

Exploiting Unlabeled Internal Data in Conditional Random Fields to Reduce Word Segmentation Errors for Chinese Texts

Richard Tzong-Han Tsai, Hsi-Chuan Hung, Hong-Jie Dai, and Wen-Lian Hsu

Institute of Information Science, Academia Sinica, Taipei, Taiwan, R.O.C.

{thtsai, yabt, hongjie, hsu}@iis.sinica.edu.tw

Abstract

The application of text-to-speech (TTS) conversion has become widely used in recent years. Chinese TTS faces several unique difficulties. The most critical is caused by the lack of word delimiters in written Chinese. This means that Chinese word segmentation (CWS) must be the first step in Chinese TTS. Unfortunately, due to the ambiguous nature of word boundaries in Chinese, even the best CWS systems make serious segmentation errors. Incorrect sentence interpretation causes TTS errors, preventing TTS's wider use in applications such as automatic customer services or computer reader systems for the visually impaired. In this paper, we propose a novel method that exploits unlabeled internal data to reduce word segmentation errors without using external dictionaries. To demonstrate the generality of our method, we verify our system on the most widely recognized CWS evaluation tool--the SIGHAN bakeoff, which includes datasets in both traditional and simplified Chinese. These datasets are provided by four representative academies or industrial research institutes in HK, Taiwan, Mainland China, and the U.S. Our experimental results show that with only internal data and unlabeled test data, our approach reduces segmentation errors by an average of 15% compared to the traditional approach. Moreover, our approach achieves comparable performance to the best CWS systems that use external resources. Further analysis shows that our method has the potential to become more accurate as the amount of test data increases.

Index Terms: text-to-speech, Chinese word segmentation, segmentation errors, internal unlabeled data

1 Introduction

For most languages, word segmentation is a trivial problem because individual words are delimited by spaces. In contrast, Chinese, Japanese and a handful of other languages do not have word delimiters, which poses a serious obstacle to text-to-speech (TTS): Before Chinese text can be processed by any TTS system, it must be segmented into words, a process termed Chinese word segmentation (CWS). To understand the difficulties facing CWS, we must outline some basic information about the Chinese language. An oft held but erroneous view of Chinese is that it is a monosyllabic tonal language, with each Chinese character corresponding to a single syllable and tone. In fact, most nouns, verbs and adjectives are character pairs pronounced one after the other. Obviously this has a big impact on the rhythm, cadence and stress of sentence pronunciation. Secondly, the tones of certain characters are not fixed, but change depending on adjacent characters [1]. A well-known example of this is the rule of third-tone pairs: when one third tone character follows another, the first character is pronounced with the second tone. Furthermore, the tonal and/or phonetic pronunciations of other characters will change depending on their grammatical role (ie: noun/verb) or sense. These types of characters are referred to in Chinese as Poyin characters, or "broken sound".

The following example shows how a Chinese sentence might be segmented in two ways:

- 學 | 會計 | 算 | 微分方程 (wrong)
learn | accounting | (and) calculate | differential equations
- 學會 | 計算 | 微分方程 (correct)
learn to | calculate | differential equations

As you can see, even an error of only one character can dramatically change the interpretation of a sentence. Such mix-ups can make the implementation of TTS in applications like customer service or computer reader systems for the blind impractical or even unsafe.

Recently, statistical machine learning (ML) has become a more desirable Chinese word segmentation (CWS) approach. It formulates the CWS problem as a character tagging problem [2]. That is, assigning each character a tag to indicate its position in a word. N -gram features that represent the current character and its context by describing the relative positions of certain characters to the current character within a context window are the major features in this approach. In spite of their popularity, n -gram features have some weakness. Short n -gram features may cause CWS systems to over-segment words longer than n , especially when processing a long compound word. However, employing longer n -gram features consumes a large amount of system memory. Some ML-based CWS systems [2] employed dictionary features to relieve the over segmentation problem. These features indicate whether or not a group of consecutive characters in the context matches a word in a given dictionary. All matches with the same relative position to the current character and the same length are considered as enabling the same feature; therefore, the number of dictionary features is much less than that of n -gram features.

The main challenge of employing dictionary features is how to create the dictionary. The simplest dictionary is constructed by collecting all words from the training set, which is referred to as the training dictionary. However, using solely the training dictionary leads to a large number of OOV words, which are often incorrectly segmented into single-character or over-short words [3] and therefore the words are read character by character by the TTS system. To overcome this problem, the dictionary coverage of the test set must also be improved. One possible solution is to identify new words from the test set and add them to the dictionary. The resulting dictionary is called transductive dictionary since the construction procedure follows the general concepts of transductive learning.

In this paper, we propose a method to hybridize n -gram and dictionary features which relieves the segmentation error problem. We also apply reliable measurements to show the generality and performance of our proposed method. Another noticeable advantage is that as the size of the unlabeled text data increases, our method achieves a higher error reduction rate.

2 The Model

2.1 Dataset and Formulation

We use the four datasets in SIGHAN Bakeoff 2006 [4], which is the best-known CWS evaluation. Table 1 illustrates their statistics. These datasets are sufficiently large and provided by four experienced institutes in the Chinese language processing field: the City University of Hong Kong (CITYU), Academia Sinica in Taiwan (CKIP), Microsoft Research Asia (MSRA) in China, and University of Pennsylvania (UPUC) in the U.S.. The former two are in traditional Chinese, while the others are in simplified Chinese.

Table 1. *Corpus statistics*

Source	Training (Wds/Distinct Wds)	Test (Wds/Distinct Wds)
CITYU	1.6M/76K	220K/23K
CKIP	5.5M/146K	91K/15K
MSRA	1.3M/63K	100K/13K
UPUC	509K/37K	155K/17K

SIGHAN bakeoff have two sessions: closed and open. Closed has a stringent but interesting constraint, which does not allow participating systems to use external resources, such as external corpora and dictionaries. The motivation of closed session is to test if the participating system can improve CWS performance without using any additional corpora and handcrafted dictionaries/rules. On the contrary, the open session does not have this constraint. Since our goal is to study how to fully utilize the internal data, we follow closed constraints.

For the training datasets, we convert the manually segmented words (namely training data) into tagged sequences of Chinese characters. To do this, we tag each character with either *B*, if it begins a word, or *I*, if it is inside or at the end of a word. A single-character word is tagged as *B*. For example, the manually segmented string in (2.1) will be tagged as (2.2) in the IOB2 format.

生产 | 总值 | 五千 | 美元... (2.1)

生/B 产/I 总/B 值/I 五/B 千/I 美/B 元/I... (2.2)

2.2 Conditional Random Fields

CRFs are undirected graphical models trained to maximize a conditional probability [5]. A linear-chain CRF with parameters $\Lambda = \{\lambda_1, \lambda_2, \dots\}$ defines a conditional probability for a state sequence $\mathbf{y} = y_1 \dots y_T$ given an input sequence $\mathbf{x} = x_1 \dots x_T$ to be

$$P_{\Lambda}(\mathbf{y} | \mathbf{x}) = \frac{1}{Z_{\mathbf{x}}} \exp\left(\sum_{t=1}^T \sum_k \lambda_k f_k(y_{t-1}, y_t, \mathbf{x}, t)\right)$$

where $Z_{\mathbf{x}}$ is the normalization that makes the probability of all state sequences sum to one; $f_k(y_{t-1}, y_t, \mathbf{x}, t)$ is often a binary-valued feature function and λ_k is its weight.

The feature functions can measure any aspect of a state transition, $y_{t-1} \rightarrow y_t$, and the entire observation sequence, \mathbf{x} , centered at the current position, t . For example, one feature function might have value 1 when y_{t-1} is the state *B*, y_t is the state *I*, and is the character "国". Large positive values for λ_k indicate a preference for such an event; large negative values make the event unlikely.

The most probable label sequence for \mathbf{x} ,

$$\mathbf{y}^* = \underset{\mathbf{y}}{\operatorname{argmax}} P_{\Lambda}(\mathbf{y} | \mathbf{x}),$$

can be efficiently determined using the Viterbi algorithm [6].

3 Features

In our CRF model, each binary feature is multiplied with all states (y_t) or all state transitions ($y_{t-1}y_t$). For simplicity, we omit them in the following discussion. In addition, we use C_0 rather than x_t to denote the current character.

3.1 N-gram Features

Character n -gram features have proven their effectiveness in statistical learning CWS [7]. We use four types of unigram feature functions, designated as C_0 , C_1 (next character), C_{-1} (previous character), C_{-2} (character preceding C_{-1}). Furthermore, six types of bigram features are used, and are designated as conjunctions of the previously specified unigram features, $C_{-2}C_{-1}$, $C_{-1}C_0$, C_0C_1 , $C_{-3}C_{-1}$, $C_{-2}C_0$, and $C_{-1}C_1$.

3.2 Transductive Dictionary Features

If a sequence of characters in a sentence matches a word w in an existing dictionary, it may indicate that the sequence of characters should be segmented as one word. We develop word match features that also contain frequency information. The word match feature described in next section is used only for comparison. In the following sections, we use D to denote the dictionary, whose construction is explained later.

3.2.1 Word Match Features (WM)

The simplest dictionary feature indicates if there is a sequence of neighboring characters around C_0 that match a word in D . Features of this type are identified by their positions relative to C_0 and their lengths. Word match features are defined as:

$$\text{WM}(w = C_{-pos} \dots C_{-pos+len-1}) = \begin{cases} 1 & \text{if } w \in D \\ 0 & \text{otherwise} \end{cases}$$

where $len \in [1..4]$ is w 's length and $pos \in [0..len]$ is C_0 's zero-based relative position in w (when $pos=len$, the previous len characters form a word found in D).

3.2.2 Word Match Frequency Features (WMF)

Given two different words that have the same position and length, WM features cannot differentiate which should have the greater weight; therefore, we add word frequency information to the basic WM feature:

$$\text{WMF}_q(w = C_{-pos} \dots C_{-pos+len-1}) = \begin{cases} 1 & \text{if } w \in D \text{ and } \log_freq(w) = q \\ 0 & \text{otherwise} \end{cases}$$

here the word frequency are discretized into 10 bins in a logarithmic scale:

$$\log_freq(w) = \min(\lceil \log_2 w\text{'s frequency} + 1 \rceil, 10)$$

thus $q \in [0..10]$ is the discretized log frequency of w . In this formulation, matching words with higher log frequencies are more likely to be the correct segmentation.

3.2.3 Discretization of Feature v.s. Zipf's Law

Since current implementation of CRF models only allows discrete features, the word frequency must be discretized. There are two commonly used discretization methods: equal-width interval and equal-frequency interval, where the latter is shown to be more suitable for data following highly skewed distribution [8]. The word frequency distribution is exactly the case: Zipf's law [9] states that the word frequency is inversely proportional to its rank:

$$f(x) \propto z^{-\alpha}$$

where $f(x)$ is x 's frequency, z is its rank in the frequency table, and α is empirically found to be close to unity. Obviously this distribution is far from flat uniform. Hence the equal-frequency binning turns out to be our choice.

Ideally, we would like that each bin has equal *expected* number of values rather than following *empirical* distribution. Therefore, we attempt to discretize according to their underlying Zipfian distribution.

[10] shows that Zipf's law is equivalent to the power law, which describes Zipf's law in a unranked form:

$$f_X(x) \propto x^{-(1+(1/\alpha))},$$

where X is the random variable denoting the word frequency and $f_X(x)$ is its probability density function. Approximated by integration, the expected number of values in the bin $[a, b]$ can be calculated as

$$\begin{aligned} \sum_{a \leq x \leq b} x \cdot \Pr[X = x] &\approx \int_a^b x \cdot f_X(x) dx \\ &\propto \int_a^b x \cdot x^{-(1+(1/\alpha))} dx \approx \ln x \Big|_a^b = \ln(b/a) \quad (\because \alpha \approx 1) \end{aligned}$$

Thus the each bin has equal number of values within it if and only if b/a is a constant, which is in a log scale. This shows that our strategy to discretize the WMF features in a log scale has theoretical support.

3.2.4 Balanced Transductive Dictionary

The first problem encountered when using the training dictionary is its lack of test set coverage. During training, since its coverage is 100%, the enabled dictionary features will be assigned very high weights while n -gram features will be assigned low weights. During testing, when coverage is far below 100%, most tags are decided by dictionary features enabled by in-vocabulary (IV) words, while n -gram features have little influence. Therefore, it is likely that only IV words are correctly segmented, while OOV words are over-segmented. Loosely speaking, a dictionary's coverage of the training set is linked to the degree of reliance placed by the CRF model on the corresponding dictionary features. Therefore, a dictionary is said to be more balanced if its coverage of the training set approximates its coverage of the test set while maximizing the latter.

One way to construct such a balanced dictionary is to extract words from the test set. Such a dictionary is defined as a transductive dictionary (TD). The most basic TD is the Naive TD (NTD), which is extracted from the gold standard training set and the tagged test set that is labeled by their initial model. [2] is an example of creating a NTD. Inspired by their NTD, we construct our own TD as follows: (1) We use the initial model, which employs only n -gram features, to label the test set and add all segmented words into our TD. (2) The next step is to balance our TD's coverage of the training and test sets. Since coverage of the test set cannot reach 100%, the only way to achieve this goal is by slightly lowering the dictionary's coverage on the training set. We apply n -fold cross-tagging to label the training set data; Each fold that has $1/n$ of the training set is tagged by the model trained on the other $n-1$ folds with only n -gram features. All the words identified by this cross-tagging process are then added to our TD.

Notice that we extract words from the cross-tagged training set not from gold-standard training set. Furthermore, our TD is used to generate dictionary features to train the final model. Since the TD constructed from cross-tagging training set and tagged test set exists more balanced coverage of the training and test set, we call such a TD "balanced TD", shorted as BTD. The goal of the whole procedure is to ensure that the dictionary coverage of the training and test sets is more balanced.

4 Evaluation Metrics and Results

4.1 Evaluation Metrics

As shown in Table 1, we use SIGHAN Bakeoff 2006's four datasets to evaluate our CWS systems. SIGHAN's official evaluation tool provides three metrics: precision (P), recall (R), and F-measure (F). They are defined as follows:

$$\begin{aligned} P &= \# \text{ of correctly segmented words} / \# \text{ of segmented words} \\ R &= \# \text{ of correctly segmented words} / \# \text{ of words} \\ F &= 2PR / (P+R) \end{aligned}$$

In addition, it also reports the NChange, which is equal to the number of segmentation errors. Since the F-measures of state-of-the-art CWS systems are over 90%, there are usually minor quantitative difference between them. Therefore, we use NChanges to calculate a finer-grained metric, the error reduction rate (ERR) from the baseline result:

$$ERR(s) = (NChange_b - NChange_s) / NChange_b$$

where s stands for the compared system and b stands for the baseline system.

4.2 Results

The baseline system uses the n -gram features described in Section 3.1. It is denoted as N-grams in Table 2.

4.2.1 Adding Dictionary Features

Without lost of generality, we use the WM features introduced in Section 3.2.1 to represent dictionary features. As shown in Table 2, we can see that in all datasets, the configuration with WM features (config. 2) outperforms that with N-grams (config. 1). In Table 2, adding WM features results in ERRs of 10.8%, 3.8%, 0.4%, and 6.5% on CTU, CKIP, MSRA and UPUC test sets, respectively

4.2.2 Adding Frequency Information

We can see that WMF features (config. 3) in Table 2 outperform WM features (config. 2) on all datasets in terms of F-Measure. These results demonstrate the effectiveness of frequency information.

4.2.3 Balancing Dictionary Coverage

In Table 2, we can also see that config. 4 (with our proposed BTD) outperforms config. 3 (with NTD). We then conduct an experiment to further investigate the relationship between coverage balance (between the training and test set dictionaries) and NChange. First, we define a metric, weighted-coverage (WCov) to measure the coverage of a dictionary D on a gold-standard corpus G :

$$WCov(D, G) = (\sum_{t \in G} t * \text{freq}(t)) / |G|$$

Then, we define D 's coverage difference between the training set, $Train$, and test set, $Test$:

$$CovDiff(D) = |WCov(D, Test) - WCov(D, Train)|$$

To normalize NChanges among different datasets, we define the relative NChange (RelNC) as follows:

$$RelNC(s) = NChange_{baseline} / NChange_s$$

where config. 3 serves as *baseline* and config. 4 as s in the above formula.

Now, we start to depict the relationship between RelNC and CovDiff. To generate different BTDs with different CovDiffs, we vary n from 2 to 40 in n -fold cross tagging. In Figure 1, data points of different datasets are denoted by different labels, and the regression line for each dataset is drawn. As we

Table 2. Performance for the four datasets of SIGHAN Bakeoff 2006

Configuration	CITYU			CKIP			MSRA			UPUC		
	F	NC	ERR	F	NC	ERR	F	NC	ERR	F	NC	ERR
1 N-grams	0.966	9642	n/a	0.954	5621	n/a	0.953	6323	n/a	0.930	14094	n/a
2 (1) + WM, NTD	0.97	8597	10.8%	0.956	5410	3.8%	0.953	6301	0.4%	0.934	13182	6.5%
3 (1)+WMF, NTD	0.972	8029	16.7%	0.958	5174	8.0%	0.955	6062	4.1%	0.937	12719	10.0%
4 (1) + WMF, BTD	0.975	7344	23.8%	0.959	5102	9.2%	0.960	5459	13.7%	0.940	12218	13.3%
5 Best Closed	0.972	n/a	n/a	0.958	n/a	n/a	0.963	n/a	n/a	0.933	n/a	n/a
6 Best Open	0.977	n/a	n/a	0.959	n/a	n/a	0.979	n/a	n/a	0.944	n/a	n/a

can see, for each dataset, there is a positive correlation between RelNC and CovDiff.

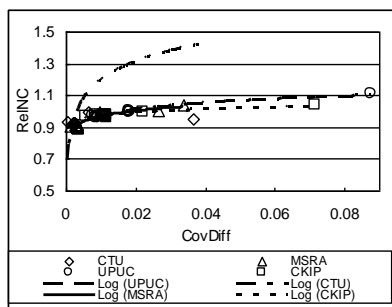


Figure 1. RelNC v.s. CovDiff

4.2.4 Compared with the Best SIGHAN Systems

As shown in Table 2, compared with the best SIGHAN closed systems (config. 5), our systems (config.4) outperforms them in three out of four datasets. Our systems also achieve comparable performance to the best SIGHAN open systems (config. 6).

4.2.5 Varying the Size of Test Data

In this section, we test if our method is effective with larger unseen test set. To calculate ERRs, we choose N-grams (config. 1) as the baseline system and N-grams+WMF (config. 4) as the compared system s , which represents our approach. We randomly selected 7183 sentences, or 1/8 of the original training set of CITYU, as our new training set. We then fixed the training set size and gradually increased the test-train ratio from 1:1 to 8:1. The test sentences were selected from the remaining training set and the entire test set.

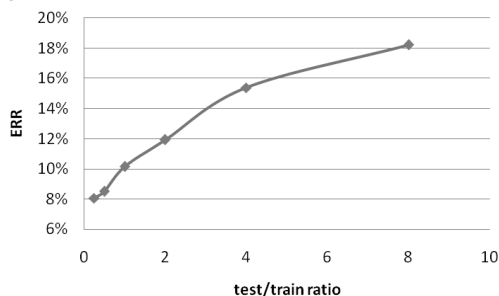


Figure 2. ERRs in different test-train ratios.

Figure 2 show the ERRs for different test-train ratios. We can see that the ERRs of our approach increase significantly (from 7% to 18%) with the test set size. This improvement is due to the fact that once a new word w is detected by its context and added to the TD, the segmentations of w 's other occurrences become more likely to be correct due to the dictionary features enabled by w .

5 Conclusion

Segmentation of Chinese sentences is the first key step to realize high performance Mandarin Chinese text-to-speech synthesis system. We propose a general approach to exploit the unlabeled internal text data to extend traditional dictionary features. We discover that the unbalanced coverage of internal dictionaries between the training and test sets causes over-weighted dictionary features. We deal with this problem by using the balanced transductive dictionary. Besides, we incorporate word frequency information both from the training set and unlabeled test set. We verify our system on the best-known CWS evaluation, the recent SIGHAN bakeoff. With only training data and unlabeled internal data, our system outperforms the best SIGHAN systems using the same resources. Notably, our system achieves a comparable performance to the best SIGHAN system that uses external annotated data and dictionaries. Compared to the system using the conventional n -gram features, our system can reduce segmentation errors by averagely 15%. Our system shows the potential to be better when more unseen data are used. In the future, we will apply our CWS method to the TTS system and evaluate the performance gain brought by the improved CWS system.

6 Acknowledgements

This research was supported in part by the National Science Council under Grant NSC 96-2752-E-001-001-PAE.

7 References

- [1] Badino, L., "Chinese Text Word Segmentation Considering Semantic Links among Sentences", in Proc. of ICSLP 2004, , 2004.
- [2] Peng, F., Feng, F., and McCallum, A. "Chinese Segmentation and New Word Detection using Conditional Random Fields", in Proc. of COLING-04, 2004.
- [3] Chen, K.-J. and Bai, M.-H., "Unknown Word Detection for Chinese by a Corpus-based Learning Method", Computational Linguistics and Chinese Language Processing, 3(1):27-44, 1998.
- [4] <http://www.sighan.org/bakeoff2006/main.html>
- [5] Lafferty, J., McCallum, A., and Pereira, F., "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data", in Proc. of ICML-01, 2001.
- [6] Rabiner, L., "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", Readings in Speech Recognition, 1990.
- [7] Gao, J., Wu, A., Li, M., Huang, C.-N., Li, H., Xia, X., and Qin, H. (2004). "Adaptive Chinese Word Segmentation", in Proc. of ACL-04, 2004.
- [8] Ismail, M. K. and Ciesielski, V., "An Empirical Investigation of the Impact of Discretization on Common Data Distributions", Design and Application of Hybrid Intelligent Systems.
- [9] Zipf, G. K. "Human Behavior and the Principle of Least Effort", 1949.
- [10] Adamic, L. A. and Huberman, B. A., "Zipf's Law and the Internet", Glottometrics, 3:143-150, 2002.