Self-supervised Chinese Word Segmentation

Fuchun Peng and Dale Schuurmans

Department of Computer Science
University of Waterloo
200 University Avenue West
Waterloo, Ontario, Canada, N2L 3G1
{f3peng,dale}@ai.uwaterloo.ca

Abstract. We propose a new unsupervised training method for acquiring probability models that accurately segment Chinese character sequences into words. By constructing a core lexicon to guide unsupervised word learning, self-supervised segmentation overcomes the local maxima problems that hamper standard EM training. Our procedure uses successive EM phases to learn a good probability model over character strings, and then prunes this model with a mutual information selection criterion to obtain a more accurate word lexicon. The segmentations produced by these models are more accurate than those produced by training with EM alone.

1 Introduction

Unlike English and other western languages, many Asian language such as Chinese, Japanese, and Thai, do not delimited words by white-space. Word segmentation is therefore a key subproblem to language processing tasks (such as information retrieval) in these languages. Unfortunately, segmenting an input sentence into words is a nontrivial task in such cases. For Chinese, there has been a significant amount of research on techniques for discovering word segmentations; see for example [14]. The main idea behind most of these techniques is to start with a lexicon that contains the set of possible Chinese words and then segment a concatenated Chinese character string by optimizing a heuristic objective such as maximum length match, mutual information, or maximum likelihood. This approach implies, however, that one of the main problems in Chinese word segmentation is constructing the original lexicon.

Methods for constructing lexicons can be classified as either supervised or unsupervised. In supervised construction, one has to segment a raw unsegmented corpus by hand and then collect all the words from the segmented corpus to build a lexicon. Unfortunately, since there are over 20,000 Chinese characters, among which 6763 are most commonly used, building a complete lexicon by hand is impractical. Therefore a number of unsupervised segmentation methods have been proposed recently to segment Chinese and Japanese text [1, 3, 8, 12, 9]. Most of these approaches use some form of EM to learn a probabilistic model of character sequences and then employ Viterbi-decoding-like procedures to segment new text into words. One reason that EM algorithm is widely adopted for unsupervised training is that it is guaranteed to converge to a good probability model that locally maximizes the likelihood or posterior probability of the training data [6]. For Chinese segmentation, EM is usually applied by first extracting a lexicon which contains the candidate multi-grams from a given training corpus, initializing a probability distribution over lexicon, and then using the standard iteration to adjust the probabilities of the multi-grams to increase the posterior probability of the training data.

One advantage of unsupervised lexicon construction is that it can automatically discover new words once other words have acquired high probability [4]. For example, if one knows the word "*computer*" then upon seeing "*computerscience*" it is natural to segment "*science*" as a new word. Based on this observation, we propose a new word discovery method that is a variant of standard EM training, but avoids getting trapped in local maxima by keeping two lexicons: a *core* lexicon which contains words that are judged to be familiar, and a *candidate* lexicon which contains all other words that are not in the core lexicon. We use EM to maximize the likelihood of the training corpus given the two lexicons; which automatically suggests new words as candidates for the core. However, once new words have been added to the core, EM is reinitialized by giving half of the total probability mass to the core lexicon, thus allowing core words to guide the segmentation and pull EM out of poor local maxima.

A problem with maximum likelihood training, however, is that it tends to put probability mass on conjunctions of shorter words. Note that since likelihood is defined by a product of individual chunk probabilities (making the standard assumption that segments are independent), the more chunks a segmentation has, the smaller its likelihood will tend to be. For example, in English, given a character sequence *sizeofthecity* and a uniform distribution over multi-grams, the segmentation *sizeof|thecity* will have higher likelihood than segmentation *size|of|the|city*. Therefore, maximum likelihood will prefer fewer chunks in its segmentation and tend to put large probability on long non-word sequences like *sizeof* and *thecity*. If one can break such sequences into shorter legal words, *size, of, the, city*, the lexicon will be much smaller and the training and segmentation performance should be improved. To this end, we employ a mutual information criterion to eliminate longer agglomerations in favor of shorter primitives (once the EM optimization has stabilized). Not only does this have the advantage of producing a smaller core lexicon, it also has the side effect of driving EM out of poor local maxima [6, 2] and yielding better segmentation performance. The remainder of the paper describes the self-supervised training procedure in detail, followed by the mutual information lexicon pruning criterion, experiments, error analysis, and discussion.

2 Self-supervised segmentation

We first develop a technique to help EM avoid poor local maxima when optimizing the probability model. This is done by dynamically repartitioning the vocabulary and reinitializing EM with successively better starting points.

Assume we have a sequence of characters $C = c_1c_2...c_T$ that we wish to segment into chunks $S = s_1s_2...s_M$, where T is the number of characters in the sequence and M is the number of words in the segmentation. Here chunks s_i will be chosen from the core lexicon $V_1 = \{s_i, i = 1, ..., |V_1|\}$ or the candidate lexicon $V_2 = \{s_j, j = 1, ..., |V_2|\}$. If we already have the probability distributions $\theta = \{\theta_i|\theta_i = p(s_i), i = 1, ..., |V_1|\}$ defined over the core lexicon and $\phi = \{\phi_j|\phi_j = p(s_j), j = 1, ..., |V_2|\}$ over the candidate lexicon, then we can recover the most likely segmentation of the sequence $C = c_1c_2...c_T$ into chunks $S = s_1s_2...s_M$ as follows. First, for any given segmentation S of C, we can calculate the joint likelihood of S and C by

$$prob(S, C|\theta, \phi) = \prod_{i=1}^{M_1} p(s_i) \prod_{j=1}^{M_2} p(s_j) = \prod_{k=1}^{M} p(s_k)$$

where M_1 is the number of chunks occurring in the core lexicon, M_2 is the number of chunks occurring in the candidate lexicon, and s_k can come from either lexicon. (Note that each chunk s_k must come from exactly one of the core or candidate lexicons.) Our task is to find the segmentation S^* that achieves the maximum likelihood:

$$S^* = \underset{S}{argmax}\{prob(S|C; \theta, \phi)\} = \underset{S}{argmax}\{prob(S, C|\theta, \phi)\} \tag{1}$$

Given a probability distribution defined by θ and ϕ over the lexicon, the Viterbi algorithm can be used to efficiently compute the best segmentation S of character string C. However, *learning* which probabilities to use given a training corpus is the job of the EM algorithm.

2.1 Parameter re-estimation

Following [6], the update Q function that we use in the EM update is given by

$$Q(k, k+1) = \sum_S prob(S|C; \theta^k, \phi^k) log(prob(C, S|\theta^{k+1}, \phi^{k+1})) \tag{2}$$

Maximizing (2) under the constraints that $\sum_i \theta_i^{k+1} = 1$ and $\sum_j \phi_j^{k+1} = 1$ yields the parameter re-estimation formulas

$$\theta_i^{k+1} = \frac{\sum_S \#(s_i, S) \times prob(S, C|\theta^k, \phi^k)}{\sum_{s_i} \sum_S \#(s_i, S) \times prob(S, C|\theta^k, \phi^k)} \tag{3}$$

$$\phi_j^{k+1} = \frac{\sum_S \#(s_j, S) \times prob(S, C|\theta^k, \phi^k)}{\sum_{s_j} \sum_S \#(s_j, S) \times prob(S, C|\theta^k, \phi^k)} \tag{4}$$

where $\#(s_i, S)$ is the number of times s_i occurring the segmentation S. These are the standard re-estimation formulas, and are the same for θ and ϕ except that each will be reinitialized differently in successive optimizations (see below).

In both cases the denominator is a weighted sum of the number of words in all possible segmentations, the numerator is a normalization constant, and (3) and (4) therefore are weighted frequency counts. Thus, the updates can be efficiently calculated using the forward and backward algorithm, or efficiently approximated using the Viterbi algorithm; see [13] and [5] for detailed algorithms.

2.2 Self-supervised training

The main difficulty with applying EM to this problem is that the probability distributions are complex and typically cause EM to get trapped in poor local maxima. To help guide EM to better probability distributions, we partition the lexicon into a core and candidate lexicon, V_1 and V_2; where V_1 is initialized to be empty and V_2 contains all words. In a first pass, starting from the uniform distribution, EM is used to increase the likelihood of the training corpus C_1. When the training process stabilizes, the N words with highest probability are selected from V_2 and moved to V_1, after which all the probabilities are rescaled so that V_1 and V_2 each contain half the total probability mass. EM is then run again. The rationale for shifting half of the probability mass to V_1 is that this increases the influence of core words in determining segmentations and allows them to act as more effective guides in processing the training sequence. We call this procedure of successively moving the top N words to V_1 *forward selection*.

Forward selection is repeated until the segmentation performance of Viterbi on the validation corpus C_2 leads to a decrease in F-measure (which means we must have included some erroneous core words). After forward selection terminates, N is decremented by 5, and we carry out a process of *backward deletion*, where the N words with the lowest probability in V_1 are moved back to V_2, and EM training is successively repeated until F-measure again decreases on the validation corpus C_2 (which means we must have deleted some correct core words). The two procedures of forward selection and backward deletion are alternated, decrementing N by 5 at each alternation, until $N \leq 0$; as shown in Fig. 1. As with EM, the outcome of this self-supervised training procedure is a probability distribution over the lexicon that can be used by Viterbi to segment test sequences.

3 Mutual information lexicon pruning

Both EM and self-supervised training acquire probability distributions over the entire lexicon. However, as pointed out by [12], the lexicon is the most important factor in the word segmentation, and therefore a better lexicon is more critical than a better model. Unfortunately, by maximizing likelihood, either through EM or self-supervised training, erroneous agglomerations tend to get naturally introduced in the lexicon. This means that after a high-likelihood model has been learned, we are still motivated to prune the lexicon to remove erroneous non-word entries. We do this by invoking a mutual information based criterion.

```
0. Input
    Completely unsegmented training corpus C1
    Validation corpus C2

1. Initialize
    V1 = empty;
    V2 contains all potential words;
    OldFmeasure = infinite small;
    bForwardSelection=true;
    set N to a fix number;

2. Iterate
    while (N > 0){
       EM Learning based on current V1 and V2 until converge;

       Calculate NewFmeasure on validation corpus C2;

       if(NewFmeasure < OldFmeasure){
          // change selection direction
          bForwardSelection = ¬bForwardSelection;
          N = N-5;
       }

       OldFmeasure = NewFmeasure;

       //SelectCoreWords(true) does forward selectoin
       //SelectCoreWords(false) does backward selectoin
       SelectCoreWords(bForwardSelection);
    }

3. Test
    Test on test corpus C3
```

Fig. 1. Self-supervised Learning

Recall that the mutual information between two random variables is given by

$$MI(X,Y) = \sum_{x,y} p(x,y) log \frac{p(x,y)}{p(x) \times p(y)} \quad (5)$$

where a large value indicates a strong dependence between X and Y, and zero indicated independence. In our case, we want to test the dependence between two chunks s_1 and s_2. Given a long word, say $s=$*"abcdefghijk"*, we consider splitting it into its most likely two-chunk segmentation, say $s_1=$*"abcd"* and $s_2=$*"efghijk"*. Let the probabilities of the original string and the two chunks be $p(s)$, $p(s_1)$, and $p(s_2)$ respectively. The *pointwise mutual information* [10] between s_1 and s_2 is

$$MI(s_1, s_2) = log \frac{p(s)}{p(s_1) \times p(s_2)}. \quad (6)$$

To apply this measure to pruning, we set two thresholds $\gamma_1 > \gamma_2$. If the mutual information is higher than the threshold γ_1, we say s_1 and s_2 are strongly correlated, and do not split s. (That is, we do not remove s from the lexicon.) If mutual information is lower than the lower threshold γ_2, we say s_1 and s_2 are independent, so we remove s from the lexicon and redistribute its probability to s_1 and s_2. If mutual information is between the two thresholds, we say s_1 and s_2 are weakly correlated, and therefore shift some of the probability from s to s_1 and s_2, by keeping a portion of s's probability for itself (1/3 in our experiments) and distributing the rest of its probability to the smaller chunks, proportional to their probabilities. The idea is to shift the weight of the probability distribution toward shorter words. This splitting process is carried out recursively for s_1 and s_2. The pseudo code is illustrated in Fig. 2.

```
1: (s1, s2) = mostlikely_split(s);

2: MI = log p(s) / (p(s1) × p(s2))

3: if(MI > γ1){//strongly dependent
        return -1;
   }
   else if(MI < γ2){//independent
       probSum = p(s1) + p(s2);
       p(s1)+ = p(s) × p(s1)/probSum;
       p(s2)+ = p(s) × p(s2)/probSum;
       p(s) = 0;
       return 1;
   }
   else{//weakly dependent
       probDistribute = p(s)/3;
       probSum = p(s1) + p(s2);
       p(s) = probDistribute;
       p(s1)+ = 2 × probDistribute × p(s1)/probSum;
       p(s2)+ = 2 × probDistribute × p(s2)/probSum;
       return 0;
   }
```

Fig. 2. Mutual information probabilistic lexicon pruning

4 Experiments

To compare our technique to previous results, we follow [8, 12] and measure performance by precision, recall, and F-measure on detecting word boundaries. Here, a word is considered to be correctly recovered iff [11]:

1. a boundary is correctly placed in front of the first character of the word,
2. a boundary is correctly placed at the end of the last character of the word,
3. and there is no boundary between the first and last character of the word.

Let N_1 denote the number of words in the test corpus C_3, let N_2 denote the number of words in the recovered corpus $C_3{}'$, and let N_3 denote the number of words correctly recovered. Then the precision, recall and F measures are defined

$$\text{precision: } p = \frac{N_3}{N_2}$$
$$\text{recall: } r = \frac{N_3}{N_1}$$
$$\text{F-measure: } F = \frac{2 \times p \times r}{p + r}$$

We use a training corpus, C_1, that consists of 90M (5,032,168 sentences) of unsegmented Chinese characters supplied by the author of [8], which contains one year of the "People's Daily" news service stories (www.snweb.com). The test corpus C_3 is ChineseTreebank from LDC[1] (1M, 10,730 sentences), which contains 325 articles from "Xinhua newswire" between 1994 and 1998 that have been correctly segmented by hand. We remove all white-space from C_3 and create an unsegmented corpus $C_3{}''$. We then use the algorithm described in Section 2 to recover $C_3{}'$ from $C_3{}''$. The validation corpus, C_2, consists of 2000 sentences randomly selected from the test corpus.

According to the *1980 Frequency Dictionary of Modern Chinese* (see [7]), the top 9000 most frequent words in Chinese consist of 26.7% unigrams, 69.8%

[1] http://www.ldc.upenn.edu/ctb/

bigrams, 2.7% trigrams, 0.007% 4-grams, and 0.002% 5-grams. So in our model, we limit the length of Chinese words up to 4 characters, which is sufficient for most situations.

The experimental results are shown in Table 1—Results 1 are obtained by standard EM training, Results 2 are obtained by self-supervised training, Results 3 are obtained by repeatedly applying lexicon pruning to standard EM training, and Results 4 are obtained by repeatedly applying lexicon pruning to self-supervised training. The row labeled Perfect lexicon is obtained by using maximum length match with the complete lexicon of the test corpus, which contains exactly all the words occurring in the test corpus. Soft-counting is the result of [8][2], which is also a EM-based unsupervised learning algorithm. The Word-based results are from [12] which uses a suffix tree model. Finally, the Perfect lexicon[12] results are obtained using a lexicon from another test corpus.

	p	r	F	final lex size
Results 1	44.6%	37.3%	40.0%	19044012
Results 2	55.7%	53.9%	54.1%	19044012
Results 3	73.2%	71.7%	72.1%	1670317
Results 4	75.1%	74.0%	74.2%	1670317
Perfect lexicon	92.2%	91.8%	91.9%	10363
Soft-counting[8]	71.9%	65.7%	68.7%	
Word-based[12]	84.4%	87.8%	86.0%	
Perfect lexicon[12]	95.9%	93.6%	94.7%	

Table 1. Experimental results

There are several observations to make from these results. First, self-supervised training improves the performance of standard EM training from 40% to 54.1%. Second, mutual information pruning gives even greater improvement (from 40% to 72.1%), verifying the claim of [12] that the lexicon is more influential than the model itself. The lexicon pruning reduces the lexicon size from 19044012 to 1670317, which makes the lexicon much smaller. By combining the two strategies, we obtain further improvement (from 40% to 74.2%), which is promising performance considering that we are using a purely unsupervised approach. By comparison, the result given by a perfect lexicon is 91.9%. Finally, the two improvement strategies of self-supervised training and lexicon pruning are not entirely complementary and therefore the resulting performance does not increase additively when both are combined (72.1% using lexicon pruning alone to 74.2% using both).

A direct comparison can be made to [8] because it also investigates a purely unsupervised training approach without exploiting any additional context information and uses the same training set we have considered. When we compare

[2] They did not supply F-measure, we calculate it for comparison.

the results, we find that self-supervised training plus lexicon pruning achieves both a higher precision and recall than [8], and obtains a **5.5%** (from 68.7% to 74.2%) improvement in F-measure. One problem with this comparison, however, is that [8] does not report precisely how the testing was performed. It is also possible to compare to [12], which uses a suffix tree model and employs context information (high entropy surroundings) to achieve an 86% F-measure score. This result suggests the context information is very important. However, because of a different test set (our test set is the 1M ChineseTreebank from LDC, whereas their test data is 61K pre-segmented by NMSU segmenter [9] and corrected by hand), the comparison is not fully calibrated. In the perfect lexicon experiments, [12] achieves higher performance (94.7% F-measure), whereas only 91.9% is achieved in our experiments. This suggests that we may obtain better performance when testing on the data used in [12]. Nevertheless, the result of [12] appears to be quite strong, and demonstrates the utility of using local context rather than assuming independence between words, as we have done.

5 Error analysis

Fig. 3 shows two categories of errors that are typically committed by our model. In each case, the left string shows the correct word and the right bracketed string shows the recovered word segmentation. The first error category is *date and number*. In Chinese, dates and numbers are represented by 10 characters. Unlike Arab numbers, these 10 Chinese number characters are not different from other Chinese characters. Most dates and numbers are not correctly recognized because they do not have sufficient joint statistics in the training corpus to prevent them from being broken down into some smaller numbers. For example, *"1937 year"* is broken into *"19"*, *"3"*, *"7 year"*; *"2 wan 3 qian 1 bai 1 shi 4"* is broken into *"2 wan"*, *"3 qian"*, *"1"*, *"bai 1 shi 4"* (*wan* denotes 10-thousand, *qian* denotes thousand, *bai* denotes hundred, and *shi* denotes ten). It turns out that one can easily use heuristic methods to correct errors in these special cases. For example, if a string of concatenated characters are all number characters, then the string is very likely to be a single date or number.

The second error category is the recognition of compound nouns. For example, the compound *"total marks"* is recovered as two words *"total"* and *"marks"*; *"team Australia"* is recovered to *"team"* and *"Australia"*. One reason for the failure to correctly recover compounds is that we are limiting the maximum length of a word to 4 characters, whereas many compounds are longer than this limit. However, simply relaxing the length limit creates a significant sparse data problem. The recognition of noun compounds appears to be difficult, and a general approach may have to consider language constraints.

6 Related work

Our work is related to many other research efforts.

```
A:Date and Numbers
一九三七年          （ 一九 三 七年 ）
二万三千一百一十四 （ 二万 三千 一 百一十四 ）

B:Compounds
总成绩              （ 总 成绩 ）
澳大利亚队          （ 澳大利亚 队 ）
```

Fig. 3. Typical segmentation errors

Unsupervised Chinese Word Segmentation: [8] uses a soft counting version of EM to learn how to segment Chinese. To augment the influence of important words, [8] shifts probability mass to likely words by soft counting. In our model, we shift half of the probability space to the core words by dividing the lexicon to two parts. Also, [8] does not employ any sort of lexicon pruning, which we have found is essential to improving performance. [12] uses a suffix tree word-based model and a bigram model to segment Chinese strings. This work takes the surrounding word information into consideration when constructing the lexicon. [14] uses a more complicated Hidden Markov Model (HMM) model that includes special recognizers for Chinese names and a component for morphologically derived words. As pointed out by [8], standard EM segmentation can be thought of as a zero order HMM.

Mutual Information Lexicon Optimization: Other researchers have considered using mutual information to build a lexicon. For example, [14] uses mutual information to build a lexicon, but only deals with words of up to 2 characters. [3, 12] uses mutual information and context information to build a lexicon based on the statistics directly obtained from the training corpus. By contrast, we are using mutual information to prune a given lexicon. That is, instead of building a lexicon from scratch, we first add all possible words and then use mutual information to prune away illegal words after training with EM. Hence the statistics we use for calculating mutual information are more reliable than those directly obtained from corpus by frequency counting. Another difference is that we are using a probabilistic splitting scheme that sometimes just shifts probability between words, instead of completely discarding words.

7 Conclusion and future work

This paper describes a new unsupervised method for discovering Chinese words from an unsegmented corpus. Combined with an efficient mutual information based lexicon pruning scheme, we achieve competitive results.

However, there is much work left to be done. One problem is that we cannot detect the hierarchical structure of Chinese compounds, which is very useful in many NLP tasks. We are currently exploring a hierarchical unsupervised method to detect Chinese compounds. Also, instead of assuming the independence of each word, we should also consider context information, which has proven to be helpful [12]. Another problem with our self-supervised training procedure is that it puts equal weight on the core and candidate lexicons. One interesting idea would be to automatically estimate the weights of the two lexicons by using a mixture model.

8 Acknowledgments

We would like to thank Ge, Xianping [8] for kindly supplying us with the training corpus. Research supported by Bell University Labs, MITACS and NSERC.

References

1. Ando, R. and Lee, L.; Mostly-Unsupervised Statistical Segmentation of Japanese: Application to Kanji. ANLP-NAACL, 2000.
2. Brand, M.; Structure learning in conditional probability models via an entropic prior and parameter extinction. In Neural Computation, vol.11, page 1155-1182, 1999.
3. Chang, J.-S. and Su, K.-Y.; An Unsupervised Iterative Method for Chinese New Lexicon Extraction. International Journal of Computational Linguistics & Chinese Language Processing, 1997.
4. Dahan, D. and Brent, M.; 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, 165-185, 1999.
5. Deligne, S. and Bimbot, F.; Language Modeling by Variable Length Sequences: Theoretical Formulation and Evaluation of Multigrams. ICASSP, 1995.
6. Dempster, A., Laird, N, and Rubin, D.; Maximum-likelihood from incomplete data via the EM algorithm. J. Royal Statist. Soc. Ser. B., 39, 1977.
7. Fung, P.; Extracting key terms from Chinese and Japnese text. The International Journal on Computer Processing of Oriental Language, Special Issue on Information Retrieval on Oriental Languages, 1998, 99-121.
8. Ge, X., Pratt, W. and Smyth, P.; Discovering Chinese Words from Unsegmented Text. SIGIR-99, pages 271-272.
9. Jin, W.; Chinese Segmentation and its Disambiguation. MCCS-92-227, Computing Research Laboratory, New Mexico State University, Las Cruces, New Mexico.
10. Manning, C. and Schütze, H.; Foundations of Statistical Natural Language Processing. MIT Press, Cambridge, Massachusetts, 1999, pages 66-68.
11. Palmer, D. and Burger, J.; Chinese Word Segmentation and Information Retrieval. AAAI Spring Symposium on Cross-Language Text and Speech Retrieval, Electronic Working Notes, 1997.
12. Ponte, J. and Croft, W.; Useg: A retargetable word segmentation procedure for information retrieval. Symposium on Document Analysis and Information Retrival 96 (SDAIR).
13. Rabiner, L.; A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of IEEE, Vol.77, No.2, 1989.
14. Sproat, R., Shih, C., Gale, W. and Chang, N.; A stochastic finite-state word-segmentation algorithm for Chinese Computational Linguistics, 22 (3), 377-404, 1996.