A Bi-directional LSTM Approach for Polyphone Disambiguation in Mandarin Chinese

Changhao Shan[1]*, Lei Xie*[1]*, Kaisheng Yao*[2]

[1]School of Computer Science, Northwestern Polytechnical University, Xi'an, China
[2]Microsoft Corporation, Redmond, 98052, WA, USA

{chshan, lxie}@nwpu-aslp.org, kaisheny@microsoft.com

Abstract

Polyphone disambiguation in Mandarin Chinese aims to pick up the correct pronunciation from several candidates for a polyphonic character. It serves as an essential component in human language technologies such as text-to-speech synthesis. Since the pronunciation for most polyphonic characters can be easily decided from their contexts in the text, in this paper, we address the polyphone disambiguation problem as a sequential labeling task. Specifically, we propose to use bidirectional long short-term memory (BLSTM) neural network to encode both the past and future observations on the character sequence as its inputs and predict the pronunciations. We also empirically study the impacts of (1) modeling different length of contexts, (2) the number of BLSTM layers and (3) the granularity of part-o-speech (POS) tags as features. Our results show that using a deep BLSTM is able to achieve state-of-the-art performance in polyphone disambiguation.

Index Terms: Polyphone disambiguation, Grapheme-to-phoneme conversion, Sequence tagging, Bi-directional LSTM, Text-to-Speech

1. Introduction

Grapheme-to-phoneme (G2P) conversion aims to predict the pronunciation of a word given its orthography, i.e., a series of characters or graphemes. It is an essential component in human language technologies, especially in speech synthesis and speech recognition. In an alphabetic language such as English, the main problem a G2P module faces is to generate pronunciation for out-of-the-vocabulary (OOV) words [1, 2, 3, 4, 5, 6, 7]. However, in Chinese, a character-based language, most characters have only one fixed pronunciation and each character is pronounced as a tonal syllable. So the difficulty in Chinese G2P conversion is *polyphone disambiguation*, which aims to pick out one correct pronunciation from several candidates for a polyphonic character [8].

A variety of approaches have been proposed to address the polyphone disambiguation problem. They can be categorized into *knowledge-based* and *learning-based* approaches. A rich pronunciation dictionary and human rules are essential in a knowledge-based system. The dictionary is designed to list as many as possible the words with polyphonic characters and their pronunciations [9]. But the dictionary cannot cover all the polyphonic cases in the language. Hence some rules are crafted by language experts to handle these cases. During the runtime, the pronunciation of a polyphone is firstly searched in the dictionary. If not found, look up the manual rules to determine the pronunciation.

Knowledge-based approach heavily relies on human expertise, while learning-based approach aims to automatically learn a polyphone disambiguation model from a set of data. Similar to G2P conversion in English, a joint n-gram model can be used in polyphone disambiguation. Because of the relatively small pronunciation cardinality of polyphonic characters, i.e., two to four, n-gram statistics can be reliably obtained from a training set, leading to reasonable performance. Such n-gram models are usually implemented as a weighted finite state transducer (WFST). Polyphone disambiguation can be treated as a classification task. Based on a set of features extracted for a polyphone, its pronunciation is predicted from a set of candidates through a decision tree (DT) [10] or a maximum entropy (Maxent) model [11]. A study has shown that a Maxent model outperforms DT in polyphone disambiguation [11]. A learning-based approach can be integrated with a knowledge-based approach to form a hybrid approach [12], where most of the polyphones are disambiguated by a learned model, but the pronunciations of some polyphonic characters are determined by human rules.

Since the pronunciation for most polyphonic characters can be easily decided from their contexts, in this paper, we address the polyphone disambiguation task as a sequential labeling (or tagging) task that models the important contextual information. Specifically, we propose to use *bidirectional long short-term memory* (BLSTM) neural network in pronunciation determination of polyphones in Mandarin Chinese. Our approach is motivated by recent tremendous success of BLSTM models in English G2P conversion [1] [13] and various sequential learning tasks [14, 15, 16, 17]. LSTM uses specifically designed gates to control information flow and thus has exceptional context modeling ability [18]. BLSTM is composed of a forward LSTM and a backward LSTM, and thus it can model both the past and the future contexts. However, the LSTM models used in English G2P is not readily applicable to polyphone disambiguation in Mandarin. Firstly, besides the tremendous difference between the two languages, we shall determine the best context to be modeled in the task. Secondly, recent studies suggest that using multiple hidden layers can learn hierarchical features and boost performances. We would like to investigate whether using deep BLSTM architecture can benefit this task. Our results show that using a deep bidirectional LSTM is able to achieve state-of-the-art performance in polyphone disambiguation.

2. Features

In Chinese, the pronunciation for a polyphonic character usually can be determined by the word that contains it and its neighboring words. For the examples in Table 1, we can eas-

Table 1: *Examples showing that the word that contains the polyphone and its POS tag are important features for polyphone disambiguation. The polyphone is in bold font.*

Word	Pronunciation	POS	English translation
朝暮	zhao1	n	**morning** and night
朝阳	chao2	ns	**Chao**yang (a place name)
朝阳	zhao1	n	**morning** sun

ily get the pronunciation of polyphone '朝', if the word that contains the polyphone is shown. Hence the identity of the word that contains the polyphone is an important feature. However, in Table 1, we still cannot make a distinction between "朝(zhao1)阳" and "朝(chao2)阳". But the two words can be discriminated (and the pronunciation of the polyphone '朝' can be determined) if we have the part-of-speech (POS) information. If the word is tagged as 'ns', then the polyphonic character '朝' in the word "朝阳" is pronounced as 'chao2'. If tagged as 'n', then '朝' in the word "朝阳" is pronounced as 'zhao1'. Thus POS tag is an important feature for polyphone disambiguation. Therefore, we simply employ the identity of the word that contains the polyphone and it's POS tag as features.

Table 2 shows that the left and right contexts are also quite useful for polyphone disambiguation. In this example, the character '转' can be used as two verbs with both different meanings and different pronunciations. But the pronunciations of the two '转' can be easily discriminated if we observe their contexts (different POS tags). This is also the reason that motivates us to use a recurrent network to modeling the contexts in the polyphone disambiguation task.

Table 2: *Examples showing that contexts are useful for polyphone disambiguation. The polyphone is in bold font.*

Sequence	Pronunciation	English translation
玩_v **转**_v 北京_ns	zhuan4	**play around** in Beijing
汉字_nz **转**_v 拼音_n	zhuan3	**convert** Hanzi to Pinyin

3. Model

3.1. LSTM

We treat polyphone disambiguation as a sequence tagging task. In the G2P conversion task [1, 2, 3, 4, 5, 6, 7], a neural network accepts a sequence of characters and outputs a sequence of pronunciations. While in polyphone disambiguation of Mandarin Chinese, the input is a sequence of characters with one or more polyphonic characters inside, and the outputs for the polyphonic character and other non-polyphonic character are the predicted pronunciation and a NULL symbol ('-'), respectively, as shown in Table 3.

Table 3: *Treating polyphone disambiguation as a sequence tagging task. The input is a sequence of character. If the character is polyphone, the output is its predicted pronunciation; otherwise, the output is a NULl symbol '-'.*

Input character	我	在	古	都	呢
Output pronunciation	−	−	−	$du1$	−

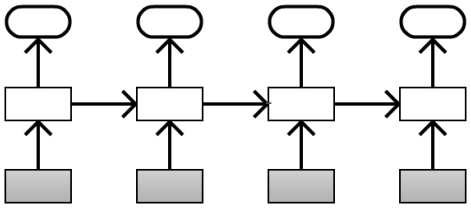

Figure 1: *Recurrent neural network unrolled in time.*

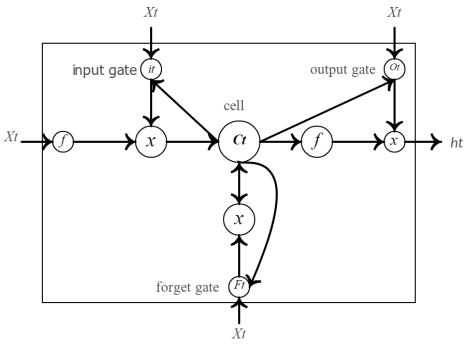

Figure 2: *A single LSTM memory cell with different gates to control information flow.*

Specifically, we use LSTM recurrent neural network (RNN) to do the sequence tagging. Allowing cyclical connections in a feed-forward neural network, we obtain a *recurrent* network, as shown in Figure 1. RNNs, especially those networks with LSTM cells, have recently produced promising results on a variety of tasks including language modeling [19] [20], speech recognition [21] and other sequential labelling tasks [14, 15, 16, 17, 22, 23, 24]. LSTM [18] uses purpose-built memory cells to store information, which is designed to model a long range of context. LSTM is composed of a set of recurrently connected memory blocks and each block consists of one or more self-connected memory cells and three multiplicative gates, i.e., input gate, forget gate and output gate, as shown in Figure 2. The three gates are designed to capture long-range contextual information by using nonlinear summation units. For LSTM, the recurrent hidden layer function is implemented as follows:

$$
\begin{aligned}
\boldsymbol{i}_t &= \sigma(\boldsymbol{W}_{xi}\boldsymbol{x}_t + \boldsymbol{W}_{hi}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{ci}\boldsymbol{c}_{t-1} + \boldsymbol{b}_i) \\
\boldsymbol{f}_t &= \sigma(\boldsymbol{W}_{xf}\boldsymbol{x}_t + \boldsymbol{W}_{hf}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{cf}c_{t-1} + \boldsymbol{b}_f) \\
\boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tanh(\boldsymbol{W}_{xc}\boldsymbol{x}_t + \boldsymbol{W}_{hc}\boldsymbol{h}_{t-1} + \boldsymbol{b}_c) \\
\boldsymbol{o}_t &= \sigma(\boldsymbol{W}_{xo}\boldsymbol{x}_t + \boldsymbol{W}_{ho}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{co}\boldsymbol{c}_t + \boldsymbol{b}_o) \\
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t)
\end{aligned}
$$

where $\boldsymbol{x}_t$ is the input feature vector; σ is the element-wise logistic sigmoid function; $\boldsymbol{i}$, $\boldsymbol{f}$, $\boldsymbol{o}$ and $\boldsymbol{c}$ denote the input gate, forget gate, output gate and memory cell respectively, and all of them are the same size as the LSTM output vector $\boldsymbol{h}$; $\boldsymbol{W}_{xi}$ is the input-input gate matrix, $\boldsymbol{W}_{hc}$ is the hidden-cell matrix, and so on; $\odot$ is the element-wise product.

3.2. Bidirectional LSTM

One shortcoming of LSTM is that it is *unidirectional*: it only makes use of previous context. But in polyphone disam-

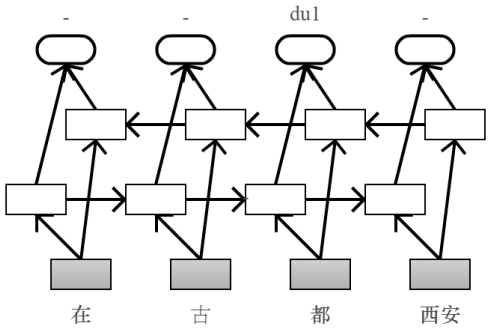

Figure 3: *A bi-directional LSTM network reads "在 古 都 西安" for the forward directional LSTM, the time-revised sequence "西安 都 古 在" for the backward directional LSTM.*

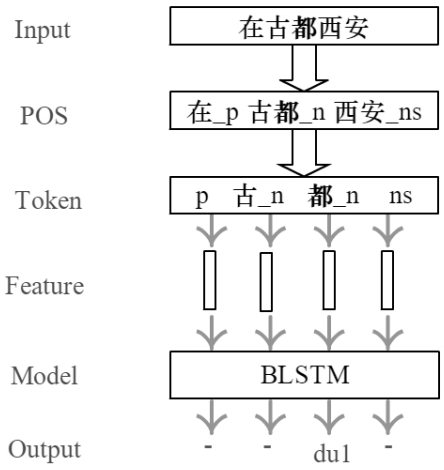

Figure 4: *The flow digram of the LSTM-based polyphone disambiguation system.*

biguation, the past and future contexts are both important (as can be seen in Section 4.3). Thus bidirectional LSTM (BLSTM) architecture is used for polyphone disambiguation, as shown in Figure 3. BLSTM consists of a forward LSTM and a backward LSTM, and the outputs of the two sub-networks are then combined [21]. Given an input sequence $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)$, the forward LSTM reads it from left to right, but the backward LSTM reads it in a reversed order. These two networks have different parameters. BLSTM can utilize both past inputs and future inputs for a specific time.

3.3. Polyphone disambiguation using BLSTM

Figure 4 shows the flow digram of the LSTM-based polyphone disambiguation system. Given an input character sequence, firstly we perform word segmentation and POS tagging. In this study, we put the word containing the polyphonic character as the center of the sequence and consider the word's left and right neighboring words as contexts. In the example shown in Figure 4, "古都" is the centering word containing the polyphone '都', while its left and right neighbors are words '在' and "西安". From the POS tag sequence, we then generate a *token* sequence, in which the center word is separated into several tokens if it has multiple characters. In the example, " 古都_n" is divided into " 古_n" and "都_n". In the token sequence, the left and right words are treated as single tokens (do not separate if have multiple characters) and we only use their POS tags as features. We found that the POS tags of neighboring words are more useful for polyphone disambiguation. The token sequence is then represented as a feature vector sequence. Each feature vector has a character identity sub-vector, a polyphone identity sub-vector and a POS tag sub-vector. Finally the feature vector sequence is fed into the BLSTM network, resulting in a pronunciation sequence with the prediction of the polyphone. The network output is represented by a posterior vector which is composed of all possible pronunciations of the considered polyphones (in this study 79 polyphones with 186 pronunciations) and a NULL label. We pick up the pronunciation with the highest posterior as the result.

4. Experiments

4.1. Dataset

We choose 79 most frequently used polyphones for the polyphone disambiguation experiments. We crawl 174899 sentences from the Internet and manually label the pronunciations (i.e., Pinyin) of the 79 polyphonic characters appeared in these sentences. We manually divide the corpus into a training set with 167221 sentences (179410 polyphonic characters) and a test set with 7678 sentences (10500 polyphonic characters).

4.2. Experimental setups

We use the NLPIR toolkit [25] to perform the POS tagging on the text. The Kaldi toolkit [26] is adopted to implement the neural networks. To speed-up training, we use data parallelism with 512 character sequences per mini-batch. Our networks have hybrid structures with one feed-forward layer sitting on top of K BLSTM layers, where the best K is empirically selected through experiments. Our empirical experiments show that this network structure can ensure the training steadily converges. The number of nodes in each hidden layer is set to 512. The hidden activation function is sigmoid and the output layer activation function is softmax. We use cross-entropy as the error function in the training. The initial learning rate is set to 0.01, but we halve the learning rate if there is no improvement in the cross-entropy score. We also realize a joint n-gram approach [2] (n=2) and a Maxent approach [11] for comparison. The training and test sets are kept the same with those in the BLSTM model training.

4.3. Modeling context

We investigate the effects from different contextual inputs, and the results are summarized in Table 4. In this experiment, we use two LSTM/BLSTM layers ($K = 2$) and test different contexts. BLSTM can model both past and future contexts, while forward and backward (unidirectional) LSTMs can model past-only and future-only contexts, respectively. From the results, we can clearly see that, without the past and future inputs (0 words), the accuracy of BLSTM degrades to the same level with the joint n-gram approach and unidirectional LSTMs have even lower accuracy. The best performances are achieved by the context of 1 words. But further expanding the context to 2 words results in clear accuracy degradations. BLSTM shows consistently better accuracy over unidirectional LSTMs. This observation shows that the use of both past and future contexts

Table 4: *Polyphone disambiguation accuracy for LSTM with different contextual inputs.*

Context	Forward LSTM	Backward LSTM	BLSTM
0 words	85.07	87.26	89.64
1 words	**88.56**	**90.78**	**93.83**
2 words	87.42	89.98	92.96

Table 5: *Polyphone disambiguation accuracy for BLSTM with different layers.*

Layer	Acc (%)
1	93.71
2	**93.83**
3	93.73

is essential in polyphone disambiguation.

4.4. Number of BLSTM Layers

We further test networks with different BLSTM layers. In this experiment, the input context is set to ± 1 words. Results are shown in Table 5. We notice that one layer BLSTM can already achieve a good performance, but the best performance is achieved by a *deeper* network with two BLSTM layers. Further deepening the network to 3 layers has negative effect. We believe that this may be caused by the limited data.

4.5. POS tagging granularity

As we discussed in Section 2, POS tags are critical features for polyphone disambiguation. So we examine the impacts from POS tagging granularity in the BLSTM-based polyphone disambiguation task. In the experiments, we use a 2-layer BLSTM (i.e., K=2) and the context of ± 1 words. We study two sets of POS tags as the network input. Table 6 provides the accuracy for two POS tagging tools, i.e., LTP [27] and NLPIR [25]. The LTP tagger outputs 28 different POS tags while the NLPIR outputs 90 different POS tags. We can clearly observe that the BLSTM with NLPIR tags outperforms the BLSTM with LTP tags. This means a finer granularity in POS tagging can lead to better polyphone disambiguation performance.

Table 6: *Polyphone disambiguation accuracy for two POS granularity (28 vs. 90) in the BLSTM-based approach.*

Tool (POS granularity)	Acc (%)
LTP (28)	93.65
NLPIR (90)	**93.83**

4.6. Comparison with other approaches

Polyphone disambiguation accuracy of different approaches is summarized in Table 7. In BLSTM, K is set to 2 and the context is set to ± 1 words. From Table 7, we clearly see that the BLSTM approach significantly outperforms the joint n-gram approach [2] and the Maxent approach [11] (The feature is similar to BLSTM). The differences are significant at 95% confident level with paired t-tests. The relative accuracy improvements are 4.7% and 5.5% as compared with the joint n-gram approach and the Maxent approach, respectively.

Table 7: *Polyphone disambiguation accuracy for BLSTM and other approaches for comparison.*

Approach	Acc (%)
Joint n-gram [2]	89.60
Maxent [11]	88.96
BLSTM	**93.83**

5. Conclusion

In this paper, we address the polyphone disambiguation problem as a sequential labeling task. Specifically, we propose to use bidirectional long short-term memory (BLSTM) neural network to encode both the past and future observations on the character sequence as its inputs and predict the pronunciations. Our conclusions are as follows. 1) By modeling both past and future contexts of inputs, bidirectional LSTM significantly outperforms unidirectional LSTM in polyphone disambiguation. 2) A 2-layer BLSTM model achieves superior performance. 3) A finer granularity in POS tagging is able to lead to better performance. We have observed relative accuracy improvements of 4.7% and 5.5% as compared with the joint n-gram approach and the Maxent approach, respectively. In future, it may be interesting to investigate if a simple recurrent neural network can achieve similar performance.

6. Acknowledgements

This work was supported by the National Natural Science Foundation of China(Grant No. 61571363).

7. References

[1] K. Rao, F. Peng, H. Sak, and F. Beaufays, "Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks," in *ICASSP*. IEEE, 2015, pp. 4225–4229.

[2] J. R. Novak, N. Minematsu, and K. Hirose, "Failure transitions for joint n-gram models and g2p conversion." in *INTERSPEECH*, 2013, pp. 1821–1825.

[3] M. Bisani and H. Ney, "Joint-sequence models for grapheme-to-phoneme conversion," *Speech Communication*, vol. 50, no. 5, pp. 434–451, 2008.

[4] S. F. Chen *et al.*, "Conditional and joint models for grapheme-to-phoneme conversion." in *INTERSPEECH*, 2003.

[5] L. Galescu and J. F. Allen, "Bi-directional conversion between graphemes and phonemes using a joint n-gram model," in *4th ISCA Tutorial and Research Workshop (ITRW) on Speech Synthesis*, 2001.

[6] J. R. Novak, N. Minematsu, K. Hirose, C. Hori, H. Kashioka, and P. R. Dixon, "Improving wfst-based g2p conversion with alignment constraints and rnnlm n-best rescoring." in *INTERSPEECH*, 2012, pp. 2526–2529.

[7] D. Caseiro, I. Trancoso, L. Oliveira, and C. Viana, "Grapheme-to-phone using finite state transducers," in *Proc. 2002 IEEE Workshop on Speech Synthesis*, vol. 2, 2002, pp. 1349–1360.

[8] Z.-R. Zhang, M. Chu, and E. Chang, "An efficient way to learn rules for grapheme-to-phoneme conversion in chinese," in *International Symposium on Chinese Spoken Language Processing*, 2002.

[9] D. Gou and W. Luo, "Processing of polyphone character in chinese tts system," *Chinese Information*, no. 1, pp. 33–36.

[10] F. Liu and Y. Zhou, "Polyphone disamblgaatlou based on tree-guided tbl," *Computer Engineering and Application*, vol. 47, no. 12, pp. 137–140, 2011.

[11] F. Liu, Q. Shi, and J. Tao, "Maximum entropy based homograph disambiguation," in *NCMMSC2007*, 2007.

[12] M. Fan, G. Hu, and R. Wang, "Multi-level polyphone disambiguation for mandarin grapheme-phoneme conversion," *Computer Engineering and Application*, vol. 42, no. 2, pp. 167–170, 2006.

[13] K. Yao and G. Zweig, "Sequence-to-sequence neural net models for grapheme-to-phoneme conversion," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[14] K. Yao, B. Peng, Y. Zhang, D. Yu, G. Zweig, and Y. Shi, "Spoken language understanding using long short-term memory neural networks," in *SLT, 2014 IEEE*. IEEE, 2014, pp. 189–194.

[15] Z. Huang, W. Xu, and K. Yu, "Bidirectional lstm-crf models for sequence tagging," *arXiv preprint arXiv:1508.01991*, 2015.

[16] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 6645–6649.

[17] T. Mikolov and G. Zweig, "Context dependent recurrent neural network language model." in *SLT*, 2012, pp. 234–239.

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[19] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, "Recurrent neural network based language model." in *Interspeech*, vol. 2, 2010, p. 3.

[20] T. Mikolov, A. Deoras, D. Povey, L. Burget, and J. Černockỳ, "Strategies for training large scale neural network language models," in *ASRU, 2011 IEEE Workshop on*. IEEE, 2011, pp. 196–201.

[21] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18, no. 5, pp. 602–610, 2005.

[22] K. Yao, B. Peng, G. Zweig, D. Yu, X. Li, and F. Gao, "Recurrent conditional random field for language understanding," in *ICASSP*. IEEE, 2014, pp. 4077–4081.

[23] C. Ding, L. Xie, J. Yan, W. Zhang, and Y. Liu, "Automatic prosody prediction for chinese speech synthesis using blstm-rnn and embedding features," in *ASRU*. IEEE, 2015, pp. 98–102.

[24] O. Tilk and T. Alumäe, "Lstm for punctuation restoration in speech transcripts," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[25] L. Zhou and D. Zhang, "Nlpir: A theoretical framework for applying natural language processing to information retrieval," *Journal of the American Society for Information Science and Technology*, vol. 54, no. 2, pp. 115–123, 2003.

[26] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The kaldi speech recognition toolkit," in *IEEE 2011 Workshop on ASRU*. IEEE Signal Processing Society, Dec. 2011, iEEE Catalog No.: CFP11SRW-USB.

[27] W. Che, Z. Li, and T. Liu, "Ltp: A chinese language technology platform," in *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*. Association for Computational Linguistics, 2010, pp. 13–16.