Gated Recursive Neural Network for Chinese Word Segmentation

Xinchi Chen, Xipeng Qiu*, Chenxi Zhu, Xuanjing Huang
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, China
{xinchichen13,xpqiu,czhu13,xjhuang}@fudan.edu.cn

Abstract

Recently, neural network models for natural language processing tasks have been increasingly focused on for their ability of alleviating the burden of manual feature engineering. However, the previous neural models cannot extract the complicated feature compositions as the traditional methods with discrete features. In this paper, we propose a gated recursive neural network (GRNN) for Chinese word segmentation, which contains reset and update gates to incorporate the complicated combinations of the context characters. Since GRNN is relative deep, we also use a supervised layer-wise training method to avoid the problem of gradient diffusion. Experiments on the benchmark datasets show that our model outperforms the previous neural network models as well as the state-of-the-art methods.

1 Introduction

Unlike English and other western languages, Chinese do not delimit words by white-space. Therefore, word segmentation is a preliminary and important pre-process for Chinese language processing. Most previous systems address this problem by treating this task as a sequence labeling problem and have achieved great success. Due to the nature of supervised learning, the performance of these models is greatly affected by the design of features. These features are explicitly represented by the different combinations of context characters, which are based on linguistic intuition and statistical information. However, the number of features could be so large that the result models are too large to use in practice and prone to overfit on training corpus.

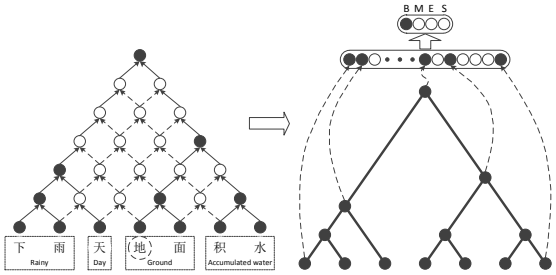

Figure 1: Illustration of our model for Chinese word segmentation. The solid nodes indicate the active neurons, while the hollow ones indicate the suppressed neurons. Specifically, the links denote the information flow, where the solid edges denote the acceptation of the combinations while the dashed edges means rejection of that. As shown in the right figure, we receive a score vector for tagging target character "地" by incorporating all the combination information.

Recently, neural network models have been increasingly focused on for their ability to minimize the effort in feature engineering. Collobert et al. (2011) developed a general neural network architecture for sequence labeling tasks. Following this work, many methods (Zheng et al., 2013; Pei et al., 2014; Qi et al., 2014) applied the neural network to Chinese word segmentation and achieved a performance that approaches the state-of-the-art methods.

However, these neural models just concatenate the embeddings of the context characters, and feed them into neural network. Since the concatenation operation is relatively simple, it is difficult to model the complicated features as the traditional discrete feature based models. Although the complicated interactions of inputs can be modeled by the deep neural network, the previous neural model shows that the deep model cannot outperform the one with a single non-linear model. Therefore, the

*Corresponding author.

Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pages 1744–1753,
Beijing, China, July 26-31, 2015.

neural model only captures the interactions by the simple transition matrix and the single non-linear transformation . These dense features extracted via these simple interactions are not nearly as good as the substantial discrete features in the traditional methods.

In this paper, we propose a gated recursive neural network (GRNN) to model the complicated combinations of characters, and apply it to Chinese word segmentation task. Inspired by the success of gated recurrent neural network (Chung et al., 2014), we introduce two kinds of gates to control the combinations in recursive structure. We also use the layer-wise training method to avoid the problem of gradient diffusion, and the dropout strategy to avoid the overfitting problem.

Figure 1 gives an illustration of how our approach models the complicated combinations of the context characters. Given a sentence "雨 (Rainy) 天 (Day) 地面 (Ground) 积水 (Accumulated water)", the target character is "地". This sentence is very complicated because each consecutive two characters can be combined as a word. To predict the label of the target character "地" under the given context, GRNN detects the combinations recursively from the bottom layer to the top. Then, we receive a score vector of tags by incorporating all the combination information in network.

The contributions of this paper can be summarized as follows:

- We propose a novel GRNN architecture to model the complicated combinations of the context characters. GRNN can select and preserve the useful combinations via reset and update gates. These combinations play a similar role in the feature engineering of the traditional methods with discrete features.
- We evaluate the performance of Chinese word segmentation on PKU, MSRA and CTB6 benchmark datasets which are commonly used for evaluation of Chinese word segmentation. Experiment results show that our model outperforms other neural network models, and achieves state-of-the-art performance.

2 Neural Model for Chinese Word Segmentation

Chinese word segmentation task is usually regarded as a character-based sequence labeling problem. Each character is labeled as one of {B, M, E, S} to indicate the segmentation. {B, M, E} represent *Begin*, *Middle*, *End* of a multi-character segmentation respectively, and S represents a *Single* character segmentation.

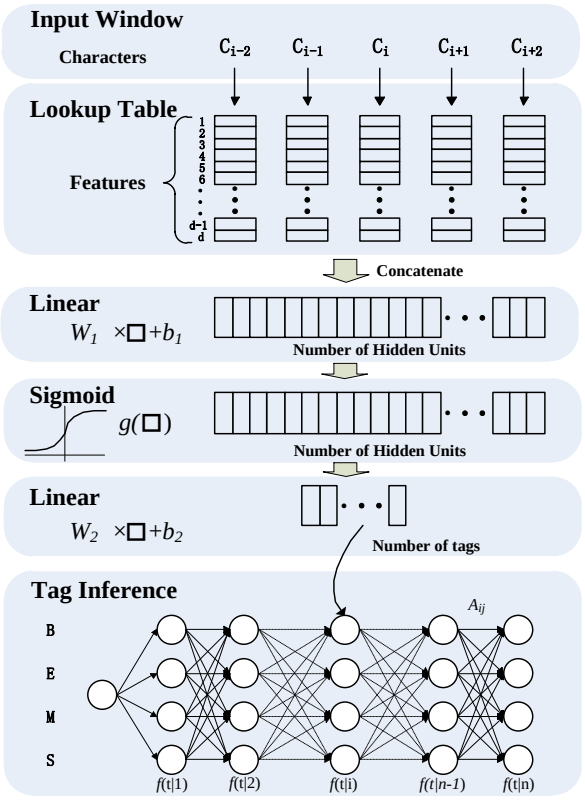

Figure 2: General architecture of neural model for Chinese word segmentation.

The general neural network architecture for Chinese word segmentation task is usually characterized by three specialized layers: (1) a character embedding layer; (2) a series of classical neural network layers and (3) tag inference layer. A illustration is shown in Figure 2.

The most common tagging approach is based on a local window. The window approach assumes that the tag of a character largely depends on its neighboring characters.

Firstly, we have a character set $\mathcal{C}$ of size $|\mathcal{C}|$. Then each character $c \in \mathcal{C}$ is mapped into an d-dimensional embedding space as $\mathbf{c} \in \mathbb{R}^d$ by a lookup table $\mathbf{M} \in \mathbb{R}^{d \times |\mathcal{C}|}$.

For each character c_i in a given sentence $c_{1:n}$, the context characters $c_{i-w_1:i+w2}$ are mapped to their corresponding character embeddings as $\mathbf{c}_{i-w_1:i+w_2}$, where w_1 and w_2 are left and right context lengths respectively. Specifically, the unknown characters and characters exceeding the

sentence boundaries are mapped to special symbols, "unknown", "start" and "end" respectively. In addition, w_1 and w_2 satisfy the constraint $w_1 + w_2 + 1 = w$, where w is the window size of the model. As an illustration in Figure 2, w_1, w_2 and w are set to 2, 2 and 5 respectively.

The embeddings of all the context characters are then concatenated into a single vector $\mathbf{a}_i \in \mathbb{R}^{H_1}$ as input of the neural network, where $H_1 = w \times d$ is the size of Layer 1. And $\mathbf{a}_i$ is then fed into a conventional neural network layer which performs a linear transformation followed by an element-wise activation function g, such as tanh.

$$\mathbf{h}_i = g(\mathbf{W}_1 \mathbf{a}_i + \mathbf{b}_1), \tag{1}$$

where $\mathbf{W}_1 \in \mathbb{R}^{H_2 \times H_1}$, $\mathbf{b}_1 \in \mathbb{R}^{H_2}$, $\mathbf{h}_i \in \mathbb{R}^{H_2}$. H_2 is the number of hidden units in Layer 2. Here, w, H_1 and H_2 are hyper-parameters chosen on development set.

Then, a similar linear transformation is performed without non-linear function followed:

$$f(t|c_{i-w_1:i+w_2}) = \mathbf{W}_2 \mathbf{h}_i + \mathbf{b}_2, \tag{2}$$

where $\mathbf{W}_2 \in \mathbb{R}^{|T| \times H_2}$, $\mathbf{b}_2 \in \mathbb{R}^{|T|}$ and T is the set of 4 possible tags. Each dimension of vector $f(t|c_{i-w_1:i+w_2}) \in \mathbb{R}^{|T|}$ is the score of the corresponding tag.

To model the tag dependency, a transition score $\mathbf{A}_{ij}$ is introduced to measure the probability of jumping from tag $i \in T$ to tag $j \in T$ (Collobert et al., 2011).

3 Gated Recursive Neural Network for Chinese Word Segmentation

To model the complicated feature combinations, we propose a novel gated recursive neural network (GRNN) architecture for Chinese word segmentation task (see Figure 3).

3.1 Recursive Neural Network

A recursive neural network (RNN) is a kind of deep neural network created by applying the same set of weights recursively over a given structure(such as parsing tree) in topological order (Pollack, 1990; Socher et al., 2013a).

In the simplest case, children nodes are combined into their parent node using a weight matrix $\mathbf{W}$ that is shared across the whole network, followed by a non-linear function $g(\cdot)$. Specifically, if $\mathbf{h}_L$ and $\mathbf{h}_R$ are d-dimensional vector representations of left and right children nodes respectively,

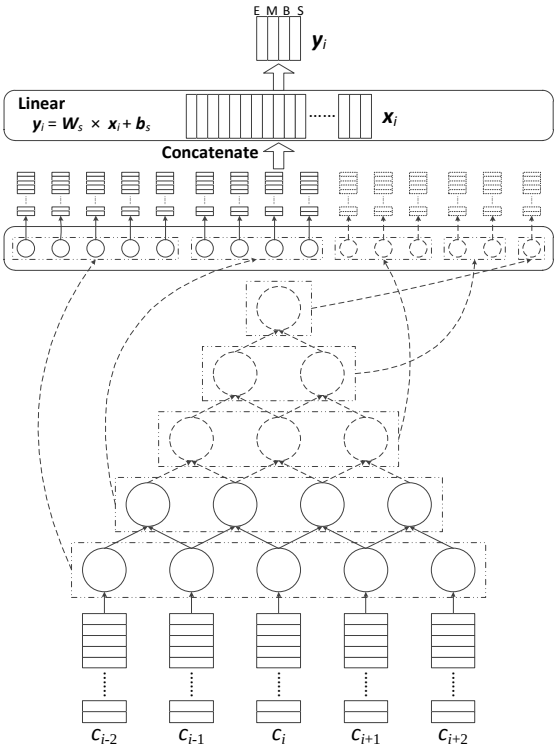

Figure 3: Architecture of Gated Recursive Neural Network for Chinese word segmentation.

their parent node $\mathbf{h}_P$ will be a d-dimensional vector as well, calculated as:

$$\mathbf{h}_P = g\left(\mathbf{W}\begin{bmatrix}\mathbf{h}_L \\ \mathbf{h}_R\end{bmatrix}\right), \tag{3}$$

where $\mathbf{W} \in \mathbb{R}^{d \times 2d}$ and g is a non-linear function as mentioned above.

3.2 Gated Recursive Neural Network

The RNN need a topological structure to model a sequence, such as a syntactic tree. In this paper, we use a directed acyclic graph (DAG), as showing in Figure 3, to model the combinations of the input characters, in which two consecutive nodes in the lower layer are combined into a single node in the upper layer via the operation as Eq. (3).

In fact, the DAG structure can model the combinations of characters by continuously mixing the information from the bottom layer to the top layer. Each neuron can be regarded as a complicated feature composition of its governed characters, similar to the discrete feature based models. The difference between them is that the neural one automatically learns the complicated combinations while the conventional one need manually design them.

When the children nodes combine into their parent node, the combination information of two children nodes is also merged and preserved by their parent node.

Although the mechanism above seem to work well, it can not sufficiently model the complicated combination features for its simplicity in practice.

Inspired by the success of the gated recurrent neural network (Cho et al., 2014b; Chung et al., 2014), we propose a gated recursive neural network (GRNN) by introducing two kinds of gates, namely "reset gate" and "update gate". Specifically, there are two reset gates, $\mathbf{r}_L$ and $\mathbf{r}_R$, partially reading the information from left child and right child respectively. And the update gates $\mathbf{z}_N$, $\mathbf{z}_L$ and $\mathbf{z}_R$ decide what to preserve when combining the children's information. Intuitively, these gates seems to decide how to update and exploit the combination information.

In the case of word segmentation, for each character c_i of a given sentence $c_{1:n}$, we first represent each context character c_j into its corresponding embedding $\mathbf{c}_j$, where $i - w_1 \leq j \leq i + w_2$ and the definitions of w_1 and w_2 are as same as mentioned above.

Then, the embeddings are sent to the first layer of GRNN as inputs, whose outputs are recursively applied to upper layers until it outputs a single fixed-length vector.

The outputs of the different neurons can be regarded as the different feature compositions. After concatenating the outputs of all neurons in the network, we get a new big vector $\mathbf{x}_i$. Next, we receive the tag score vector $\mathbf{y}_i$ for character c_j by a linear transformation of $\mathbf{x}_i$:

$$\mathbf{y}_i = \mathbf{W}_s \times \mathbf{x}_i + \mathbf{b}_s, \tag{4}$$

where $b_s \in \mathbb{R}^{|T|}$, $\mathbf{W}_s \in \mathbb{R}^{|T| \times Q}$. $Q = q \times d$ is dimensionality of the concatenated vector $\mathbf{x}_i$, where q is the number of nodes in the network.

3.3 Gated Recursive Unit

GRNN consists of the minimal structures, gated recursive units, as showing in Figure 4.

By assuming that the window size is w, we will have recursion layer $l \in [1, w]$. At each recursion layer l, the activation of the j-th hidden node $\mathbf{h}_j^{(l)} \in \mathbb{R}^d$ is computed as

$$\mathbf{h}_j^{(l)} = \begin{cases} \mathbf{z}_N \odot \hat{\mathbf{h}}_j^l + \mathbf{z}_L \odot \mathbf{h}_{j-1}^{l-1} + \mathbf{z}_R \odot \mathbf{h}_j^{l-1}, & l > 1, \\ \mathbf{c}_j, & l = 1, \end{cases} \tag{5}$$

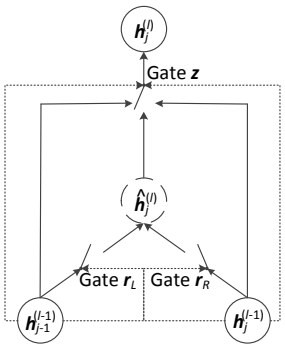

Figure 4: Our proposed gated recursive unit.

where $\mathbf{z}_N$, $\mathbf{z}_L$ and $\mathbf{z}_R \in \mathbb{R}^d$ are update gates for new activation $\hat{\mathbf{h}}_j^l$, left child node $\mathbf{h}_{j-1}^{l-1}$ and right child node $\mathbf{h}_j^{l-1}$ respectively, and $\odot$ indicates element-wise multiplication.

The update gates can be formalized as:

$$\mathbf{z} = \begin{bmatrix} \mathbf{z}_N \\ \mathbf{z}_L \\ \mathbf{z}_R \end{bmatrix} = \begin{bmatrix} 1/Z \\ 1/Z \\ 1/Z \end{bmatrix} \odot \exp(\mathbf{U} \begin{bmatrix} \hat{\mathbf{h}}_j^l \\ \mathbf{h}_{j-1}^{l-1} \\ \mathbf{h}_j^{l-1} \end{bmatrix}), \tag{6}$$

where $\mathbf{U} \in \mathbb{R}^{3d \times 3d}$ is the coefficient of update gates, and $Z \in \mathbb{R}^d$ is the vector of the normalization coefficients,

$$Z_k = \sum_{i=1}^{3} [\exp(\mathbf{U} \begin{bmatrix} \hat{\mathbf{h}}_j^l \\ \mathbf{h}_{j-1}^{l-1} \\ \mathbf{h}_j^{l-1} \end{bmatrix})]_{d \times (i-1)+k}, \tag{7}$$

where $1 \leq k \leq d$.

Intuitively, three update gates are constrained by:

$$\begin{cases} [\mathbf{z}_N]_k + [\mathbf{z}_L]_k + [\mathbf{z}_R]_k = 1, & 1 \leq k \leq d; \\ [\mathbf{z}_N]_k \geq 0, & 1 \leq k \leq d; \\ [\mathbf{z}_L]_k \geq 0, & 1 \leq k \leq d; \\ [\mathbf{z}_R]_k \geq 0, & 1 \leq k \leq d. \end{cases} \tag{8}$$

The new activation $\hat{\mathbf{h}}_j^l$ is computed as:

$$\hat{\mathbf{h}}_j^l = \tanh(\mathbf{W}_{\hat{\mathbf{h}}} \begin{bmatrix} \mathbf{r}_L \odot \mathbf{h}_{j-1}^{l-1} \\ \mathbf{r}_R \odot \mathbf{h}_j^{l-1} \end{bmatrix}), \tag{9}$$

where $\mathbf{W}_{\hat{\mathbf{h}}} \in \mathbb{R}^{d \times 2d}$, $\mathbf{r}_L \in \mathbb{R}^d$, $\mathbf{r}_R \in \mathbb{R}^d$. $\mathbf{r}_L$ and $\mathbf{r}_R$ are the reset gates for left child node $\mathbf{h}_{j-1}^{l-1}$ and right child node $\mathbf{h}_j^{l-1}$ respectively, which can be

formalized as:

$$\begin{bmatrix} \mathbf{r}_L \\ \mathbf{r}_R \end{bmatrix} = \sigma(\mathbf{G} \begin{bmatrix} \mathbf{h}_{j-1}^{l-1} \\ \mathbf{h}_{j}^{l-1} \end{bmatrix}), \tag{10}$$

$$\tag{11}$$

where $\mathbf{G} \in \mathbb{R}^{2d \times 2d}$ is the coefficient of two reset gates and σ indicates the sigmoid function.

Intuiativly, the reset gates control how to select the output information of the left and right children, which results to the current new activation $\hat{\mathbf{h}}$.

By the update gates, the activation of a parent neuron can be regarded as a choice among the the current new activation $\hat{\mathbf{h}}$, the left child, and the right child. This choice allows the overall structure to change adaptively with respect to the inputs.

This gating mechanism is effective to model the combinations of the characters.

3.4 Inference

In Chinese word segmentation task, it is usually to employ the Viterbi algorithm to inference the tag sequence $t_{1:n}$ for a given input sentence $c_{1:n}$.

In order to model the tag dependencies, the previous neural network models (Collobert et al., 2011; Zheng et al., 2013; Pei et al., 2014) introduce a transition matrix $\mathbf{A}$, and each entry $\mathbf{A}_{ij}$ is the score of the transformation from tag $i \in T$ to tag $j \in T$.

Thus, the sentence-level score can be formulated as follows:

$$s(c_{1:n}, t_{1:n}, \theta) = \sum_{i=1}^{n} \left(\mathbf{A}_{t_{i-1}t_i} + f_\theta(t_i | c_{i-w_1:i+w_2}) \right), \tag{12}$$

where $f_\theta(t_i | c_{i-w_1:i+w_2})$ is the score for choosing tag t_i for the i-th character by our proposed GRNN (Eq. (4)). The parameter set of our model is $\theta = (\mathbf{M}, \mathbf{W}_s, \mathbf{b}_s, \mathbf{W}_{\hat{\mathbf{h}}}, \mathbf{U}, \mathbf{G}, \mathbf{A})$.

4 Training

4.1 Layer-wise Training

Deep neural network with multiple hidden layers is very difficult to train for its problem of gradient diffusion and risk of overfitting.

Following (Hinton and Salakhutdinov, 2006), we employ the layer-wise training strategy to avoid problems of overfitting and gradient vanishing. The main idea of layer-wise training is to train the network with adding the layers one by one. Specifically, we first train the neural network with the first hidden layer only. Then, we train at the network with two hidden layers after training at first layer is done and so on until we reach the top hidden layer. When getting convergency of the network with layers 1 to l , we preserve the current parameters as initial values of that in training the network with layers 1 to $l + 1$.

4.2 Max-Margin Criterion

We use the Max-Margin criterion (Taskar et al., 2005) to train our model. Intuitively, the Max-Margin criterion provides an alternative to probabilistic, likelihood based estimation methods by concentrating directly on the robustness of the decision boundary of a model. We use $Y(x_i)$ to denote the set of all possible tag sequences for a given sentence x_i and the correct tag sequence for x_i is y_i. The parameter set of our model is θ. We first define a structured margin loss $\Delta(y_i, \hat{y})$ for predicting a tag sequence $\hat{y}$ for a given correct tag sequence y_i:

$$\Delta(y_i, \hat{y}) = \sum_{j}^{n} \eta \mathbf{1}\{y_{i,j} \neq \hat{y}_j\}, \tag{13}$$

where n is the length of sentence x_i and η is a discount parameter. The loss is proportional to the number of characters with an incorrect tag in the predicted tag sequence. For a given training instance (x_i, y_i), we search for the tag sequence with the highest score:

$$y^* = \arg\max_{\hat{y} \in Y(x)} s(x_i, \hat{y}, \theta), \tag{14}$$

where the tag sequence is found and scored by the proposed model via the function $s(\cdot)$ in Eq. (12). The object of Max-Margin training is that the tag sequence with highest score is the correct one: $y^* = y_i$ and its score will be larger up to a margin to other possible tag sequences $\hat{y} \in Y(x_i)$:

$$s(x, y_i, \theta) \geq s(x, \hat{y}, \theta) + \Delta(y_i, \hat{y}). \tag{15}$$

This leads to the regularized objective function for m training examples:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} l_i(\theta) + \frac{\lambda}{2} \|\theta\|_2^2, \tag{16}$$

$$l_i(\theta) = \max_{\hat{y} \in Y(x_i)} (s(x_i, \hat{y}, \theta) + \Delta(y_i, \hat{y})) - s(x_i, y_i, \theta). \tag{17}$$

By minimizing this object, the score of the correct tag sequence y_i is increased and score of the highest scoring incorrect tag sequence $\hat{y}$ is decreased. The objective function is not differentiable due to the hinge loss. We use a generalization of gradient descent called subgradient method (Ratliff et al., 2007) which computes a gradient-like direction.

Following (Socher et al., 2013a), we minimize the objective by the diagonal variant of AdaGrad (Duchi et al., 2011) with minibatchs. The parameter update for the i-th parameter $\theta_{t,i}$ at time step t is as follows:

$$\theta_{t,i} = \theta_{t-1,i} - \frac{\alpha}{\sqrt{\sum_{\tau=1}^{t} g_{\tau,i}^2}} g_{t,i}, \qquad (18)$$

where α is the initial learning rate and $g_\tau \in \mathbb{R}^{|\theta_i|}$ is the subgradient at time step τ for parameter θ_i.

5 Experiments

We evaluate our model on two different kinds of texts: newswire texts and micro-blog texts. For evaluation, we use the standard Bakeoff scoring program to calculate precision, recall, F1-score.

5.1 Word Segmentation on Newswire Texts

5.1.1 Datasets

We use three popular datasets, PKU, MSRA and CTB6, to evaluate our model on newswire texts. The PKU and MSRA data are provided by the second International Chinese Word Segmentation Bakeoff (Emerson, 2005), and CTB6 is from Chinese TreeBank 6.0 (LDC2007T36) (Xue et al., 2005), which is a segmented, part-of-speech tagged, and fully bracketed corpus in the constituency formalism. These datasets are commonly used by previous state-of-the-art models and neural network models. In addition, we use the first 90% sentences of the training data as training set and the rest 10% sentences as development set for PKU and MSRA datasets, and we divide the training, development and test sets according to (Yang and Xue, 2012) for the CTB6 dataset.

All datasets are preprocessed by replacing the Chinese idioms and the continuous English characters and digits with a unique flag.

5.1.2 Hyper-parameters

We set the hyper-parameters of the model as list in Table 1 via experiments on development set. In addition, we set the batch size to 20. And we find that it is a good balance between model performance and efficiency to set character embedding size $d = 50$. In fact, the larger embedding size leads to higher cost of computational resource, while lower dimensionality of the character embedding seems to underfit according to the experiment results.

Window size	$k = 5$
Character embedding size	$d = 50$
Initial learning rate	$\alpha = 0.3$
Margin loss discount	$\eta = 0.2$
Regularization	$\lambda = 10^{-4}$
Dropout rate on input layer	$p = 20\%$

Table 1: Hyper-parameter settings.

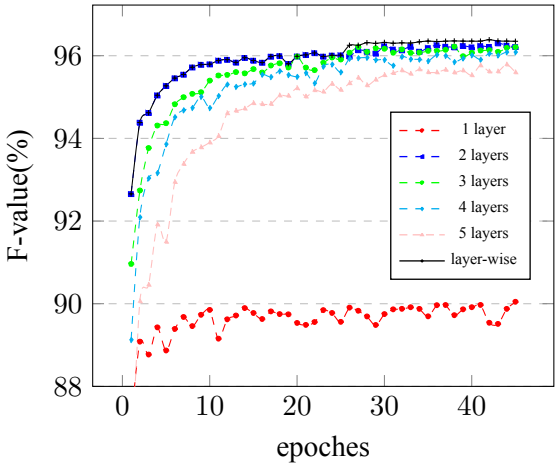

Figure 5: Performance of different models with or without layer-wise training strategy on PKU development set.

Deep neural networks contain multiple nonlinear hidden layers are always hard to train for it is easy to overfit. Several methods have been used in neural models to avoid overfitting, such as early stop and weight regularization. Dropout (Srivastava et al., 2014) is also one of the popular strategies to avoid overfitting when training the deep neural networks. Hence, we utilize the dropout strategy in this work. Specifically, dropout is to temporarily remove the neuron away with a fixed probability p independently, along with the incoming and outgoing connections of it. As a result, we find dropout on the input layer with probability $p = 20\%$ is a good tradeoff between model efficiency and performance.

models	without layer-wise			with layer-wise		
	P	R	F	P	R	F
GRNN (1 layer)	90.7	89.6	90.2	-	-	-
GRNN (2 layers)	96.0	95.6	95.8	96.0	95.6	95.8
GRNN (3 layers)	95.9	95.4	95.7	96.0	95.7	95.9
GRNN (4 layers)	95.6	95.2	95.4	96.1	95.7	95.9
GRNN (5 layers)	95.3	94.7	95.0	96.1	95.7	95.9

Table 2: Performance of different models with or without layer-wise training strategy on PKU test set.

5.1.3 Layer-wise Training

We first investigate the effects of the layer-wise training strategy. Since we set the size of context window to five, there are five recursive layers in our architecture. And we train the networks with the different numbers of recursion layers. Due to the limit of space, we just give the results on PKU dataset.

Figure 5 gives the convergence speeds of the five models with different numbers of layers and the model with layer-wise training strategy on development set of PKU dataset. The model with one layer just use the neurons of the lowest layer in final linear score function. Since there are no non-linear layer, its seems to underfit and perform poorly. The model with two layers just use the neurons in the lowest two layers, and so on. The model with five layers use all the neurons in the network. As we can see, the layer-wise training strategy lead to the fastest convergence and the best performance.

Table 2 shows the performances on PKU test set. The performance of the model with layer-wise training strategy is always better than that without layer-wise training strategy. With the increase of the number of layers, the performance also increases and reaches the stable high performance until getting to the top layer.

5.1.4 Results

We first compare our model with the previous neural approaches on PKU, MSRA and CTB6 datasets as showing in Table 3. The character embeddings of the models are random initialized. The performance of word segmentation is significantly boosted by exploiting the gated recursive architecture, which can better model the combinations of the context characters than the previous neural models.

Previous works have proven it will greatly improve the performance to exploit the pre-trained character embeddings instead of that with random initialization. Thus, we pre-train the embeddings on a huge unlabeled data, the Chinese Wikipedia corpus, with word2vec toolkit (Mikolov et al., 2013). By using these obtained character embeddings, our model receives better performance and still outperforms the previous neural models with pre-trained character embeddings. The detailed results are shown in Table 4 (1st to 3rd rows).

Inspired by (Pei et al., 2014), we utilize the bigram feature embeddings in our model as well. The concept of feature embedding is quite similar to that of character embedding mentioned above. Specifically, each context feature is represented as a single vector called feature embedding. In this paper, we only use the simply bigram feature embeddings initialized by the average of two embeddings of consecutive characters element-wisely.

Although the model of Pei et al. (2014) greatly benefits from the bigram feature embeddings, our model just obtains a small improvement with them. This difference indicates that our model has well modeled the combinations of the characters and do not need much help of the feature engineering. The detailed results are shown in Table 4 (4-th and 6-th rows).

Table 5 shows the comparisons of our model with the state-of-the-art systems on F-value. The model proposed by Zhang and Clark (2007) is a word-based segmentation method, which exploit features of complete words, while remains of the list are all character-based word segmenters, whose features are mostly extracted from the context characters. Moreover, some systems (such as Sun and Xu (2011) and Zhang et al. (2013)) also exploit kinds of extra information such as the unlabeled data or other knowledge. Although our model only uses simple bigram features, it outperforms the previous state-of-the-art methods which use more complex features.

models	PKU			MSRA			CTB6		
	P	R	F	P	R	F	P	R	F
(Zheng et al., 2013)	92.8	92.0	92.4	92.9	93.6	93.3	94.0*	93.1*	93.6*
(Pei et al., 2014)	93.7	93.4	93.5	94.6	94.2	94.4	94.4*	93.4*	93.9*
GRNN	**96.0**	**95.7**	**95.9**	**96.3**	**96.1**	**96.2**	**95.4**	**95.2**	**95.3**

Table 3: Performances on PKU, MSRA and CTB6 test sets with random initialized character embeddings.

models	PKU			MSRA			CTB6		
	P	R	F	P	R	F	P	R	F
+Pre-train									
(Zheng et al., 2013)	93.5	92.2	92.8	94.2	93.7	93.9	93.9*	93.4*	93.7*
(Pei et al., 2014)	94.4	93.6	94.0	95.2	94.6	94.9	94.2*	93.7*	94.0*
GRNN	96.3	95.9	96.1	96.2	96.3	96.2	95.8	95.4	95.6
+bigram									
GRNN	**96.6**	96.2	**96.4**	**97.5**	97.3	97.4	**95.9**	**95.7**	**95.8**
+Pre-train+bigram									
(Pei et al., 2014)	-	95.2	-	-	97.2	-	-	-	-
GRNN	96.5	**96.3**	**96.4**	97.4	**97.8**	**97.6**	95.8	**95.7**	**95.8**

Table 4: Performances on PKU, MSRA and CTB6 test sets with pre-trained and bigram character embeddings.

models	PKU	MSRA	CTB6
(Tseng et al., 2005)	95.0	96.4	-
(Zhang and Clark, 2007)	95.1	97.2	-
(Sun and Xu, 2011)	-	-	95.7
(Zhang et al., 2013)	96.1	97.4	-
This work	**96.4**	**97.6**	**95.8**

Table 5: Comparison of GRNN with the state-of-the-art methods on PKU, MSRA and CTB6 test sets.

5.2 Word Segmentation on Micro-blog Texts

5.2.1 Dataset

We use the NLPCC 2015 dataset[1] (Qiu et al., 2015) to evaluate our model on micro-blog texts. The NLPCC 2015 data are provided by the shared task in the 4th CCF Conference on Natural Language Processing & Chinese Computing (NLPCC 2015): Chinese Word Segmentation and POS Tagging for micro-blog Text. Different with the popular used newswire dataset, the NLPCC 2015 dataset is collected from Sina Weibo[2], which consists of the relatively informal texts from micro-blog with the various topics, such as finance, sports, entertainment, and so on. The information of the dataset is shown in Table 6.

To train our model, we also use the first 90% sentences of the training data as training set and the rest 10% sentences as development set.

Here, we use the default setting of CRF++ toolkit with the feature templates as shown in Table 7. The same feature templates are also used for FNLP.

5.2.2 Results

Since the NLPCC 2015 dataset is a new released dataset, we compare our model with the two popular open source toolkits for sequence labeling task: FNLP[3] (Qiu et al., 2013) and CRF++[4]. Our model uses pre-trained and bigram character embeddings.

Table 8 shows the comparisons of our model with the other systems on NLPCC 2015 dataset.

6 Related Work

Chinese word segmentation has been studied with considerable efforts in the NLP community. The most popular word segmentation method is based on sequence labeling (Xue, 2003). Recently, researchers have tended to explore neural network

[1] http://nlp.fudan.edu.cn/nlpcc2015/

[2] http://www.weibo.com/

[3] https://github.com/xpqiu/fnlp/

[4] http://taku910.github.io/crfpp/

[*] The result is from our own implementation of the corresponding method.

Dataset	Sents	Words	Chars	Word Types	Char Types	OOV Rate
Training	10,000	215,027	347,984	28,208	39,71	-
Test	5,000	106,327	171,652	18,696	3,538	7.25%
Total	15,000	322,410	520,555	35,277	4,243	-

Table 6: Statistical information of NLPCC 2015 dataset.

unigram feature	$c_{-2}, c_{-1}, c_0, c_{+1}, c_{+2}$
bigram feature	$c_{-1} \circ c_0, c_0 \circ c_{+1}$
trigram feature	$c_{-2} \circ c_{-1} \circ c_0, c_{-1} \circ c_0 \circ c_{+1}, c_0 \circ c_{+1} \circ c_{+2}$

Table 7: Templates of CRF++ and FNLP.

models	P	R	F
CRF++	93.3	93.2	93.3
FNLP	94.1	93.9	94.0
This work	94.7	94.8	94.8

Table 8: Performances on NLPCC 2015 dataset.

based approaches (Collobert et al., 2011) to reduce efforts of the feature engineering (Zheng et al., 2013; Qi et al., 2014). However, the features of all these methods are the concatenation of the embeddings of the context characters.

Pei et al. (2014) also used neural tensor model (Socher et al., 2013b) to capture the complicated interactions between tags and context characters. But the interactions depend on the number of the tensor slices, which cannot be too large due to the model complexity. The experiments also show that the model of (Pei et al., 2014) greatly benefits from the further bigram feature embeddings, which shows that their model cannot even handle the interactions of the consecutive characters. Different with them, our model just has a small improvement with the bigram feature embeddings, which indicates that our approach has well modeled the complicated combinations of the context characters, and does not need much help of further feature engineering.

More recently, Cho et al. (2014a) also proposed a gated recursive convolutional neural network in machine translation task to solve the problem of varying lengths of sentences. However, their approach only models the update gate, which can not tell whether the information is from the current state or from sub notes in update stage without reset gate. Instead, our approach models two kinds of gates, reset gate and update gate, by incorporating which we can better model the combinations of context characters via selection function of reset gate and collection function of update gate.

7 Conclusion

In this paper, we propose a gated recursive neural network (GRNN) to explicitly model the combinations of the characters for Chinese word segmentation task. Each neuron in GRNN can be regarded as a different combination of the input characters. Thus, the whole GRNN has an ability to simulate the design of the sophisticated features in traditional methods. Experiments show that our proposed model outperforms the state-of-the-art methods on three popular benchmark datasets.

Despite Chinese word segmentation being a specific case, our model can be easily generalized and applied to other sequence labeling tasks. In future work, we would like to investigate our proposed GRNN on other sequence labeling tasks.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. This work was partially funded by the National Natural Science Foundation of China (61472088, 61473092), the National High Technology Research and Development Program of China (2015AA015408), Shanghai Science and Technology Development Funds (14ZR1403200), Shanghai Leading Academic Discipline Project (B114).

References

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of Workshop on Syntax, Semantics and Structure in Statistical Translation*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of EMNLP*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

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.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

T. Emerson. 2005. The second international Chinese word segmentation bakeoff. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, pages 123–133. Jeju Island, Korea.

Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

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 Chang Baobao. 2014. Max-margin tensor neural network for chinese word segmentation. In *Proceedings of ACL*.

Jordan B Pollack. 1990. Recursive distributed representations. *Artificial Intelligence*, 46(1):77–105.

Yanjun Qi, Sujatha G Das, Ronan Collobert, and Jason Weston. 2014. Deep learning for character-based information extraction. In *Advances in Information Retrieval*, pages 668–674. Springer.

Xipeng Qiu, Qi Zhang, and Xuanjing Huang. 2013. FudanNLP: A toolkit for Chinese natural language processing. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*.

Xipeng Qiu, Peng Qian, Liusong Yin, and Xuanjing Huang. 2015. Overview of the NLPCC 2015 shared task: Chinese word segmentation and POS tagging for micro-blog texts. *arXiv preprint arXiv:1505.07599*.

Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. 2007. (online) subgradient methods for structured prediction. In *Eleventh International Conference on Artificial Intelligence and Statistics (AIStats)*.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013a. Parsing with compositional vector grammars. In *In Proceedings of the ACL conference*. Citeseer.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013b. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*.

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

Weiwei Sun and Jia Xu. 2011. Enhancing Chinese word segmentation using unlabeled data. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 970–979. Association for Computational Linguistics.

Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. 2005. Learning structured prediction models: A large margin approach. In *Proceedings of the international conference on Machine learning*.

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A conditional random field word segmenter for sighan bakeoff 2005. In *Proceedings of the fourth SIGHAN workshop on Chinese language Processing*, volume 171.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The Penn Chinese TreeBank: Phrase structure annotation of a large corpus. *Natural language engineering*, 11(2):207–238.

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

Yaqin Yang and Nianwen Xue. 2012. Chinese comma disambiguation for discourse analysis. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 786–794. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2007. Chinese segmentation with a word-based perceptron algorithm. In *ACL*.

Longkai Zhang, Houfeng Wang, Xu Sun, and Mairgup Mansur. 2013. Exploring representations from unlabeled data with co-training for Chinese word segmentation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

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