Neural Network Language Model for Chinese Pinyin Input Method Engine

Shen-Yuan Chen[1,2], **Rui Wang**[1,2] **and Hai Zhao**[1,2*]
[1]Center for Brain-Like Computing and Machine Intelligence,
Department of Computer Science and Engineering,
Shanghai Jiao Tong University, Shanghai 200240, China
[2]Key Laboratory of Shanghai Education Commission for Intelligent Interaction
and Cognitive Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China
chenshenyuan1992@sina.com, wangrui.nlp@gmail.com
and zhaohai@cs.sjtu.edu.cn

Abstract

Neural network language models (NNLMs) have been shown to outperform traditional n-gram language model. However, too high computational cost of NNLMs becomes the main obstacle of directly integrating it into pinyin IME that normally requires a real-time response. In this paper, an efficient solution is proposed by converting NNLMs into back-off n-gram language models, and we integrate the converted NNLM into pinyin IME. Our experimental results show that the proposed method gives better decoding predictive performance for pinyin IME with satisfied efficiency.

1 Introduction

Input method engine (IME) is the encoding method to input a variety of characters into computer or other information processing and communication devices, such as mobile phone. People working with computer cannot live without IMEs. With the development and improvement of IMEs, people are paying more and more attention to its efficiency and humanization. Since there are thousands of Chinese characters and only 26 English letters on standard keyboard, we cannot directly input the Chinese characters to the computer by simply hitting keys. A mapping or encoding from Chinese characters to English letters is required, and IME is such a system software to do the mapping (Chen and Lee, 2000; Wu et al., 2009; Zhao et al., 2006). Among various encoding schemes, most IMEs adopt Chinese pinyin, which is the phonetic representation of Chinese characters through Latin (English) letters. The advantage of pinyin IMEs is that it only requires for knowledge of pinyin rules, which are known by most educated Chinese users. Being easy to learn and use, pinyin IME is becoming the first choice of more and more Chinese users.

However, there are only under 500 pinyin syllables in standard Chinese, mandarin, but over 6,000 common Chinese characters, bringing huge ambiguities for pinyin-to-characters mapping (Jia and Zhao, 2014; Xu and Zhao, 2012; Zhang et al., 2014). Modern pinyin IMEs commonly use sentences-based decoding method (Chen and Lee, 2000; Zhang et al., 2012), that is, generating a Chinese sentence according to a pinyin sequence for disambiguation. The decoding method usually works with the help of statistic language model or other techniques.

Back-off N-gram language models (BNLMs) (Chen and Goodman, 1996; Chen and Goodman, 1999; Stolcke, 2002a) have been broadly used for pinyin IME because of their simplicity and efficiency. Recently, continuous-space language models (CLSMs), especially neural network language models (NNLMs) (Bengio et al., 2003; Schwenk, 2007; Le et al., 2010) are used in more and more natural language processing tasks, and practically outperform BNLMs in prediction accuracy. However, NNLMs are still out of consideration for IMEs according to our best knowledge. The main obstacle about using NNLMs in IME is that it is too time-consuming to meet the requirement from IME that needs a real-time response as human-computer interface.

[*]Corresponding author

29th Pacific Asia Conference on Language, Information and Computation pages 455 - 461
Shanghai, China, October 30 - November 1, 2015

Actually, too long computational time makes the direct integration of NNLMs infeasible for pinyin IMEs. We can hardly image that users have to wait for over 10 seconds to get the result after typing the pinyin sequence. So we have to find another way. Although some work have reduced the decoding time of NNLMs, such as (Arisoy et al., 2014), (Vaswani et al., 2013) and (Devlin et al., 2014), these methods are mainly designed for speech recognition or machine translation and can not be integrated into IME directly.

Instead of replacing BNLMs with NNLMs in IME, we propose to use NNLMs to enhance BNLMs, which means recalculating the probabilities of n-grams in the BNLMs with NNLMs. Thus we take advantage of the probabilities provided by NNLMs without increasing its on-site computational cost. Furthermore, we will also demonstrate that our method may indeed improve the prediction performance of pinyin IMEs.

In Section 2, we introduce the typical pinyin IME model and explain how language models work on the process of candidate sentence generation. In Section 3, we describe the basic concept of BNLMs and NNLMs, then we describe our approach of converting the NNLMs into BNLMs. In section 4 and 5, we do experiments and conclude.

2 Pinyin IME Model

Basically the core engine of IME is a pipeline of three parts: pinyin segmentation, candidate words fetching and candidate sentence generation.

Pinyin segmentation is a word segmentation task which may not be as typical as Chinese word segmentation. Since pinyin has a very small vocabulary that contains about 500 legal pinyin syllables, rule-based segmentation methods are widely used, i.e., backward maximum matching algorithm with additional rules (Goh et al., 2005). Carefully written rules can deal with most of the ambiguous conditions.

Pinyin segmentation breaks continuous user input into separated pinyin syllables and passes them on to the next stage, candidate words fetching. It is a table look-up task finding Chinese words corresponding to pinyin syllables. A table of candidate words is built according to pinyin syllables. Each column of the table is the words corresponding to the syllable and sorted by its probability of existing.

The most important part of pinyin IME is candidate sentence generation, into which we put much of our effort. Language model is commonly used for generating the most likely sentence through providing a conditional probability of words by its context (Chen and Lee, 2000; Zhao et al., 2013). So sentence generation is to find the sentence with the maximum probability, which is constructed by the candidate words fetched previously.

Candidate sentence generation can be regarded as a decoding problem. The goal is to find the most likely Chinese word sequence W^* with the highest joint probability that

$$\begin{aligned} W^* &= \arg\max_W P(W|S) \\ &= \arg\max_W \frac{P(W)P(S|W)}{P(S)} \\ &= \arg\max_W P(W)P(S|W) \\ &= \arg\max_{w_1,w_2,\ldots,w_M} \prod_{w_i} P(w_i|w_{i-1}) \prod_{w_i} P(s_i|w_i) \end{aligned}$$

where $P(s_i|w_i)$ is the conditional probability mapping a word to its pinyin syllable, and $P(w_i|w_{i-1})$ is the transition probability. Here $P(s_i|w_i)$ is 1 while the word w_i is corresponding to the pinyin syllable w_i and 0 otherwise. Note that practically the transition probability is $P(w_i|w_{i-(n-1)},...,w_{i-1})$ provided by language model. We use Viterbi algorithm (Viterbi, 1967) to decode the character sentence. This problem is same as finding the shortest path on the candidate word lattice (Forney, 1973), which is shown in Figure 1, with all probabilities determined by the language models. Hence, we have reasons to believe that a better language model makes better candidate sentences.

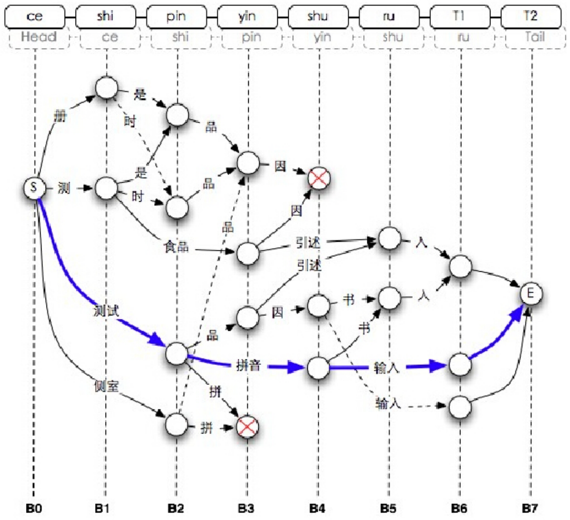

Figure 1: Candidate Sentence Generation

3 Our Approach

This section will introduce the proposed method that uses NNLMs to enhance BNLMs.

3.1 Back-off n-gram language model

A BNLM is composed of the condition probabilities $P(w_i|h_i)$, which means the probability of word w_i given the previous history h_i of $n-1$ words. Its advantage comes from its simplicity. However, it suffers from data sparseness, that is, the probability of the history h_i not appearing in the training will be zero. Therefore, we need smoothing technologies to alleviate this shortcoming. In the case of interpolated Kneser-Ney smoothing (Chen and Goodman, 1999), the probability under BNLM, $P_b(w_i|h_i)$, is:

$$P_b(w_i|h_i) = \hat{P}_b(w_i|h_i) + \gamma(h_i)P_b(w_i|w_{i-n+2}^{i-1})$$

where $\hat{P}_b(w_i|h_i)$ is a discounted probability and $\gamma(h_i)$ is the back-off weight.

3.2 Neural network language model

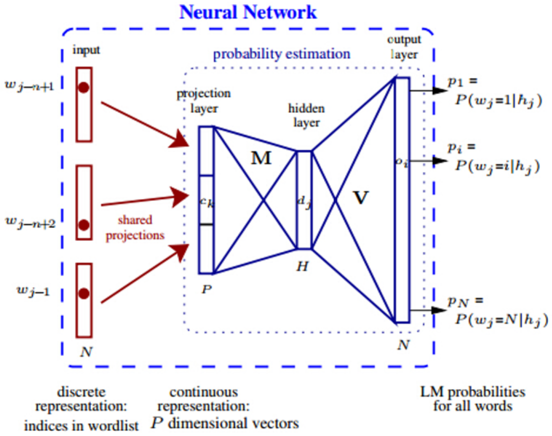

Figure 2: Neural Network Language Model

NNLM provides the condition probabilities $P(w_i|h_i)$ given the histories of the preceding words. , which is shown in Figure 2 (Schwenk, 2013) A typical NNLM consists of four layers: a input layer, a projection layer, a hidden layer and a output layer. The input layer represents the history of $n-1$ words, using one hot coding. Since this kind of coding makes the input layer very large and sparse, a shared matrix is used to project the high-dimensional input layer to the low dimensional projection layer. After that, the non-linear (commonly sigmoid or hyperbolic tangent) hidden layer, with usually hundreds of neurons, and the non-linear (commonly softmax) output layers, with the same size as the full vocabulary, jointly calculate the probability of each word in the vocabulary (Schwenk, 2007; Wang et al., 2013b; Wang et al., 2013c).

Since NNLMs calculate the probabilities of n-grams on the continuous space, it can estimate probabilities for any possible n-grams, without worrying about the zero-probability problem, in comparison with BNLM. Hence, there is no need for backing-off to smaller history. Experiments have shown that the NNLM has lower perplexity than the BNLM trained on the same corpus (Schwenk, 2010; Huang et al., 2013). However, the computational complexity of NNLMs is quite high. To reduce the computational costs, NNLMs are only used to compute the probabilities for the subset containing the most frequent words in the vocabulary, called short-list (Schwenk, 2007; Schwenk, 2010); the probabilities of the rest words are given by BNLMs. The probability $P(w_i|h_i)$ using short-list is calculated as follows:

$$P(w_i|h_i) = \begin{cases} \frac{P_c(w_i|h_i)P_s(h_i)}{1-P_c(o|h_i)} & \text{if } w_i \in \text{short-list} \\ P_b(w_i|h_i) & \text{otherwise} \end{cases}$$

where $P_c(\cdot)$ is the probability calculated by NNLM, $P_c(o|h_i)$ is given by the neuron assigned for the words not in the short-list, $P_b(\cdot)$ is calculated by BNLM, and

$$P_s(h_i) = \sum_{v \in \text{short-list}} P_b(v|h_i)$$

In view of the pros and cons of NNLMs, we tried another way to use NNLMs in pinyin IME as the following section.

3.3 NNLM-enhanced BNLM

The main disadvantage of NNLMs is the large computational cost. During the process of pinyin IME, the language model needs to be frequently accessed, so simply replacing the BNLMs with NNLMs will make the process be very slow. This problem is especially serious to IMEs, which request for fast response. This is why it is infeasible to directly integrate the NNLMs into pinyin IME.

To solve this problem, we propose a method to efficiently apply NNLMs. In the case of a particular NNLM, it will provide the same probability distribution given the same input history. Thus, we can use NNLMs to calculate the probabilities of all the possible n-grams in advance and save the results, which is just like constructing the BNLMs from CSLMs. Our approach is as follow: First, a BNLM and a NNLM are respectively trained on the same corpus. Second, we extract all the n-gram from the BNLM and calculate the probability of them with the NNLM. Third, we re-write the BNLM with the probability computed by NNLM. Finally, we re-normalize the probabilities of BNLM. In this way, we convert the NNLM to the BNLM [1], which is a "conditional-probabilities-to-conditional-probabilities" adjustment. We may now use the NNLM-enhanced BNLM in the pinyin IME.

4 Experiment

4.1 Commen Settings

We use the Chinese corpus from (Yang et al., 2012), which is extracted from the corpus of *People's Daily*. The sentences are already segmented into words and labeled with pinyin. The corpus is divided into training set, development set and testing set, which are shown in Table 1.

	Train	Dev	Test
Sentence	1M	2K	100K
Character	43,679,593	83,765	4,123,184

Table 1: Data set size.

In consideration of the model size and efficiency, most existing IMEs use a standard trigram setting for language model (Chen and Lee, 2000), therefore we evaluate the proposed method by following the setting. We first trained a trigram BNLM as the baseline with interpolated Kneser-Ney smoothing, using SRILM toolkit (Stolcke, 2002b). Note that SRILM is also used to re-normalize the re-written BNLMs. We then trained a trigram NNLM on the same training data, using CSLM toolkit (Schwenk, 2013). The vocabulary was extracted from (Wang et al., 2013a; Wang et al., 2014) with the size of 914,728 words, and were labeled with pinyin using the Pinyin-To-Chinese dictionary of *sunpinyin* [2], an open source pinyin IME. In addition, the re-written BNLM is interpolated with the baseline BNLM, using the interpolation weight 0.5, which is empirically tuned using development data.

4.2 Running Time

This subsection compares running time for different types of language models. First, we run the task of calculating the probabilities of 7 million trigrams using different language models to compare their time performance, each for 3 times, as shown in Table 2. The running time of CSLM is up to 100 times greater than our model, which is definitely unacceptable to pinyin IMEs. Besides, since our model has

[1] Arsoy and Wang also proposed NNLM converting methods, however their methods are specially applied to speech recognition or machine translation.

[2] http://code.google.com/p/sunpinyin

the same structure as the BNLM, extra time cost is not requested. [3]

LMS	Run Time1	Run Time2	Run Time3
BNLM	17.9s	16.7s	16.9s
NNLM	1,684.5s	1,684.5s	1,683.9s
Our LM	17.0s	16.8s	16.5s

Table 2: Running Time of Language models.

4.3 Language Model Performance

We compare the perplexity of our model with the baseline BNLM. Table 3 show that our model outperforms the baseline BNLM in perplexity.

Language Model	perplexity
Baseline (trigram BNLM)	202.5
NNLM-enhanced trigram BNLM	196.4

Table 3: Perplexity of Language models.

However, the lower perplexity does not inevitably equal to the better performance in pinyin IME. We still need to integrate the our model into the pinyin IME to evaluate its actual effect.

4.4 Pinyin IME Performance

To evalue the performance of pinyin IME. We use hit rate of the first candidate sentence (HRF), hit rate of the first k (here $k = 10$) candidate sentences (HRF10) and character accuracy of the first candidate sentence (CA) as evaluation measures. The result is shown in Table 4:

Test	Models	HRF	HRF10	CA
10K	Baseline	74.72	89.92	96.80
10K	Our model	75.71	90.14	96.80
400K	Baseline	67.02	86.08	95.46
400K	Our model	67.68	86.45	95.59

Table 4: Pinyin IME performance(%).

The experimental results show that our method can effectively improve the performance of pinyin IME. Figures 3 and 4 indicate that the improvement of the pinyin IME with our model is particularly significant when the length of the input pinyin sequence is between 10-30 characters, meanwhile almost 69% of the sentences in the corpus are with the length of over 10 pinyin characters, which means our method perform better in most cases, especially for the long pinyin sequences. Note that since we only change the probabilities in the BNLM, the improvement does not call for extra time cost.

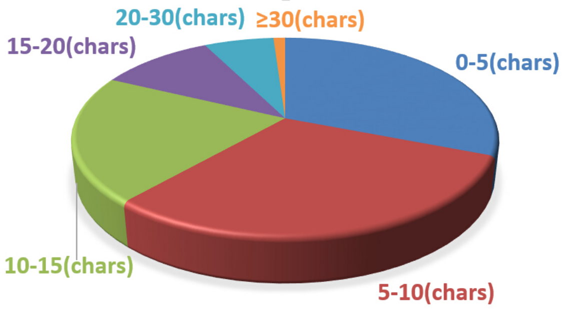

Figure 3: The length distribution of pinyin sequences in testing set

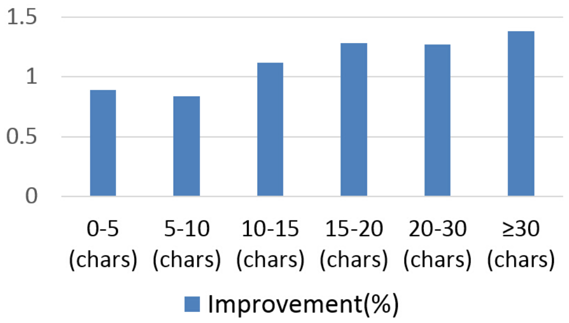

Figure 4: The improvement of HRF using our approach

5 Conclusion

In this paper, we propose an efficient way to apply NNLM to pinyin IME. The experiments demonstrate that our method can effectively improve the predictive performance of pinyin IME without extra time cost.

6 Acknowledge

We appreciate the anonymous reviewers for valuable comments and suggestions on our paper. Shen-

[3] Here only language model computation cost is demonstrated. For decoding a sentence in IME, generally, thousands of language model accessing is usually required. Therefore, according to the above empirical results, integrating NNLM into IME without any modification will result in obvious response retardation, which gives a poor user experience.

yuan Chen, Rui Wang and Hai Zhao were partially supported by the National Natural Science Foundation of China (No. 60903119, No. 61170114, and No. 61272248), the National Basic Research Program of China (No. 2013CB329401), the Science and Technology Commission of Shanghai Municipality (No. 13511500200), the European Union Seventh Framework Program (No. 247619), the Cai Yuanpei Program (CSC fund 201304490199 and 201304490171), and the art and science interdiscipline funds of Shanghai Jiao Tong University(A study on mobilization mechanism and alerting threshold setting for online community, and media image and psychology evaluation: a computational intelligence approach).

References

Ebru Arisoy, Stanley F. Chen, Bhuvana Ramabhadran, and Abhinav Sethy. 2014. Converting neural network language models into back-off language models for efficient decoding in automatic speech recognition. *IEEE/ACM Transactions on, Audio, Speech, and Language Processing*, 22(1):184–192.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, ACL '96, pages 310–318, Santa Cruz, California, June. Association for Computational Linguistics.

Stanley F. Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–393.

Zheng Chen and Kai-Fu Lee. 2000. A New Statistical Approach To Chinese Pinyin Input. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics, ACL*, pages 241–247, Hong Kong, October.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL, (Volume 1: Long Papers)*, pages 1370–1380, Baltimore, Maryland, June. Association for Computational Linguistics.

Jr George David Forney. 1973. The Viterbi Algorithm. *Proceedings of the IEEE*, 61(3):268–278.

Chooi-Ling Goh, Masayuki Asahara, and Yuji Matsumoto. 2005. Chinese word segmentation by classification of characters. *Computational Linguistics and Chinese Language Processing*, 10(3):381–396.

Zhongqiang Huang, Jacob Devlin, Spyros Matsoukas, and Rich Schwartz. 2013. BBN's systems for the chinese-english sub-task of the NTCIR-10 patentmt evaluation. *Proceedings of NII Testbeds and Community for Information access Research 10, NTCIR-10*, pages 287–293.

Zhongye Jia and Hai Zhao. 2014. A joint graph model for pinyin-to-chinese conversion with typo correction. In *Proceedings of Annual Meeting of the Association for Computational Linguistics, ACL*, pages 1512–1523, Baltimore, Maryland, June.

Hai-son Le, Alexandre Allauzen, Guillaume Wisniewski, and François Yvon. 2010. Training continuous space language models: some practical issues. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 778–788, Cambridge, Massachusetts, October. Association for Computational Linguistics.

Holger Schwenk. 2007. Continuous space language models. *Computer Speech & Language*, 21(3):492–518.

Holger Schwenk. 2010. Continuous-space language models for statistical machine translation. *The Prague Bulletin of Mathematical Linguistics*, 93:137–146.

Holger Schwenk. 2013. CSLM - a modular open-source continuous space language modeling toolkit. In *Interspeech*, pages 1198–1202.

Andreas Stolcke. 2002a. SRILM-An Extensible Language Modeling Toolkit. In *Proceedings of the international conference on spoken language processing*, volume 2, pages 901–904.

Andreas Stolcke. 2002b. Srilm-an extensible language modeling toolkit. In *Proceedings of International Conference on Spoken Language Processing, ICSLP*, pages 257–286, Seattle, USA, November.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 1387–1392, Seattle, Washington, USA, October. Association for Computational Linguistics.

Andrew James Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269.

Peilu Wang, Ruihua Sun, Hai Zhao, and Kai Yu. 2013a. A New Word Language Model Evaluation Metric for Character Based Languages. In *Chinese Computational Linguistics and Natural Language Processing*

Based on Naturally Annotated Big Data, pages 315–324. Springer.

Rui Wang, Masao Utiyama, Isao Goto, Eiichro Sumita, Hai Zhao, and Bao-Liang Lu. 2013b. Converting Continuous-Space Language Models into N-Gram Language Models for Statistical Machine Translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 845–850, Seattle, Washington, USA, October. Association for Computational Linguistics.

Rui Wang, Hai Zhao, Bao-Liang Lu, Masao Utiyama, and Eiichiro Sumita. 2013c. Bilingual continuous-space language model growing for statistical machine translation.

Rui Wang, Hai Zhao, Bao-Liang Lu, Masao Utiyama, and Eiichro Sumita. 2014. Neural network based bilingual language model growing for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 189–195.

Jun Wu, Huican Zhu, and Hongjun Zhu. 2009. Systems and Methods for Translating Chinese Pinyin to Chinese Characters, January 13. US Patent 7,478,033.

Qiongkai Xu and Hai Zhao. 2012. Using deep linguistic features for finding deceptive opinion spam. In *COLING (Posters)*, pages 1341–1350. Citeseer.

Shaohua Yang, Hai Zhao, and Bao-liang Lu. 2012. A Machine Translation Approach for Chinese Whole-Sentence Pinyin-to-Character Conversion. In *Proceedings of the 26th Pacific Asia Conference on Language, Information, and Computation, PACLIC*, pages 333–342, Bali,Indonesia, November. Faculty of Computer Science, Universitas Indonesia.

Xiaotian Zhang, Hai Zhao, and Cong Hui. 2012. A machine learning approach to convert ccgbank to penn treebank. In *COLING (Demos)*, pages 535–542.

Jingyi Zhang, Masao Utiyama, Eiichro Sumita, and Hai Zhao. 2014. Learning hierarchical translation spans. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 183–188.

Hai Zhao, Chang-Ning Huang, and Mu Li. 2006. An improved chinese word segmentation system with conditional random field. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, volume 1082117. Sydney: July.

Hai Zhao, Masao Utiyama, Eiichiro Sumita, and Bao-Liang Lu. 2013. An Empirical Study on Word Segmentation for Chinese Machine Translation. In *Computational Linguistics and Intelligent Text Processing*, pages 248–263. Springer.