Subword Encoding in Lattice LSTM for Chinese Word Segmentation

Jie Yang[♠♣], **Yue Zhang**[♦], **Shuailong Liang**[▲]
[♠]Brigham and Women's Hospital. Boston, USA
[♣]Harvard Medical School, Harvard University. Boston, USA
[♦]School of Engineering, Westlake University. Hangzhou, China
[▲]Singapore University of Technology and Design. Singapore
jieynlp@gmail.com
yue.zhang@wias.org.cn
shuailong_liang@mymail.sutd.edu.sg

Abstract

We investigate subword information for Chinese word segmentation, by integrating sub word embeddings trained using byte-pair encoding into a Lattice LSTM (LaLSTM) network over a character sequence. Experiments on standard benchmark show that subword information brings significant gains over strong character-based segmentation models. To our knowledge, this is the first research on the effectiveness of subwords on neural word segmentation.

1 Introduction

Chinese word segmentation (CWS) is a traditional NLP task (Sproat et al., 1996), the features for which have been a central research topic. Statistical methods consider characters (Xue et al., 2003), subwords (Zhang et al., 2006), and words (Zhang and Clark, 2007) as input features. Among these, both characters (Chen et al., 2015a) and words (Zhang et al., 2016; Cai and Zhao, 2016; Yang et al., 2017) have also shown useful in recent neural models. However, how to utilize the subword features in neural networks has not been investigated yet.

In this paper, we fill this gap by proposing a subword-based neural word segmentor, by integrating two strands of works: the byte pair encoding (BPE) algorithm (Gage, 1994) and the lattice LSTM structure (Zhang and Yang, 2018). The BPE algorithm constructs a subword list from raw data and lattice LSTM introduces subwords into character LSTM representation. In particular, our baseline is a BiLSTM-CRF segmentor (Chen et al., 2015b) and we replace LSTM with lattice LSTM using subwords to encode character composition information. Our code[1] is based on NCRF++ (Yang and Zhang, 2018).

[1]Our code is released at https://github.com/jiesutd/SubwordEncoding-CWS.

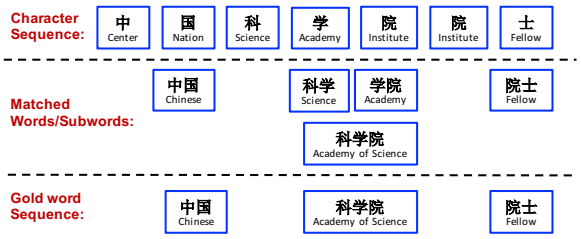

Figure 1: Segmentation with ambiguous words.

Compared with character-based neural segmentors, our model can utilize abundant character combination (subword) information, which is effective to disambiguate characters. For example, in Figure 1, the subword "学 院(Academy)" ensures that the character "学" means "Academy(noun)" rather than "study(verb)". Compared with the word-based neural models (Zhang et al., 2016; Cai and Zhao, 2016), ambiguous subwords in a context can provide additional information for disambiguation. For instance, the subword "科学院(Academy of Sciences)" and "学院(Academy)" can be useful in determining the correct segmentation, which is "科学院/(Academy of Sciences/)".

To our knowledge, we are the first to use subwords in a neural network segmentor. We investigate the contributions of subword lexicons and their pretrained embeddings through controlled experiments. Results on four benchmarks show that the proposed model can give comparable results with state-of-the-art models.

2 Related Work

State-of-the-art statistical segmentors use either sequence labeling methods e.g. CRF (Lafferty et al., 2001) with character features (Peng et al., 2004; Zhao et al., 2006) or the transition-based models with word features (Zhang and Clark, 2007; Sun, 2010). Neural segmentors (Chen et al.,

Proceedings of NAACL-HLT 2019, pages 2720–2725
Minneapolis, Minnesota, June 2 - June 7, 2019.

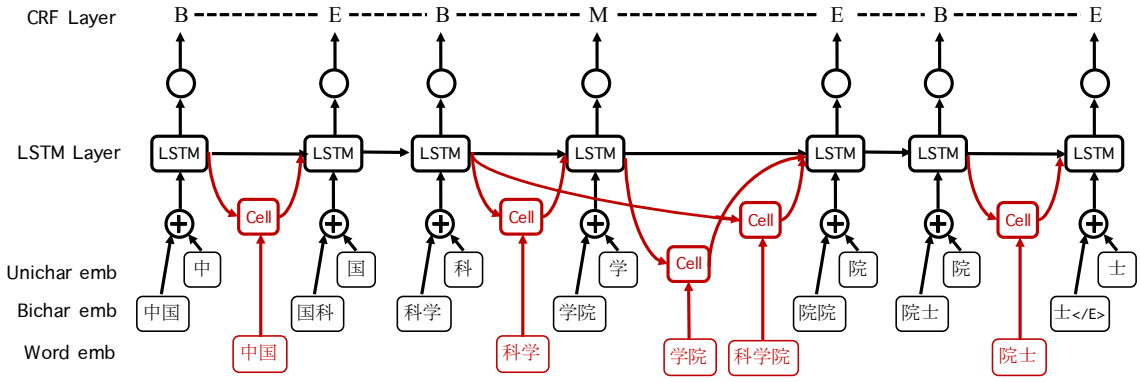

Figure 2: Models. Only forward LSTM is illustrated here.

2015a; Cai and Zhao, 2016) generally take the same framework except using neural networks as automatic feature extractor.

Lattice LSTM was proposed by Zhang and Yang (2018) for Chinese named entity recognition (NER). It integrates the character sequence features and all lexicon word embeddings that match a character subsequence in the input into a sequence labeling model. Zhu et al. (2016) proposed a DAG-structured LSTM structure which is similar to the lattice LSTM model but binarizing the paths in the merging process. Chen et al. (2017) also built a DAG-LSTM structure for word segmentation but without memory cells. Our model consistently gives better performance.

BPE is a data compression algorithm (Gage, 1994) which has been used in neural machine translation (NMT) by capturing the most frequent subwords instead of words (Sennrich et al., 2016). Here we use it for collecting subwords in Chinese, similar to the use in Chinese NMT.

3 Models

We take the state-of-the-art LSTM-CRF framework as our baseline. For an input sentence with m characters $s = c_1, c_2, \ldots, c_m$, where c_i denotes the ith character, the segmentor is to assign each character c_i with a label l_i. Figure 2 shows the segmentor framework on input character sequence "中国科学院院士 (Fellow of the Chinese Academy of Sciences)", where the black part represents the baseline LSTM-CRF model and the red part shows the lattice structure.

3.1 Baseline Model

As shown in Figure 2, for each input character c_i, the corresponding character unigram embeddings and character bigram embeddings are represented as $\mathbf{e}_{c_i}$ and $\mathbf{e}_{c_i c_{i+1}}$, respectively. The character representation is calculated as following:

$$\mathbf{x}_i = \mathbf{e}_{c_i} \oplus \mathbf{e}_{c_i c_{i+1}}, \quad (1)$$

where $\oplus$ represents *concatenate* operation.

Unlike Zhang et al. (2016) which uses a window to strengthen the local features, or Zhou et al. (2017) which adds a non-linear layer before the LSTM layer, we feed $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$ into a bidirectional LSTM:

$$\begin{aligned} \overrightarrow{\mathbf{h}}_1, \overrightarrow{\mathbf{h}}_2, \ldots, \overrightarrow{\mathbf{h}}_m &= \overrightarrow{LSTM}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m) \\ \overleftarrow{\mathbf{h}}_1, \overleftarrow{\mathbf{h}}_2, \ldots, \overleftarrow{\mathbf{h}}_m &= \overleftarrow{LSTM}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m), \end{aligned} \quad (2)$$

where $\overrightarrow{LSTM}$ and $\overleftarrow{LSTM}$ represent the forward and backward LSTM, respectively. The detailed equations are listed in *Appendix*. The hidden vector of character c_i is

$$\mathbf{h}_i = \overrightarrow{\mathbf{h}}_i \oplus \overleftarrow{\mathbf{h}}_i \quad (3)$$

3.2 Lattice LSTM

The lattice LSTM adds "shortcut paths" (red part in Figure 2) to LSTM. The input of the lattice LSTM model is character sequence and all subsequences which are matched words in a lexicon $\mathbb{D}$, collected from BPE. Following Zhang and Yang (2018), we use $w_{b,e}$ to represent the subsequence that has a start character index b and a end character index e, and the embeddings of the subsequence is represented as $\mathbf{e}_{w_{b,e}}$.

During the forward lattice LSTM calculation, the "cell" in Figure 2 of a subsequence $w_{b,e}$ takes the hidden vector of the start character $\mathbf{h}_b$ and the subsequence embeddings $\mathbf{e}_{w_{b,e}}$ as input, an extra

Parameter	Value	Parameter	Value
char emb size	50	bigram emb size	50
word emb size	50	subword emb size	50
char dropout	0.5	lattice dropout	0.5
LSTM layer	1	LSTM hidden	200
learning rate *lr*	0.01	*lr* decay	0.05

Table 1: Hyper-parameter values.

LSTM cell is applied to calculate the memory vector of the sequence $\mathbf{c}_{w_{b,e}}$:

$$\mathbf{c}_{w_{b,e}} = LSTMCell(\mathbf{h}_b, \mathbf{e}_{w_{b,e}}), \quad (4)$$

where the $LSTMCell$ is a simplified LSTM unit which calculate the memory only. The output memory vector $\mathbf{c}_{w_{b,e}}$ links to the end character c_e to calculate its hidden vector $\overrightarrow{\mathbf{h}}_e$. For character with multiple memory cell inputs[2], we assign a gate for each subsequence input to control its contribution. The detailed equations are listed in *Appendix*. The final output $\overrightarrow{\mathbf{h}}_i$ includes both the character sequence history information and all the matched subsequence information.

3.3 Decoding and Training

We use a standard CRF layer for inference (details in *Appendix*). Viterbi (1967) is used to find the highest scored label sequence over the input. During training, we choose sentence-level log-likelihood as the loss function.

$$Loss = \sum_{i=1}^{N} log(P(y_i|s_i)), \quad (5)$$

where y_i is the gold labels of sentence s_i.

4 Experiments

4.1 Experimental Settings

Data. We evaluate our model on four standard Chinese word segmentation datasets: CTB6, PKU, MSR, and Weibo. PKU and MSR are taken from the SIGHAN 2005 bake-off (Emerson, 2005) and Weibo dataset is the NLPCC 2016 shared task (Qiu et al., 2016), standard split are used. We take CTB6 as the main dataset and split the train/dev/test following Zhang et al. (2016). The statistics of the datasets are listed in *Appendix*.

[2]e.g. The first "院" in Figure 2 takes two subsequence memory vectors of both "学院" and "科学院" as input.

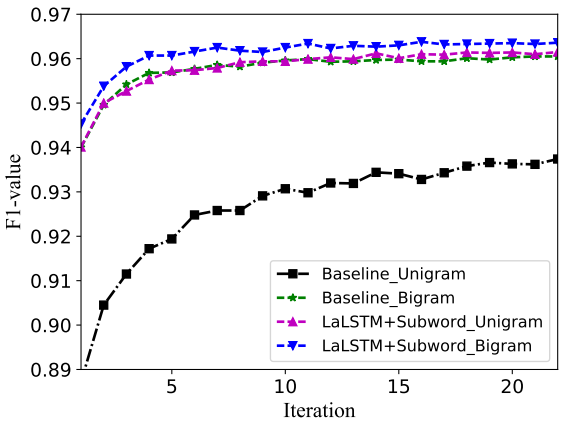

Figure 3: F1-value against training iterations.

Hyperparameters. We keep the hyperparameters the same among all datasets. Standard gradient descent (SGD) with a learning rate decay is used as the optimizer. The embedding sizes of character unigram/bigram and subword are all 50. Dropout (Srivastava et al., 2014) is used on both the character input and the subword input to prevent overfitting. Details are listed in Table 1.

Embeddings. We take the same character unigram and bigram embeddings as Zhang et al. (2016), who pretrain embeddings using word2vec (Mikolov et al., 2013) on Chinese Gigaword[3]. The vocabulary of subword is constructed with 200000 merge operations and the subword embeddings are also trained using word2vec (Heinzerling and Strube, 2018). Trie (Fredkin, 1960) is used to accelerate lattice building. All the embeddings are fine-tuned during training.

4.2 Development Experiments

We perform experiments on the CTB6 development dataset to investigate the contribution of character bigram information and the subword information. Figure 3 shows the iteration curve of F-scores against different numbers of training iterations with different character representations. "_Bigram" represents the model using both character unigram and bigram information (embedding concatenation). Character bigram information can improve the baseline significantly. When the "LaLSTM+Subword" structure is added, the model performance is further improved. This shows that subword information has a great ability to disambiguate the characters.

[3]https://catalog.ldc.upenn.edu/LDC2011T13.

Models	CTB6	SIGHAN		Weibo
		MSR	PKU	
Zheng et al. (2013)	–	93.3	92.4	–
Pei et al. (2014)	–	97.2	95.2	–
Ma and Hinrichs (2015)	–	96.6	95.1	–
Liu et al. (2016)	95.5	97.6	95.7	–
Zhang et al. (2016)	96.0	**97.7**	95.7	–
Xu and Sun (2016)	95.8	96.3	**96.1**	–
Cai et al. (2017)	–	97.1	95.8	–
Chen et al. (2017)	95.6	96.1	–	–
Yang et al. (2017)†	95.4	96.8	95.0	**94.5**
Ma et al. (2018)	**96.7**	97.4	**96.1**	–
Baseline	95.8	97.4	95.3	95.0
LaLSTM+Subword	**96.1**	**97.8**	**95.8**	**95.3**

Table 2: Main results (F1).

Model	P	R	F1	ER%	R_{IV}	R_{OOV}
Baseline	95.93	95.62	95.78	0	96.70	77.36
Random Emb	96.13	95.82	95.97	-4.5	96.85	78.37
Pretrain Emb	96.23	95.90	96.07	-6.9	96.86	79.79

Table 3: Lexicon and embeddings on CTB6.

Zhang and Yang (2018) observed that character bigram information has a negative effect in lattice LSTM on Chinese NER task, while we find a different result on Chinese word segmentation where character bigram information gives significant improvements in the lattice LSTM. This is likely because character bigrams are informative but ambiguous. They can provide more useful character disambiguation evidence in segmentation than in NER where lattice LSTM works well in disambiguating characters.

4.3 Results

Table 2 shows the main results and the recent state-of-the-art neural CWS models. Zhang et al. (2016) integrated both discrete features and neural features in a transition-based framework. Xu and Sun (2016) proposed the dependency-based gated recursive neural network to utilize long distance dependencies. Yang et al. (2017)† utilized pretrained character representations from multitasks. We examine their non-pretrained model performance for fair comparison. Ma et al. (2018) built a bidirectional LSTM model with carefully hyperparameter selection. These methods are orthogonal to and can be integrated into our lattice structure.

As shown in Table 2, the subword lattice LSTM gives significant improvements on all evaluated datasets. In the PKU dataset, our model is slightly behind Xu and Sun (2016) which preprocesses the dataset by replacing all the Chinese idioms, leading the comparison not entirely fair. Our model gives the best performance on MSR and Weibo datasets, which demonstrates that subword encoding can help the lattice LSTM model gives comparable performance to the state-of-the-art word segmentation models.

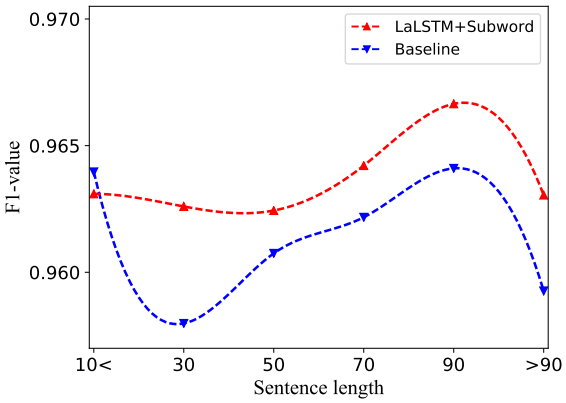

Figure 4: F1-value against the sentence length.

4.4 Analysis

Lexicon and Embeddings. To distinguish the contribution of subword lexicon and their pretrained embeddings, we conduct a set of experiments by using the same subword lexicon with randomly initialized embeddings[4] on CTB6 data. As shown in Table 3, the contribution of the error reduction by the lexicon is 4.5%. While 6.9% error reduction comes from both lexicon and pretrained embeddings. We can estimate that the contribution of pretraining is $(6.9\% - 4.5\%) = 2.4\%$. This roughly shows that both lexicon and pretraining are useful to lattice LSTM, and the former contributes more than the latter.

OOV Analysis. Table 3 also shows the recall of in-vocabulary ($\mathbf{R}_{IV}$) and out-of-vocabulary ($\mathbf{R}_{OOV}$) words, respectively. As shown in the table, the $\mathbf{R}_{OOV}$ can be largely improved with the lattice structure (2.43% absolute improvement).

Sentence Length. We compare the baseline model with our proposed model on the sentence length distribution in Figure 4. The performance of the baseline has a valley in around 30-character length and decreases when the sentence length over 90. This phenomenon has also been observed in transition-based neural segmentor Yang et al. (2017). While "LaLSTM+Subword" gives a more stable performance along sentence length.

[4]Within $[-\sqrt{\frac{3}{dim}}, \sqrt{\frac{3}{dim}}]$, dim is the embedding size.

Data	Split	#Word	#Match	Ratio (%)	ER (%)
CTB6	Train	641k	536k	83.57	–
	Test	81.6k	68.6k	84.13	7.14
MSR	Train	2.12m	1.93m	91.12	–
	Test	107k	98.2k	91.91	15.4
PKU	Train	1.01m	918k	90.87	–
	Test	104k	95.4k	91.42	10.6
Weibo	Train	421k	337k	80.10	–
	Test	188k	147k	78.39	6.0

Table 4: Subword coverage.

Sentence		国际生物多样性日纪念大会在京举行 Int'l Biological Diversity Day COMM meeting in Beijing hold
Gold Segmentation		国际/ 生物/ 多样性/ 日/ 纪念/ 大会/ 在/ 京/ 举行 Int'l/Biological/Diversity/Day/COMM/meeting/in/Beijing/hold
Baseline		国际/ 生物/ **多样性日**/ 纪念/ 大会/ 在/ 京/ 举行 Int'l/Biological/ **DiversityDay**/ COMM/meeting/in/Beijing/hold
LaLSTM +Subword	**Matched**	国际,生物多样性,多样性,纪念,大会,在京,举行 Int'l,BiologicalDiversity,Diversity,COMM,meeting,inBeijing,hold
	Decode	国际/ 生物/ 多样性/ 日/ 纪念/ 大会/ 在/ 京/ 举行 Int'l/Biological/Diversity/Day/COMM/meeting/in/Beijing/hold

Figure 5: Example.

Subword Coverage in lexicon. Table 4[5] shows the subword coverage rate in four datasets. Subword level coverage is consistently higher than the entity level coverage in Zhang and Yang (2018). We can see that higher subword coverage (PKU/MSR, $> 90\%$) gives better error reduction rate. Weibo dataset gets the minimum improvement due to the low subword coverage.

Case Study. Figure 5 shows an example of CTB6 test dataset. In this example, there are two matched subwords "生物多样性(BiologicalDiversity)" and "多样性(Diversity)" which can guide the segmentor to get the right split of "多样性日(DiversityDay)", which is segmented incorrectly by the baseline.

5 Conclusion

We examined the effectiveness of subwords for neural CWS. Subwords are deduced using BPE, and then integrated into a character-based neural segmentor through lattice LSTM. Results on four benchmarks show that subword brings significant improvements over a character baseline, and our proposed model gives comparable performances to the best systems on all datasets. Our experiments also showed that the matched subwords contribute more than embedding pertaining, which indicates that the lattice LSTM structure with domain lexicons can be useful for cross-domain segmentation training.

[5] **#Word** is the word number in the corresponding dataset, **#Match** is the matched words number between the dataset and subword lexicon, **#Ratio** $= \frac{\#\text{Match}}{\#\text{Word}}$ represents the subword coverage rate. **#ER** is the error reduction compared with baseline model.

Acknowledgments

We thank the anonymous reviewers for their insightful comments. Yue Zhang is the corresponding author.

References

Deng Cai and Hai Zhao. 2016. Neural word segmentation learning for chinese. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 409–420.

Deng Cai, Hai Zhao, Zhisong Zhang, Yuan Xin, Yongjian Wu, and Feiyue Huang. 2017. Fast and accurate neural word segmentation for chinese. In *Proceedings of ACL*, pages 608–615.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, and Xuanjing Huang. 2015a. Gated recursive neural network for chinese word segmentation. In *Proceedings of the 53th Annual Meeting of the Association for Computational Linguistics*.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015b. Long short-term memory neural networks for chinese word segmentation. In *Proceedings of EMNLP*, pages 1385–1394.

Xinchi Chen, Zhan Shi, Xipeng Qiu, and Xuanjing Huang. 2017. Dag-based long short-term memory for neural word segmentation. *arXiv preprint arXiv:1707.00248*.

Thomas Emerson. 2005. The second international chinese word segmentation bakeoff. In *Proceedings of the fourth SIGHAN workshop on Chinese language Processing*, volume 133.

Edward Fredkin. 1960. Trie memory. *Communications of the ACM*, 3(9):490–499.

Philip Gage. 1994. A new algorithm for data compression. *The C Users Journal*, 12(2):23–38.

Benjamin Heinzerling and Michael Strube. 2018. BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation*, Miyazaki, Japan.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Yijia Liu, Wanxiang Che, Jiang Guo, Bing Qin, and Ting Liu. 2016. Exploring segment representations for neural segmentation models. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*.

Ji Ma, Kuzman Ganchev, and David Weiss. 2018. State-of-the-art chinese word segmentation with bi-lstms. In *Proceedings of EMNLP*, pages 4902–4908, Brussels, Belgium. Association for Computational Linguistics.

Jianqiang Ma and Erhard Hinrichs. 2015. Accurate linear-time chinese word segmentation via embedding matching. In *Proceedings of ACL-IJCNLP*, pages 1733–1743, Beijing, China.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Wenzhe Pei, Tao Ge, and Baobao Chang. 2014. Max-margin tensor neural network for chinese word segmentation. In *Proceedings of the 52th Annual Meeting of the Association for Computational Linguistics*, pages 293–303. Association for Computational Linguistics.

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. In *Proceedings of the International Conference on Computational Linguistics*, page 562.

Xipeng Qiu, Peng Qian, and Zhan Shi. 2016. Overview of the nlpcc-iccpol 2016 shared task: Chinese word segmentation for micro-blog texts. In *International Conference on Computer Processing of Oriental Languages*, pages 901–906. Springer.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of ACL*, pages 1715–1725, Berlin, Germany.

Richard Sproat, William Gale, Chilin Shih, and Nancy Chang. 1996. A stochastic finite-state word-segmentation algorithm for chinese. *Computational linguistics*, 22(3):377–404.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Weiwei Sun. 2010. Word-based and character-based word segmentation models: Comparison and combination. In *Proceedings of the International Conference on Computational Linguistics*, pages 1211–1219.

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

Jingjing Xu and Xu Sun. 2016. Dependency-based gated recursive neural network for chinese word segmentation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, page 567. Association for Computational Linguistics.

Nianwen Xue et al. 2003. Chinese word segmentation as character tagging. *Computational Linguistics and Chinese Language Processing*, 8(1):29–48.

Jie Yang and Yue Zhang. 2018. NCRF++: An open-source neural sequence labeling toolkit. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Jie Yang, Yue Zhang, and Fei Dong. 2017. Neural word segmentation with rich pretraining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 839–849, Vancouver, Canada.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2016. Transition-based neural word segmentation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Ruiqiang Zhang, Genichiro Kikui, and Eiichiro Sumita. 2006. Subword-based tagging by conditional random fields for chinese word segmentation. In *Proceedings of HLT-NAACL*, pages 193–196, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2007. Chinese segmentation with a word-based perceptron algorithm. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, volume 45, page 840.

Yue Zhang and Jie Yang. 2018. Chinese ner using lattice lstm. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, Melbourne, Australia.

Hai Zhao, Chang-Ning Huang, Mu Li, and Bao-Liang Lu. 2006. Effective tag set selection in chinese word segmentation via conditional random field modeling. In *Proceedings of PACLIC*, volume 20, pages 87–94. Citeseer.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for chinese word segmentation and pos tagging. In *Proceedings of EMNLP*, pages 647–657.

Hao Zhou, Zhenting Yu, Yue Zhang, Shujian Huang, XIN-YU DAI, and Jiajun Chen. 2017. Word-context character embeddings for chinese word segmentation. In *Proceedings of EMNLP*, pages 760–766, Copenhagen, Denmark.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. 2016. Dag-structured long short-term memory for semantic compositionality. In *Proceedings of NAACL-HLT*, pages 917–926.