Look!* would generally be interpreted as a single word and stored in the list of familiar words. If it occurred again in the phrase *Lookhere!*, it would tend to be segmented out, leaving *here* to be treated as though a separate phrase.

PPM, on the other hand, is not based on hypothesizing words but rather on estimating the probability of word *boundaries* in various contexts. Since word boundaries appear only in segmented training text, PPM does not learn from unsegmented text. The fact that it can learn only from the training corpus and not from exposure to unsegmented text may be one reason that it requires a larger training sample.

The training and test materials for this experiment, while distinct, came from exactly the same source. The performance of both algorithms can be expected to deteriorate as the training and test corpora diverge in genre.

5 General Discussion

The process of adapting a natural language processing algorithm to a new language holds great theoretical interest. In general, algorithms that can be adapted automatically and cheaply are to

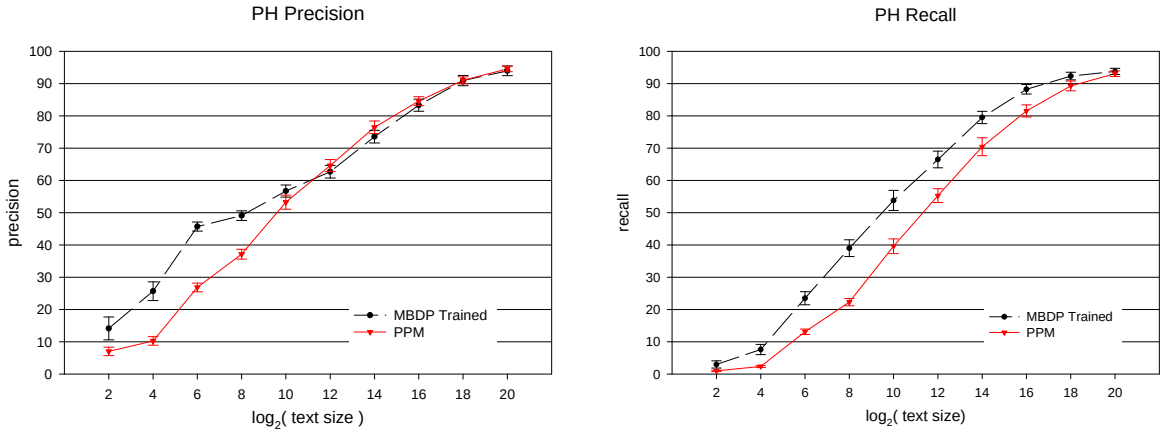

Figure 2: Precision and recall on a sample the Chinese PH corpus as a function of $\log_2$ training corpus size: MBDP-1 trained and PPM trained. Error bars are two standard errors of the mean.

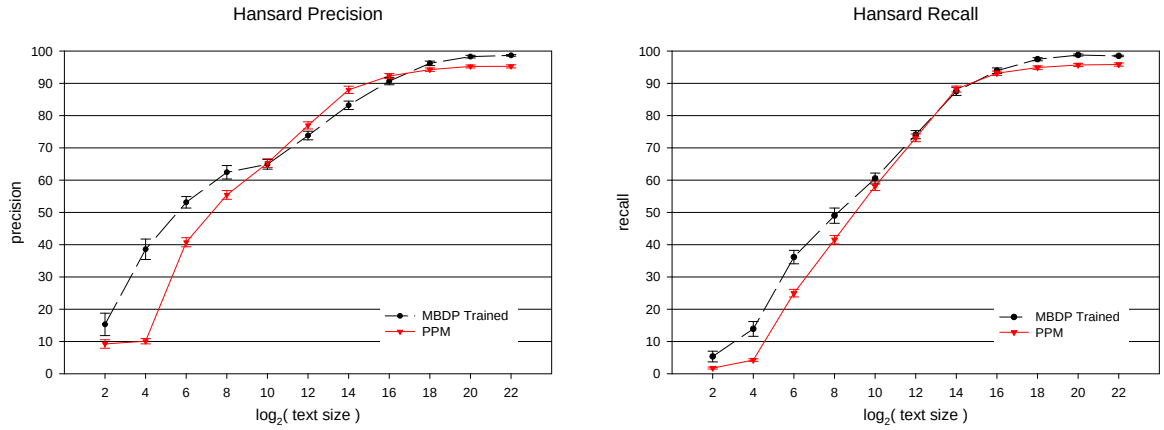

Figure 3: Precision and recall on a sample the English Hansard corpus as a function of $\log_2$ training corpus size: MBDP-1 trained and PPM trained. Error bars are two standard errors of the mean.

be preferred over those that are more difficult to adapt, all other things being equal or nearly so. From this theoretical perspective, then, MBDP-1 appears to be preferable to PPM. If I thought I might be dropped by helicopter into an unexplored region of Borneo with only a palmtop computer, discover a new written language, and need to port a text segmenter to it with minimal effort, I would plan to bring MBDP-1 along.

In practice, adaptation to a completely new language is relatively rare. If the new language represents an important application, significant resources are likely to be available for the port — perhaps even a hand-segmented million-word corpus. Adaptation to a new text genre, however, is extremely common. An individual researcher might want to apply natural language processing tools to, say, novels, poetry, scientific writing, or netnews, without the backing of significant resources such as large, carefully annotated corpora. Such a researcher might be willing to segment 2^{12} or even 2^{14} words of text in the new genre by hand, but probably could not produce a corpus of 2^{16} words, and certainly not 2^{18} words. For this application, then, MBDP-1 would appear to be the best available algorithm.

In the future, we plan to investigate the issue of training on one genre and testing on another, or using a large training corpus from one genre (such as journalistic text) supplemented by a small training corpus from a different test genre. We also plan to investigate the performance of MBDP-1 when it is given a small, hand-segmented training corpus followed by a large, unsegmented "practice" corpus, before testing. This regimen could be very useful for adaptation to a new genre. Finally, we plan to investigate more elaborate probability models and search algorithms. Ultimately, we hope to develop a tool that can adapt rapidly to a new genre with little or no hand-

segmented training text.

5.1 Web site for more information

For more information, please see the web site for the Language Science Research Group at `lsrg.cs.wustl.edu`. An online demo of MBDP-1 (using no training text whatsoever) can be found on the demos page.

Acknowledgments

We are very grateful to Bill Teahan, Yingying Wen, and Ian Witten for sharing their source code with us and helping us to reproduce their experiments. This work was supported, in part, by grant number DC03082 from the National Institutes of Health to MRB.

References

Michael R. Brent. 1999a. An efficient, probabilistically sound algorithm for segmentation and word discovery. *Machine Learning*, 34:71–106.

Michael R. Brent. 1999b. Speech segmentation and word discovery: A computational perspective. *Trends in Cognitive Science*, 3:294–301.

Kwok-Shing Cheng, Gilbert H. Young, and Kam-Fai Wong. 1999. A sudy on word-based and integral-bit chinese text compression algorithm. *Journal of the American Society for Information Science*, 50:218–228.

Delphine Dahan and Michael R. Brent. 1999. On the discovery of novel word-like units from utterances: An artificial-language study with implications for native-language acquisition. *Journal of Experimental Psychology: General*, 128:165–188.

Yubin Dai, Christopher S. G. Khoo, and Teck Ee Loh. 1999. A new statistical formula for chinese text segmentation incorporating contextual information. In *Proceedings of ACM SIGIR*, pages 82–89.

Julia Hockenmaier and Chris Brew. 1998. Error driven segmentation of chinese. In *Communications of COLIPS*, volume 8, pages 69–84.

David Palmer. 1997. A trainable rule-based algorithm for word segmentation. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics.*

Jay M. Ponte and W. Bruce Croft. 1996. USeg: a retargetable word segmentation procedure for information retrieval. Technical Report TR96-2, University of Massachusetts, Amherst, MA.

W. J. Teahan, S. Inglis, John G. Cleary, and G. Holmes. 1998. Correcting english text using ppm models. In J. A. Storer and J. H. Reif, editors, *Proceedings of Data Compression Conference*, pages 289–298, Los Alamitos, CA. IEEE Computer Society Press.

W. J. Teahan, Yingying Wen, Rodger McNab, and Ian H. Witten. 2000. A compression-based algorithm for chinese word segmentation. *Computational Linguistics*, 26:375–393.

Zimin Wu and Gwyneth Tseng. 1993. Chinese text segmentation for text retrieval: Achievements and problems. *JASIS*, 44:532–542.