Feature-based Neural Language Model and Chinese Word Segmentation

Mairgup Mansur, Wenzhe Pei, Baobao Chang
Key Laboratory of Computational Linguistics, Ministry of Education
Institute of Computational Linguistics, Peking University
mairgup@gmail.com, williampei1988@126.com, chbb@pku.edu.cn

Abstract

In this paper we introduce a feature-based neural language model, which is trained to estimate the probability of an element given its previous context features. In this way our feature-based language model can learn representation for more sophisticated features. We introduced the deep neural architecture into the Chinese Word Segmentation task. We got a significant improvement on segmenting performance by sharing the pre-learned representation of character features. The experimental result shows that, while using the same feature sets, our neural segmentation model has a better segmenting performance than CRF-based segmentation model.

1 Introduction

Nowadays, as a distributed representation learning framework, Deep Learning Architecture has received a lot of attention in the NLP literature.

As Hinton introduced the idea of distributed representation for symbolic data in (Hinton, 1986), this idea has been a research hot spot for more than twenty years.

Bengio (2003) applied Hinton's idea in the context of language modeling, and developed a neural language model. Mikolov (2011) improved Bengio's neural language model by adding recurrence to the hidden layers, allowing it to beat the state-of-the-art n-gram model not only in terms of perplexity but also in terms of word error rate in speech recognition. Schwenk (2012) applied similar models in statistical machine translation and improved the BLEU score by almost 2 points.

Different from the traditional n-gram model, a distributed representation of word can be learned by neural language model from unlabeled raw data. As the most important philosophy of Deep Learning architecture, the property of learning distributed word representation of neural language model have been proved very useful in NLP tasks.

Colobert et al. (2011) developed the SENNA system that shares word representations across the tasks of language modeling, part-of-speech tagging, chunking, named entity recognition, semantic role labeling and syntactic parsing, which approaches or surpasses the state-of-the-art on these tasks. More interestingly, their experimental result shows that, sharing the word representations that are learned by the neural language model on massive raw data can significantly improve the overall performance of the other tasks.

All these research have showed us the very good capability of the neural language model in learning word representation. However, for many NLP tasks, not only the distributed representation of word, but also the representation of features is important. Since the neural language model can learn the representation of words, can we use it to learn representation of features?

In this paper, we introduce a more generalized feature-based neural language model, which can learn distributed representation of features, which is very useful for many NLP tasks.

Traditional neural language model aims to estimate the probability of a word given words in its previous context, our feature-based neural language model is a generalization of the traditional neural language model, which views the language modeling problem as feature-based prediction problem. The features used to predict an element can be words or characters as well as other sophisticated features extracted from the previous context. After training, our feature-based neural language model can learn a distributed representation of those sophisticated features.

To test the usefulness of the feature representation learned by our feature-based neural language model, we introduce a deep neural archi-

International Joint Conference on Natural Language Processing, pages 1271–1277,
Nagoya, Japan, 14-18 October 2013.

tecture into the Chinese Word Segmentation task. We conducted a series of experiments on the SIGHAN-2005 datasets, and got a positive result.

By sharing the feature representation learned by our feature-based neural language model with our neural segmentation model, we got a significant improvement on the segmentation performance.

We also made comparison between our neural segmentation model and the classical segmentation method based on Conditional Random Fields(CRF). The experimental result shows that, while using the same feature sets, the neural segmentation model have a better performance than that of CRF-based method, especially when sharing the pre-learned feature representations.

The rest of the paper is organized as follows. Section 2 overviews the task of Chinese Word Segmentation. Section 3 introduce our feature-based neural language model. The deep neural architecture used for segmentation is described in Section 4. Section 5 gives the experimental result. The last section concludes this paper.

2 Chinese Word Segmentation

Unlike English and other western languages, Chinese do not delimit words by white-space. Therefore word segmentation is a very basic and important pre-process for Chinese language processing.

Traditional word segmentation approaches are lexicon-driven (Liang, 1987). Lexicon-driven methods assume that predefined Chinese word lexicon is available, hence the segmentation performance strongly depends on the predefined lexicon.

Xue (2003) proposed a novel way of segmenting Chinese texts, and views the Chinese word segmentation task as a character tagging task. According to Xue's approach, a tagging model is learned from manually segmented training texts, and then used to assign each character a tag indicating the position of this character within a word it belongs to. Xue's approach, which did not require any predefined lexicon and have a high performance, became the most popular appoach to Chinese word segmentation in recent years.

In Sighan Bakeoff-2005, two participants (Low, 2005) and (Tseng, 2005) have given the best results in almost all word segmentation tracks. Both of their systems use sequence tagging methods based on Conditional Random Field (CRF).

CRF is a statistical sequence modeling framework first introduced into natural language processing in (Lafferty et al., 2001). Peng et al. (2004) first used this framework for Chinese word segmentation by treating it as a binary decision task, such that each character is labeled either as the beginning of a word or the continuation of one.

In this paper, we use a CRF-based segmentation system to do a series of comparative experiments.

As in most other work did on segmentation, we use a 4-tag schema, such that each Chinese character is labeled by a tag in the tag set "B,M,E,S". In which, "B", "E" and "M" stands for the character in the beginning, ending or middle of a word, "S" means that the character is a word by itself.

We use the following feature templates, which are widely used in most segmentation work:

(a) $C_n(n = -2, -1, 0, 1, 2)$

(b) $C_nC_{n+1}(n = -2, -1, 0, 1)$

(c) $C_{-1}C_1$

Here C refers to a character; n refers to the position index relative to the current character. By setting the above feature templates, we actually set a 5-character window to extract features, the current character, 2 characters to its left and 2 to its right.

Other work on segmentation used much more sophisticated feature templates other than the one introduced above. However, defeating the state-of-the-art segmentation work is not the main purpose of this paper. Here, we just want to test whether our model can use the same feature set more efficiently than the CRF-based model. Since it is practicable but not necessary to use other sophisticated feature templates to do the model comparison, we just use the aforementioned feature template as the standard feature set for the Chinese word segmentation task.

3 Feature-based Neural Language Model

In this section, we first overview the widely used traditional neural language model.Then we introduce our feature-based neural language model, which is a generalization of the traditional one and can learn representation of features from unlabeled raw data through unsupervised training.

3.1 Traditional Neural Language Model

Language models are widely used in many NLP applications to compute "scores" describing how likely a piece of text is. The classical n-gram language model is a direct application of Markov

models, estimating the probability of a word given its previous words in a sentence.

Neural language models were proposed by Bengio and Ducharame (2003) and Schwenk and Gauvain (2004). These neural language models were designed to estimate the conditional probability of a word given its previous context in a sentence.

Different from the traditional n-gram language model, neural language model can learn a distributed representation of words from unlabeled raw data. As an application of the deep learning architecture, what we value most is the representation learning ability of the neural language model.

For many NLP tasks, the distributed representation of features is important as well as the one of words. However, the traditional neural language model have this shortcoming that it is designed to learn representation of words only.

To overcome this shortcoming of the traditional neural language model, we propose a much more generalized neural language model that can learn distributed representation of features. To distinguish from the traditional one, we call it the feature-based neural language model.

3.2 Feature-based Neural Language Model

Unlike the traditional neural language model, which aims to estimate the probability of a word given its previous context words, our feature-based neural language model views the language modeling problem as feature-based prediction problem. Our feature-based neural language model estimate the probability of an element given its history context feature set. The predicted element could be a word or a character in a sentence.

The architecture of our feature-based neural language model is summarized in Figure 1.

More formally, the probability estimated by our feature-based neural language model is:

$$p(E|F_{history}, \theta)$$

where θ is the set of parameters of our model. We denote the feature set extracted from the history context of an element E as $F_{history}$:

$$F_{history} = \{f_1, f_2, \ldots, f_k\}$$

The first layer of our feature-based neural language model is called lookup table layer, which maps each $f \in F_{history}$ into a d_f-dimensional feature vector $W_{\cdot f} \in \mathbb{R}^{d_f}$:

$$LT_W(f) = W_{\cdot f}$$

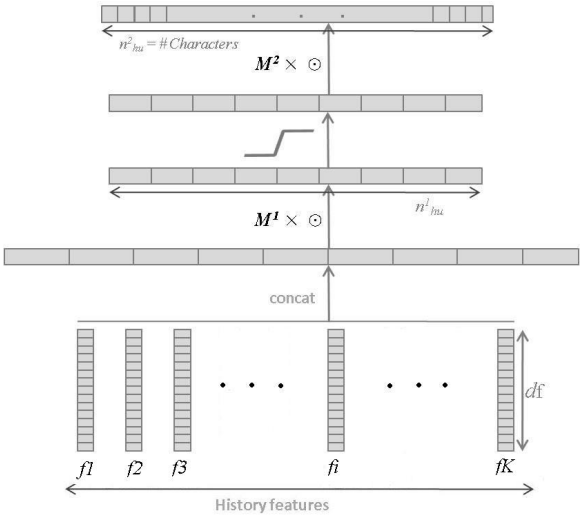

Figure 1: Feature-based Neural Language Model

where W is called the lookup table, which contains the distributed representation of features, and it is the most important parameter of our feature-based neural language model.

Given a set of features $F_{history} = [f]_1^k$, the lookup table layer applies the same operation for each feature in $F_{history}$, producing the following $d_f \times k$ matrix:

$$LT_W([f]_1^k) = (W_{\cdot[f]_1} \; W_{\cdot[f]_2} \; \ldots \; W_{\cdot[f]_k})$$

This matrix can be viewed as a $d_f k$-dimensional vector z^1 by concatenating its column vectors. Then z^1 can be fed to further neural network layers which perform affine transformations as below:

$$z^l = M^l z^{l-1} + b^l$$

where $M^l \in \mathbb{R}^{n^l_{hu} \times n^{l-1}_{hu}}$ and $b^l \in \mathbb{R}^{n^l_{hu}}$ are the parameters to be trained. The hyper-parameter n^l_{hu} indicates the number of hidden units of the l^{th} layer. If the l^{th} layer is not the last layer, to extract highly non-linear features, a non-linearity function must follow. We use $Tanh()$ to be our non-linearity function.

$$Tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$$

Finally, the output size of the last layer L of the network is equal to the number of possible elements. Each output can be then interpreted as the corresponding probability of an element given the history context feature set.

3.3 Model Training

All the parameters of our feature-based neural language model are trained by maximizing a likelihood over the unlabeled training data, using stochastic gradient ascent method (Bottou, 1991).

If we denote θ to be all the trainable parameters of the network, which are trained using a training set $\mathcal{T}$ we want to maximize:

$$\theta \mapsto \sum_{(f_h,e)\in\mathcal{T}} \log p(e \mid f_h, \theta) \qquad (1)$$

where e is an element and f_h is the corresponding history context feature set of e.

We maximize equation (1) with stochastic gradient method, by iteratively selecting a random example (f_h,e) and making a gradient step:

$$\theta \leftarrow \theta + \lambda \frac{\partial \log p(e \mid \mathbf{f_h}, \theta)}{\partial \theta}$$

where λ is a chosen learning rate, which is also one of the hyper-parameters of our model.

Since our feature-based neural language model is trained on the unlabeled raw data, so its training process is an unsupervised learning procedure.

4 Deep Neural Architecture For Chinese Word Segmentation

In this section, we introduce the deep neural architecture into the Chinese word segmentation task.

4.1 Neural Network Segmentation Model

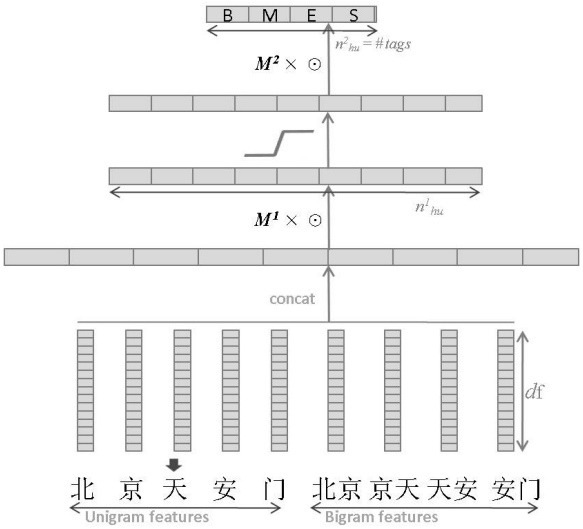

Figure 2: Neural Network Architecture for CWS

The architecture of our neural segmentation model is summarized in Figure 2. We can see that this architecture is almost the same as our feature-based neural language model, only have slightly difference at the output layer.

As input, character features (unigram or bigram features) are fed as indices taken from a finite dictionary $\mathcal{D}$. This dictionary $\mathcal{D}$ contains all the character features which appeared in the training data.

The first layer of the network maps each of these character feature $c_f \in \mathcal{D}$ into a d_f-dimensional feature vector $W_{.c} \in \mathbb{R}^{d_f}$, where W contains distributed representations of the character features. Given a set of character features $[c_f]_1^T$ in $\mathcal{D}$, the lookup table layer applies the same operation for each character feature in the sequence, producing a $d_f \times T$ matrix.

Given a character to tag, we consider a fixed-size k_{sz} window of characters around this character to collect the character features. If we only use the unigram features, then there will be k_{sz} ($T = k_{sz}$) character features that are first passed through the lookup table layer, producing a matrix of fixed size $d_f \times k_{sz}$. If we use both the unigram and bigram features, then there will be $2k_{sz} - 1$ ($T = k_{sz} + k_{sz} - 1$) charater features that will produce a matrix in shape of $d_f \times (2k_{sz} - 1)$.

And this matrix can be viewed as a $d_f T$-dimensional vector by concatenating its column vectors. Then this concatenated vector z^1 can be fed to further neural network layers which perform affine transformations over their inputs.

To extract highly non-linear features at hidden layers, we also use $Tanh()$ to be our non-linearity function here.

Finally, the output size of the last layer L of the segmentation model is equal to the number of possible tags, which is 4 (B,M,E,S) in our case. Each output can be then interpreted as probability of tagging the current Chinese character with the corresponding tag.

4.2 Model training

All the parameters of our neural segmentation model are trained by maximizing a likelihood over the training data, using stochastic gradient ascent, like we did in previous section.

Since the lookup table of feature vectors is the most important parameter in this neural segmentation model, sharing a well trained lookup table must have a positive impact on the segmentation performance.

We will use our feature-based neural language

model to pre-train a lookup table on a larger raw text data, and then use this pre-trained lookup table to initialize the corresponding lookup table parameter of the neural segmentation model.

Since the pre-training process is unsupervised, so the whole training process of our segmentation model is actually a semi-supervised procedure.

5 Experiments

5.1 Experimental Setup

To make a comprehensive comparison between our neural segmentation model and the CRF-based model, we conduct a closed test on the PKU dataset from the Sighan-2005 Chinese word segmentation bake-off competition.

We remove the the white-spaces between words in the training and testing data sets of PKU and MSRA datasets and make it all a raw text training data for our feature-based neural language model. Some statistical information of the data sets are given in Table 1.

	PKU	MSRA
Word Types	55,303	88,119
Words	1,109,947	2,368,391
Character Types	4,698	5,167
Characters	1,826,448	4,050,469

Table 1: Statistics of Sighan-2005 data sets.

For evaluation, we use the standard bake-off scoring program to calculate precision, recall, F1, OOV and IV word recall.

To make a performance comparison between our neural segmentation model and the CRF-based model under the same feature sets, we did experiments under three different scenarios.

In each scenario, both our neural segmentation model and CRF-based model are trained using the same feature sets described in Table 2.

As shown in Table 2, in scenario 1, both two models use unigram character features only, in scenario 2 the unigram and bigram character features are used, and in scenario 3 the standard feature set for the CRF-based model.

In each scenario, a feature-based neural language model is trained to learn the distributed representation (lookup table) of the features that is used by the neural segmentation model.

For each experimental setup, we provide the results of our neural segmentation model with or without sharing the look up table parameters learned by our feature-based language model.

	Feature Sets
Scenario 1	C_{-2},C_{-1},C_0,C_1,C_2
Scenario 2	C_{-2},C_{-1},C_0,C_1,C_2 $C_{-2}C_{-1}$,$C_{-1}C_0$,C_0C_1,C_1C_2
Scenario 3	C_{-2},C_{-1},C_0,C_1,C_2 $C_{-2}C_{-1}$,$C_{-1}C_0$,C_0C_1,C_1C_2 $C_{-1}C_1$

Table 2: Feature Sets used in 3 scenarios.

In this paper, we use an open source toolkit "CRF++"[1] to train the CRF models. While training the CRF models, we set the parameter cutoff threshold for the features to be 3 and other parameters are set by default. The hyper-parameters of our feature-based neural language model and neural segmentation model are set as in Table 3.

Window size	$k_{sz} = 5$
Feature dim	$d_f = 30$
Hidden units	$n_{hu} = 300$
learning rate	$\lambda = 0.001$

Table 3: Hyper-Parameters of our neural segmentation model and feature-based neural language model.

5.2 Scenario 1:Only Unigram Features

In Table 4 are given the results of experiments on scenario 1, in which only unigram character features are used by both segmentation models.

As the result shows, while using the same feature set, our neural segmentation model have a better performance than that of the CRF-based model, which means our model makes use of the features more efficiently than the CRF-based model.

From the result we can see that sharing the lookup table parameters with the pre-trained feature-based neural language model can significantly improve the performance of our neural segmentation model. The OOV recall is boosted from 48.9% to 68.8%, this means pre-trained lookup table can help neural segmentation model to deal with OOV problem more smoothly.

5.3 Scenario 2:Unigram & Bigram Features

The results in Table 5 shows the performance of our neural segmentation model and the CRF-

[1] crfpp.sourceforge.net

	P	R	F1	R_{OOV}	R_{IV}
CRF	0.846	0.863	0.855	0.533	0.866
NN	0.871	0.879	0.875	0.489	0.895
NN+LM	0.912	0.927	0.920	0.688	0.926

Table 4: Results of our Neural Segmentation Model (NN) and CRF-based Segmentation Model, using only unigram features, "+LM" means sharing pre-trained lookup table parameters.

based model when unigram and bigram features are used. The experimental result again shows that the performance of our neural segmentation model surpasses that of the CRF-based model, while using the same feature set.

We can see that using our feature-based neural language model to pre-train the lookup table parameters can significantly boost the system performance, which means our feature-based neural language model can learn a better lookup table that contains task-useful information.

	P	R	F1	R_{OOV}	R_{IV}
CRF	0.918	0.934	0.926	0.566	0.940
NN	0.927	0.933	0.930	0.566	0.949
NN+LM	0.932	0.940	0.936	0.628	0.950

Table 5: Results of our Neural Segmentation Model (NN) and CRF-based Segmentation Model, using unigram and bigram features, "+LM" means sharing pre-trained lookup table parameters.

5.4 Scenario 3:All Standard Features

The results in Table 6 shows the performance of our neural segmentation model and the CRF-based model when the standard feature set are used.

The $+LM_1$ in Table 6 means that neural segmentation model only share unigram feature representations, which are learned by our feature-based language model in scenario 1. The $+LM_2$ means neural segmentation model share the representations of the standard feature set learned by our feature-based language model.

We can see that our neural segmentation model have almost the same but still better performance than that of CRF-based model. The result also shows that, only sharing pre-learned unigram feature representation is not helpful to boost the performance of our neural segmentation model. When sharing the pre-learned representations of the same features used by neural segmentation model, a significant improvement on the system performance can be achieved.

From Table 4, 5 and 6 we can see that, sharing pre-learned feature representations can always boost the performance of our neural segmentation model. This means that our feature-based neural language model is effective in learning feature representations, even if the features are more sophisticated. The performance boost mainly come from the improvement on OOV recall, means that the pre-learned feature representations is helpful while dealing with the OOV problem in Chinese Word Segmentation task.

	P	R	F1	R_{OOV}	R_{IV}
CRF	0.924	0.939	0.931	0.609	0.943
NN	0.936	0.928	0.932	0.579	0.958
NN+LM_1	0.935	0.929	0.932	0.569	0.958
NN+LM_2	0.940	0.939	0.940	0.695	0.955

Table 6: Results of our Neural Segmentation Model (NN) and CRF-based Segmentation Model using standard features, "+LM" means sharing pre-trained lookup table parameters.

6 Conclusion

In this paper we introduced a generalized feature-based neural language model, which is trained to estimate the probability of an element given its previous context features, thus making it possible to learn distributed representation of more sophisticated features, which is useful for NLP tasks.

To test the efficiency of the feature representation learned by our feature-based neural language model, we introduced the deep neural architecture into the Chinese Word Segmentation task. We conducted a series of experiments on the Sighan-2005 PKU-dataset.

Experimental result shows that our neural segmentation model have a better performance than that of CRF-based model, while these two models are using the same feature sets.

By sharing the representation learned by our feature-based neural language model with the neural segmentation model, we got a significant improvement on system performance. This proved that useful feature representation can be learned by our feature-based neural language model.

Acknowledgments

This work is supported by National Natural Science Foundation of China under Grant No. 61273318.

References

Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

Léon Bottou. 1991. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes*, 91:8.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Geoffrey E Hinton. 1986. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, pages 1–12. Amherst, MA.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *In Proceedings of the 18th International Conf. on Machine Learning*, pages 282–289.

Nanyuan Liang. 1987. "written chinese word segmentation system–cdws". *Journal of Chinese Information Processing*, 21:9–19.(in Chinese).

Jin Kiat Low. 2005. A maximum entropy approach to chinese word segmentation. *Proceedings of the 4th SIGHAN Workshop on Chinese Language Processing, Jeju Island, Korea*, pages 161–164.

Tomas Mikolov, Stefan Kombrink, Lukas Burget, JH Cernocky, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5528–5531. IEEE.

Fuchun Peng, Fangfang Feng, and McCallum Andrew. 2004. Chinese segmentation and new word detection using conditional random fields. *In Proceedings of the 20th International Conf. on Computational Linguistics*, pages 562–568.

Holger Schwenk and Jean-Luc Gauvain. 2004. Neural network language models for conversational speech recognition. In *ICSLP 2004*.

Holger Schwenk, Anthony Rousseau, and Mohammed Attik. 2012. Large, pruned or continuous space language models on a gpu for statistical machine translation. *NAACL-HLT 2012*, page 11.

Huihsin Tseng. 2005. A conditional random field word segmenter for sighan 2005. *Proceedings of the 4th SIGHAN Workshop on Chinese Language Processing, Jeju Island, Korea*, pages 168–171.

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