Max-Margin Tensor Neural Network for Chinese Word Segmentation

Wenzhe Pei **Tao Ge** **Baobao Chang**[*]
Key Laboratory of Computational Linguistics, Ministry of Education
School of Electronics Engineering and Computer Science, Peking University
Beijing, P.R.China, 100871
{peiwenzhe,getao,chbb}@pku.edu.cn

Abstract

Recently, neural network models for natural language processing tasks have been increasingly focused on for their ability to alleviate the burden of manual feature engineering. In this paper, we propose a novel neural network model for Chinese word segmentation called Max-Margin Tensor Neural Network (**MMTNN**). By exploiting tag embeddings and tensor-based transformation, **MMTNN** has the ability to model complicated interactions between tags and context characters. Furthermore, a new tensor factorization approach is proposed to speed up the model and avoid overfitting. Experiments on the benchmark dataset show that our model achieves better performances than previous neural network models and that our model can achieve a competitive performance with minimal feature engineering. Despite Chinese word segmentation being a specific case, **MMTNN** can be easily generalized and applied to other sequence labeling tasks.

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 where each character is assigned a tag indicating its position in the word. These systems are effective because researchers can incorporate a large body of handcrafted features into the models. However, the ability of these models is restricted by the design of features and the number of features could be so large that the result models are too large for practical use and prone to overfit on training corpus.

Recently, neural network models have been increasingly focused on for their ability to minimize the effort in feature engineering. Collobert et al. (2011) developed the SENNA system that approaches or surpasses the state-of-the-art systems on a variety of sequence labeling tasks for English. Zheng et al. (2013) applied the architecture of Collobert et al. (2011) to Chinese word segmentation and POS tagging and proposed a perceptron-style algorithm to speed up the training process with negligible loss in performance.

Workable as previous neural network models seem, a limitation of them to be pointed out is that the tag-tag interaction, tag-character interaction and character-character interaction are not well modeled. In conventional feature-based linear (log-linear) models, these interactions are explicitly modeled as features. Take phrase “打篮球(play basketball)” as an example, assuming we are labeling character C_0=“篮”, possible features could be:

$$f_1 = \begin{cases} 1 & C_{-1}\text{=“打” and } C_1\text{=“球” and } y_0\text{=“B”} \\ 0 & \text{else} \end{cases}$$

$$f_2 = \begin{cases} 1 & C_0\text{=“篮” and } y_0\text{=“B” and } y_{-1}\text{=“S”} \\ 0 & \text{else} \end{cases}$$

To capture more interactions, researchers have designed a large number of features based on linguistic intuition and statistical information. In previous neural network models, however, hardly can such interactional effects be fully captured relying only on the simple transition score and the single non-linear transformation (See section 2). In order to address this problem, we propose a new model called Max-Margin Tensor Neural Network (**MMTNN**) that explicitly models the interactions

[*] Corresponding author

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 293–303,
Baltimore, Maryland, USA, June 23-25 2014. ©2014 Association for Computational Linguistics

between tags and context characters by exploiting tag embeddings and tensor-based transformation. Moreover, we propose a tensor factorization approach that effectively improves the model efficiency and prevents from overfitting. We evaluate the performance of Chinese word segmentation on the PKU and MSRA benchmark datasets in the second International Chinese Word Segmentation Bakeoff (Emerson, 2005) which are commonly used for evaluation of Chinese word segmentation. Experiment results show that our model outperforms other neural network models.

Although we focus on the question that how far we can go without using feature engineering in this paper, the study of deep learning for NLP tasks is still a new area in which it is currently challenging to surpass the state-of-the-art without additional features. Following Mansur et al. (2013), we wonder how well our model can perform with minimal feature engineering. Therefore, we integrate additional simple character bigram features into our model and the result shows that our model can achieve a competitive performance that other systems hardly achieve unless they use more complex task-specific features.

The main contributions of our work are as follows:

- We propose a Max-Margin Tensor Neural Network for Chinese word segmentation without feature engineering. The test results on the benchmark dataset show that our model outperforms previous neural network models.

- We propose a new tensor factorization approach that models each tensor slice as the product of two low-rank matrices. Not only does this approach improve the efficiency of our model but also it avoids the risk of overfitting.

- Compared with previous works that use a large number of handcrafted features, our model can achieve a competitive performance with minimal feature engineering.

- Despite Chinese word segmentation being a specific case, our approach can be easily generalized to other sequence labeling tasks.

The remaining part of this paper is organized as follows. Section 2 describes the details of conventional neural network architecture. Section 3 describes the details of our model. Experiment results are reported in Section 4. Section 5 reviews the related work. The conclusions are given in Section 6.

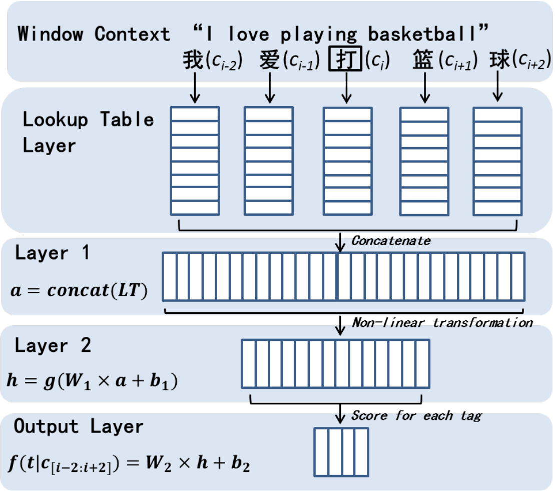
Window Context "I love playing basketball"
我(c_{i-2}) 爱(c_{i-1}) 打(c_i) 篮(c_{i+1}) 球(c_{i+2})
Lookup Table Layer
Concatenate
Layer 1
$a = concat(LT)$
Non-linear transformation
Layer 2
$h = g(W_1 \times a + b_1)$
Score for each tag
Output Layer
$f(t|c_{[i-2:i+2]}) = W_2 \times h + b_2$

Figure 1: Conventional Neural Network

2 Conventional Neural Network

2.1 Lookup Table

The idea of distributed representation for symbolic data is one of the most important reasons why the neural network works. It was proposed by Hinton (1986) and has been a research hot spot for more than twenty years (Bengio et al., 2003; Collobert et al., 2011; Schwenk et al., 2012; Mikolov et al., 2013a). Formally, in the Chinese word segmentation task, we have a character dictionary D of size $|D|$. Unless otherwise specified, the character dictionary is extracted from the training set and unknown characters are mapped to a special symbol that is not used elsewhere. Each character $c \in D$ is represented as a real-valued vector (character embedding) $Embed(c) \in \mathbb{R}^d$ where d is the dimensionality of the vector space. The character embeddings are then stacked into a embedding matrix $M \in \mathbb{R}^{d \times |D|}$. For a character $c \in D$ that has an associated index k, the corresponding character embedding $Embed(c) \in \mathbb{R}^d$ is retrieved by the Lookup Table layer as shown in Figure 1:

$$Embed(c) = Me_k \tag{1}$$

Here $e_k \in \mathbb{R}^{|D|}$ is a binary vector which is zero in all positions except at k-th index. The Lookup Table layer can be seen as a simple projection layer where the character embedding for each context character is achieved by table lookup operation according to their indices. The embedding matrix M is initialized with small random numbers and trained by back-propagation. We will analyze in more detail about the effect of character embeddings in Section 4.

2.2 Tag Scoring

The most common tagging approach is the window approach. The window approach assumes that the tag of a character largely depends on its neighboring characters. Given an input sentence $c_{[1:n]}$, a window of size w slides over the sentence from character c_1 to c_n. We set $w = 5$ in all experiments. As shown in Figure 1, at position $c_i, 1 \leq i \leq n$, the context characters are fed into the Lookup Table layer. The characters exceeding the sentence boundaries are mapped to one of two special symbols, namely "start" and "end" symbols. The character embeddings extracted by the Lookup Table layer are then concatenated into a single vector $a \in \mathbb{R}^{H_1}$, where $H_1 = w \cdot d$ is the size of Layer 1. Then a is fed into the next layer which performs linear transformation followed by an element-wise activation function g such as *tanh*, which is used in our experiments:

$$h = g(W_1 a + b_1) \tag{2}$$

where $W_1 \in \mathbb{R}^{H_2 \times H_1}$, $b_1 \in \mathbb{R}^{H_2 \times 1}$, $h \in \mathbb{R}^{H_2}$. H_2 is a hyper-parameter which is the number of hidden units in Layer 2. Given a set of tags T of size $|T|$, a similar linear transformation is performed except that no non-linear function is followed:

$$f(\boldsymbol{t}|c_{[i-2:i+2]}) = W_2 h + b_2 \tag{3}$$

where $W_2 \in \mathbb{R}^{|T| \times H_2}$, $b_2 \in \mathbb{R}^{|T| \times 1}$. $f(\boldsymbol{t}|c_{[i-2:i+2]}) \in \mathbb{R}^{|T|}$ is the score vector for each possible tag. In Chinese word segmentation, the most prevalent tag set T is BMES tag set, which uses 4 tags to carry word boundary information. It uses B, M, E and S to denote the Beginning, the Middle, the End of a word and a Single character forming a word respectively. We use this tag set in our method.

2.3 Model Training and Inference

Despite sharing commonalities mentioned above, previous work models the segmentation task differently and therefore uses different training and inference procedure. Mansur et al. (2013) modeled Chinese word segmentation as a series of classification task at each position of the sentence in which the tag score is transformed into probability using softmax function:

$$p(t_i|c_{[i-2:i+2]}) = \frac{exp(f(t_i|c_{[i-2:i+2]}))}{\sum_{t'} exp(f(t'|c_{[i-2:i+2]}))}$$

The model is then trained in MLE-style which maximizes the log-likelihood of the tagged data. Obviously, it is a local model which cannot capture the dependency between tags and does not support to infer the tag sequence globally.

To model the tag dependency, previous neural network models (Collobert et al., 2011; Zheng et al., 2013) introduce a transition score A_{ij} for jumping from tag $i \in T$ to tag $j \in T$. For a input sentence $c_{[1:n]}$ with a tag sequence $t_{[1:n]}$, a sentence-level score is then given by the sum of transition and network scores:

$$s(c_{[1:n]}, t_{[1:n]}, \theta) = \sum_{i=1}^{n} (A_{t_{i-1} t_i} + f_\theta(t_i|c_{[i-2:i+2]})) \tag{4}$$

where $f_\theta(t_i|c_{[i-2:i+2]})$ indicates the score output for tag t_i at the i-th character by the network with parameters $\theta = (M, A, W_1, b_1, W_2, b_2)$. Given the sentence-level score, Zheng et al. (2013) proposed a perceptron-style training algorithm inspired by the work of Collins (2002). Compared with Mansur et al. (2013), their model is a global one where the training and inference is performed at sentence-level.

Workable as these methods seem, one of the limitations of them is that the tag-tag interaction and the neural network are modeled seperately. The simple tag-tag transition neglects the impact of context characters and thus limits the ability to capture flexible interactions between tags and context characters. Moreover, the simple non-linear transformation in equation (2) is also poor to model the complex interactional effects in Chinese word segmentation.

3 Max-Margin Tensor Neural Network

3.1 Tag Embedding

To better model the tag-tag interaction given the context characters, distributed representation for tags instead of traditional discrete symbolic representation is used in our model. Similar to character embeddings, given a fixed-sized tag set T, the tag embeddings for tags are stored in a tag embedding matrix $L \in \mathbb{R}^{d \times |T|}$, where d is the dimensionality

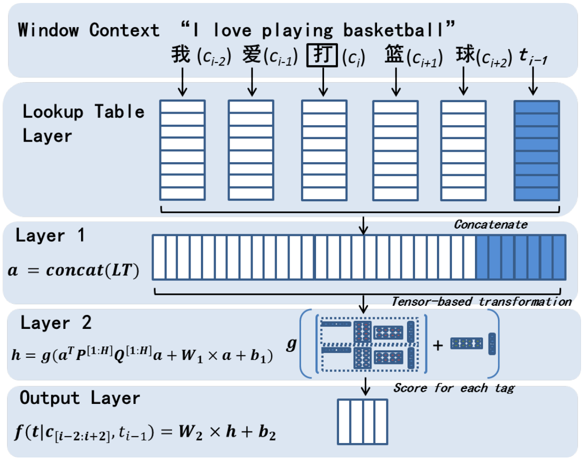
Window Context "I love playing basketball"
我 (c_{i-2}) 爱 (c_{i-1}) 打 (c_i) 篮 (c_{i+1}) 球 (c_{i+2}) t_{i-1}
Lookup Table Layer
Concatenate
Layer 1
$a = concat(LT)$
Tensor-based transformation
Layer 2
$h = g(a^T P^{[1:H]} Q^{[1:H]} a + W_1 \times a + b_1)$
g
+
Score for each tag
Output Layer
$f(t|c_{[i-2:i+2]}, t_{i-1}) = W_2 \times h + b_2$

Figure 2: Max-Margin Tensor Neural Network

of the vector space (same with character embeddings). Then the tag embedding $Embed(t) \in \mathbb{R}^d$ for tag $t \in T$ with index k can be retrieved by the lookup operation:

$$Embed(t) = Le_k \tag{5}$$

where $e_k \in \mathbb{R}^{|T| \times 1}$ is a binary vector which is zero in all positions except at k-th index. The tag embeddings start from a random initialization and can be automatically trained by back-propagation. Figure 2 shows the new Lookup Table layer with tag embeddings. Assuming we are at the i-th character of a sentence, besides the character embeddings, the tag embeddings of the previous tags are also considered[1]. For a fast tag inference, only the previous tag t_{i-1} is used in our model even though a longer history of tags can be considered. The concatenation operation in Layer 1 then concatenates the character embeddings and tag embedding together into a long vector a. In this way, the tag representation can be directly incorporated in the neural network so that the tag-tag interaction and tag-character interaction can be explicitly modeled in deeper layers (See Section 3.2). Moreover, the transition score in equation (4) is not necessary in our model, because, by incorporating tag embedding into the neural network, the effect of tag-tag interaction and tag-character interaction are covered uniformly in one same model. Now equation (4) can be rewritten as follows:

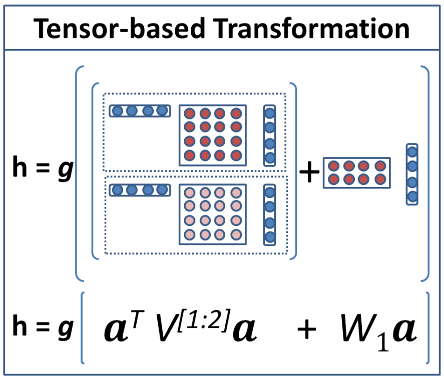

Figure 3: The tensor-based transformation in Layer 2. a is the input from Layer 1. V is the tensor parameter. Each dashed box represents one of the H_2-many tensor slices, which defines the bilinear form on vector a.

$$s(c_{[1:n]}, t_{[1:n]}, \theta) = \sum_{i=1}^{n} f_\theta(t_i | c_{[i-2:i+2]}, t_{i-1}) \tag{6}$$

where $f_\theta(t_i | c_{[i-2:i+2]}, t_{i-1})$ is the score output for tag t_i at the i-th character by the network with parameters θ. Like Collobert et al. (2011) and Zheng et al. (2013), our model is also trained at sentence-level and carries out inference globally.

3.2 Tensor Neural Network

A tensor is a geometric object that describes relations between vectors, scalars, and other tensors. It can be represented as a multi-dimensional array of numerical values. An advantage of the tensor is that it can explicitly model multiple interactions in data. As a result, tensor-based model have been widely used in a variety of tasks (Salakhutdinov et al., 2007; Krizhevsky et al., 2010; Socher et al., 2013b).

In Chinese word segmentation, a proper modeling of the tag-tag interaction, tag-character interaction and character-character interaction is very important. In linear models, these kinds of interactions are usually modeled as features. In conventional neural network models, however, the input embeddings only implicitly interact through the non-linear function which can hardly model the complexity of the interactions. Given the advantage of tensors, we apply a tensor-based transformation to the input vector. Formally, we use a 3-way tensor $V^{[1:H_2]} \in \mathbb{R}^{H_2 \times H_1 \times H_1}$ to directly model the interactions, where H_2 is the size of

[1] We also tried the architecture in which the tag embedding of current tag is also considered, but this did not bring much improvement and runs slower

Layer 2 and $H_1 = (w+1) \cdot d$ is the size of concatenated vector a in Layer 1 as shown in Figure 2. Figure 3 gives an example of the tensor-based transformation[2]. The output of a tensor product is a vector $z \in \mathbb{R}^{H_2}$ where each dimension z_i is the result of the bilinear form defined by each tensor slice $V^{[i]} \in \mathbb{R}^{H_1 \times H_1}$:

$$z = a^T V^{[1:H_2]} a; z_i = a^T V^{[i]} a = \sum_{j,k} V^{[i]}_{jk} a_j a_k \tag{7}$$

Since vector a is the concatenation of character embeddings and the tag embedding, equation (7) can be written in the following form:

$$z_i = \sum_{p,q} \sum_{j,k} V^{[i]}_{(p,q,j,k)} E^{[p]}_j E^{[q]}_k$$

where $E^{[p]}_j$ is the j-th element of the p-th embedding in Lookup Table layer and $V^{[i]}_{(p,q,j,k)}$ is the corresponding coefficient for $E^{[p]}_j$ and $E^{[q]}_k$ in $V^{[i]}$. As we can see, in each tensor slice i, the embeddings are explicitly related in a bilinear form which captures the interactions between characters and tags. The multiplicative operations between tag embeddings and character embeddings can somehow be seen as "feature combination", which are hand-designed in feature-based models. Our model learns the information automatically and encodes them in tensor parameters and embeddings. Intuitively, we can interpret each slice of the tensor as capturing a specific type of tag-character interaction and character-character interaction.

Combining the tensor product with linear transformation, the tensor-based transformation in Layer 2 is defined as:

$$h = g(a^T V^{[1:H_2]} a + W_1 a + b_1) \tag{8}$$

where $W_1 \in \mathbb{R}^{H_2 \times H_1}$, $b_1 \in \mathbb{R}^{H_2 \times 1}$, $h \in \mathbb{R}^{H_2}$. In fact, equation (2) used in previous work is a special case of equation (8) when V is set to 0.

3.3 Tensor Factorization

Despite tensor-based transformation being effective for capturing the interactions, introducing tensor-based transformation into neural network models to solve sequence labeling task is time prohibitive since the tensor product operation drastically slows down the model. Without considering matrix optimization algorithms, the complexity of the non-linear transformation in equation (2) is $O(H_1 H_2)$ while the tensor operation complexity in equation (8) is $O(H_1^2 H_2)$. The tensor-based transformation is H_1 times slower. Moreover, the additional tensor could bring millions of parameters to the model which makes the model suffer from the risk of overfitting. To remedy this, we propose a tensor factorization approach that factorizes each tensor slice as the product of two low-rank matrices. Formally, each tensor slice $V^{[i]} \in \mathbb{R}^{H_1 \times H_1}$ is factorized into two low rank matrix $P^{[i]} \in \mathbb{R}^{H_1 \times r}$ and $Q^{[i]} \in \mathbb{R}^{r \times H_1}$:

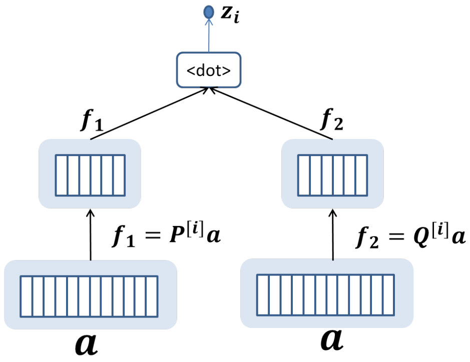

Figure 4: Tensor product with tensor factorization

$$V^{[i]} = P^{[i]} Q^{[i]}, 1 \leq i \leq H_2 \tag{9}$$

where $r \ll H_1$ is the number of factors. Substituting equation (9) into equation (8), we get the factorized tensor function:

$$h = g(a^T P^{[1:H_2]} Q^{[1:H_2]} a + W_1 a + b_1) \tag{10}$$

Figure 4 illustrates the operation in each slice of the factorized tensor. First, vector a is projected into two r-dimension vectors f_1 and f_2. Then the output z_i for each tensor slice i is the dot-product of f_1 and f_2. The complexity of the tensor operation is now $O(rH_1H_2)$. As long as r is small enough, the factorized tensor operation would be much faster than the un-factorized one and the number of free parameters would also be much smaller, which prevent the model from overfitting.

3.4 Max-Margin Training

We use the Max-Margin criterion 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 (Taskar et al., 2005). We use $Y(x_i)$ to denote the set of all possible tag sequences for

[2] The bias term is omitted in Figure 3 for simplicity

a given sentence x_i and the correct tag sequence for x_i is y_i. The parameters of our model are $\theta = \{W_1, b_1, W_2, b_2, M, L, P^{[1:H_2]}, Q^{[1:H_2]}\}$. We first define a structured margin loss $\triangle(y_i, \hat{y})$ for predicting a tag sequence $\hat{y}$ for a given correct tag sequence y_i:

$$\triangle(y_i, \hat{y}) = \sum_{j}^{n} \kappa \mathbf{1}\{y_{i,j} \neq \hat{y}_j\} \tag{11}$$

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

$$y^* = \arg\max_{\hat{y} \in Y(x)} s(x_i, \hat{y}, \theta) \tag{12}$$

where the tag sequence is found and scored by the Tensor Neural Network via the function s in equation (6). The object of Max-Margin training is that the highest scoring tag sequence is the correct one: $y^* = y_i$ and its score will be larger up to a margin to other possible tag sequences $\hat{y} \in Y(x_i)$:

$$s(x, y_i, \theta) \geq s(x, \hat{y}, \theta) + \triangle(y_i, \hat{y})$$

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

$$\begin{array}{rcl} J(\theta) & = & \frac{1}{m}\sum_{i=1}^{m} l_i(\theta) + \frac{\lambda}{2}||\theta||^2 \\ l_i(\theta) & = & \max_{\hat{y} \in Y(x_i)} (s(x_i, \hat{y}, \theta) + \triangle(y_i, \hat{y})) \\ & & -s(x_i, y_i, \theta)) \end{array} \tag{13}$$

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. The subgradient of equation (13) is:

$$\frac{\partial J}{\partial \theta} = \frac{1}{m}\sum_{i}(\frac{\partial s(x_i, \hat{y}_{max}, \theta)}{\partial \theta} - \frac{\partial s(x_i, y_i, \theta)}{\partial \theta}) + \lambda\theta$$

where $\hat{y}_{max}$ is the tag sequence with the highest score in equation (13). Following Socher et al. (2013a), we use the diagonal variant of AdaGrad (Duchi et al., 2011) with minibatchs to minimize the objective. The parameter update for the i-th parameter $\theta_{t,i}$ at time step t is as follows:

$$\theta_{t,i} = \theta_{t-1,i} - \frac{\alpha}{\sqrt{\sum_{\tau=1}^{t} g_{\tau,i}^2}} g_{t,i} \tag{14}$$

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.

	PKU	MSRA
Identical words	5.5×10^4	8.8×10^4
Total words	1.1×10^6	2.4×10^6
Identical characters	5×10^3	5×10^3
Total characters	1.8×10^6	4.1×10^6

Table 1: Details of the PKU and MSRA datasets

Window size	$w = 5$
Character(tag) embedding size	$d = 25$
Hidden unit number	$H_2 = 50$
Number of factors	$r = 10$
Initial learning rate	$\alpha = 0.2$
Margin loss discount	$\kappa = 0.2$
Regularization	$\lambda = 10^{-4}$

Table 2: Hyperparameters of our model

4 Experiment

4.1 Data and Model Selection

We use the PKU and MSRA data provided by the second International Chinese Word Segmentation Bakeoff (Emerson, 2005) to test our model. They are commonly used by previous state-of-the-art models and neural network models. Details of the data are listed in Table 1. For evaluation, we use the standard bake-off scoring program to calculate precision, recall, F1-score and out-of-vocabulary (OOV) word recall.

For model selection, we use the first 90% sentences in the training data for training and the rest 10% sentences as development data. The minibatch size is set to 20. Generally, the number of hidden units has a limited impact on the performance as long as it is large enough. We found that 50 is a good trade-off between speed and model performance. The dimensionality of character (tag) embedding is set to 25 which achieved the best performance and faster than 50- or 100-dimensional ones. We also validated on the number of factors for tensor factorization. The performance is not boosted and the training time in-

	P	R	F	OOV
CRF	87.8	85.7	86.7	57.1
NN	92.4	92.2	92.3	60.0
NN+Tag Embed	93.0	92.7	92.9	61.0
MMTNN	**93.7**	**93.4**	**93.5**	**64.2**

Table 3: Test results with different configurations. NN stands for the conventional neural network. NN+Tag Embed stands for the neural network with tag embeddings.

creases drastically when the number of factors is larger than 10. We hypothesize that larger factor size results in too many parameters to train and hence perform worse. The final hyperparameters of our model are set as in Table 2.

4.2 Experiment Results

We first perform a close test[3] on the PKU dataset to show the effect of different model configurations. We also compare our model with the CRF model (Lafferty et al., 2001), which is a widely used log-linear model for Chinese word segmentation. The input feature to the CRF model is simply the context characters (unigram feature) without any additional feature engineering. We use an open source toolkit *CRF++*[4] to train the CRF model. All the neural networks are trained using the Max-Margin approach described in Section 3.4. Table 3 summarizes the test results. As we can see, by using Tag embedding, the F-score is improved by +0.6% and OOV recall is improved by +1.0%, which shows that tag embeddings succeed in modeling the tag-tag interaction and tag-character interaction. Model performance is further boosted after using tensor-based transformation. The F-score is improved by +0.6% while OOV recall is improved by +3.2%, which denotes that tensor-based transformation captures more interactional information than simple non-linear transformation.

Another important result in Table 3 is that our neural network models perform much better than CRF-based model when only unigram features are used. Compared with CRF, there are two differences in neural network models. First, the discrete feature vector is replaced with dense character embeddings. Second, the non-linear transformation is used to discover higher level representation. In fact, CRF can be regarded as a special neural network without non-linear function (Wang and Manning, 2013). Wang and Manning (2013) conduct an empirical study on the effect of non-linearity and the results suggest that non-linear models are highly effective only when distributed representation is used. To explain why distributed representation captures more information than discrete features, we show in Table 4 the effect of character embeddings which are obtained from the lookup table of MMTNN after training. The first row lists three characters we are interested in. In each column, we list the top 5 characters that are nearest (measured by Euclidean distance) to the corresponding character in the first row according to their embeddings. As we can see, characters in the first column are all Chinese number characters and characters in the second column and the third column are all Chinese family names and Chinese punctuations respectively. Therefore, compared with discrete feature representations, distributed representation can capture the syntactic and semantic similarity between characters. As a result, the model can still perform well even if some words do not appear in the training cases.

一(one)	**李(Li)**	**。(period)**
二(two)	赵(Zhao)	，(comma)
三(three)	蒋(Jiang)	：(colon)
四(four)	孔(Kong)	？(question mark)
五(five)	冯(Feng)	“(quotation mark)
六(six)	吴(Wu)	、(Chinese comma)

Table 4: Examples of character embeddings

We further compare our model with previous neural network models on both PKU and MSRA datasets. Since Zheng et al. (2013) did not report the results on the these datasets, we re-implemented their model and tested it on the test data. The results are listed in the first three rows of Table 5, which shows that our model achieved higher F-score than the previous neural network models.

4.3 Unsupervised Pre-training

Previous work found that the performance can be improved by pre-training the character embeddings on large unlabeled data and using the obtained embeddings to initialize the character lookup table instead of random initialization

[3] No other material or knowledge except the training data is allowed

[4] `http://crfpp.googlecode.com/svn/trunk/doc/index.html?source=navbar`

Models	PKU				MSRA			
	P	R	F	OOV	P	R	F	OOV
(Mansur et al., 2013)	87.1	87.9	87.5	48.9	92.3	92.2	92.2	53.7
(Zheng et al., 2013)	92.8	92.0	92.4	63.3	92.9	93.6	93.3	55.7
MMTNN	**93.7**	**93.4**	**93.5**	**64.2**	**94.6**	**94.2**	**94.4**	**61.4**
(Mansur et al., 2013) + Pre-training	91.2	92.7	92.0	68.8	93.1	93.1	93.1	59.7
(Zheng et al., 2013) + Pre-training	93.5	92.2	92.8	69.0	94.2	93.7	93.9	64.1
MMTNN + Pre-training	**94.4**	**93.6**	**94.0**	**69.0**	**95.2**	**94.6**	**94.9**	**64.8**

Table 5: Comparison with previous neural network models

(Mansur et al., 2013; Zheng et al., 2013). Mikolov et al. (2013b) show that pre-trained embeddings can capture interesting semantic and syntactic information such as $king-man+woman \approx queen$ on English data. There are several ways to learn the embeddings on unlabeled data. Mansur et al. (2013) used the model proposed by Bengio et al. (2003) which learns the embeddings based on neural language model. Zheng et al. (2013) followed the model proposed by Collobert et al. (2008). They constructed a neural network that outputs high scores for windows that occur in the corpus and low scores for windows where one character is replaced by a random one. Mikolov et al. (2013a) proposed a faster skip-gram model *word2vec*[5] which tries to maximize classification of a word based on another word in the same sentence. In this paper, we use *word2vec* because preliminary experiments did not show differences between performances of these models but *word2vec* is much faster to train. We pre-train the embeddings on the Chinese Giga-word corpus (Graff and Chen, 2005). As shown in Table 5 (last three rows), both the F-score and OOV recall of our model boost by using pre-training. Our model still outperforms other models after pre-training.

4.4 Minimal Feature Engineering

Although we focus on the question that how far we can go without using feature engineering in this paper, the study of deep learning for NLP tasks is still a new area in which it is currently challenging to surpass the state-of-the-art without additional features. To incorporate features into the neural network, Mansur et al. (2013) proposed the feature-based neural network where each context feature is represented as feature embeddings. The idea of feature embeddings is similar to that of character embeddings described in section 2.1. Formally, we assume the extracted features form a feature dictionary D_f. Then each feature $f \in D_f$ is represented by a d-dimensional vector which is called feature embedding. Following their idea, we try to find out how well our model can perform with minimal feature engineering.

Model	PKU	MSRA
Best05(Chen et al., 2005)	95.0	96.0
Best05(Tseng et al., 2005)	95.0	96.4
(Zhang et al., 2006)	95.1	97.1
(Zhang and Clark, 2007)	94.5	97.2
(Sun et al., 2009)	95.2	97.3
(Sun et al., 2012)	95.4	97.4
(Zhang et al., 2013)	96.1	97.4
MMTNN	94.0	94.9
MMTNN + bigram	95.2	97.2

Table 6: Comparison with state-of-the-art systems

A very common feature in Chinese word segmentation is the character bigram feature. Formally, at the i-th character of a sentence $c_{[1:n]}$, the bigram features are $c_kc_{k+1}(i-3<k<i+2)$. In our model, the bigram features are extracted in the window context and then the corresponding bigram embeddings are concatenated with character embeddings in Layer 1 and fed into Layer 2. In Mansur et al. (2013), the bigram embeddings are pre-trained on unlabeled data with character embeddings, which significantly improves the model performance. Given the long time for pre-training bigram embeddings, we only pre-train the character embeddings and the bigram embeddings are initialized as the average of character embeddings of c_k and c_{k+1}. Further improvement could be obtained if the bigram embeddings are also pre-trained. Table 6 lists the segmentation performances of our model as well as previous state-of-the-art systems. When bigram features are added, the F-score of our model improves

[5] https://code.google.com/p/word2vec/

from 94.0% to 95.2% on PKU dataset and from 94.9% to 97.2% on MSRA dataset. It is a competitive result given that our model only use simple bigram features while other models use more complex features. For example, Sun et al. (2012) uses additional word-based features. Zhang et al. (2013) uses eight types of features such as Mutual Information and Accessor Variety and they extract dynamic statistical features from both an in-domain corpus and an out-of-domain corpus using co-training. Since feature engineering is not the main focus of this paper, we did not experiment with more features.

5 Related Work

Chinese word segmentation has been studied with considerable efforts in the NLP community. The most popular approach treats word segmentation as a sequence labeling problem which was first proposed in Xue (2003). Most previous systems address this task by using linear statistical models with carefully designed features such as bigram features, punctuation information (Li and Sun, 2009) and statistical information (Sun and Xu, 2011). Recently, researchers have tended to explore new approaches for word segmentation which circumvent the feature engineering by automatically learning features with neural network models (Mansur et al., 2013; Zheng et al., 2013). Our study is consistent with this line of research, however, our model explicitly models the interactions between tags and context characters and accordingly captures more semantic information.

Tensor-based transformation was also used in other neural network models for its ability to capture multiple interactions in data. For example, Socher et al. (2013b) exploited tensor-based function in the task of Sentiment Analysis to capture more semantic information from constituents. However, given the small size of their tensor matrix, they do not have the problem of high time cost and overfitting problem as we faced in modeling a sequence labeling task like Chinese word segmentation. That's why we propose to decrease computational cost and avoid overfitting with tensor factorization.

Various tensor factorization (decomposition) methods have been proposed recently for tensor-based dimension reduction (Cohen et al., 2013; Van de Cruys et al., 2013; Chang et al., 2013). For example, Chang et al. (2013) proposed the Multi-Relational Latent Semantic Analysis. Similar to LSA, a low rank approximation of the tensor is derived using a tensor decomposition approch. Similar ideas were also used for collaborative filtering (Salakhutdinov et al., 2007) and object recognition (Ranzato et al., 2010). Our tensor factorization is related to these work but uses a different tensor factorization approach. By introducing tensor factorization into the neural network model for sequence labeling tasks, the model training and inference are speeded up and overfitting is prevented.

6 Conclusion

In this paper, we propose a new model called Max-Margin Tensor Neural Network that explicitly models the interactions between tags and context characters. Moreover, we propose a tensor factorization approach that effectively improves the model efficiency and avoids the risk of overfitting. Experiments on the benchmark datasets show that our model achieve better results than previous neural network models and that our model can achieve a competitive result with minimal feature engineering. In the future, we plan to further extend our model and apply it to other structure prediction problems.

Acknowledgments

This work is supported by National Natural Science Foundation of China under Grant No. 61273318 and National Key Basic Research Program of China 2014CB340504.

References

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

Kai-Wei Chang, Wen-tau Yih, and Christopher Meek. 2013. Multi-relational latent semantic analysis. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1602–1612, Seattle, Washington, USA, October. Association for Computational Linguistics.

Aitao Chen, Yiping Zhou, Anne Zhang, and Gordon Sun. 2005. Unigram language model for chinese word segmentation. In *Proceedings of the 4th SIGHAN Workshop on Chinese Language Processing*, pages 138–141. Association for Computational Linguistics Jeju Island, Korea.

Shay B Cohen, Giorgio Satta, and Michael Collins. 2013. Approximate pcfg parsing using tensor decomposition. In *Proceedings of NAACL-HLT*, pages 487–496.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

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*, 999999:2121–2159.

Thomas Emerson. 2005. The second international chinese word segmentation bakeoff. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, volume 133.

David Graff and Ke Chen. 2005. Chinese gigaword. *LDC Catalog No.: LDC2003T09, ISBN*, 1:58563–230.

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.

Alex Krizhevsky, Geoffrey E Hinton, et al. 2010. Factored 3-way restricted boltzmann machines for modeling natural images. In *International Conference on Artificial Intelligence and Statistics*, pages 621–628.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Zhongguo Li and Maosong Sun. 2009. Punctuation as implicit annotations for chinese word segmentation. *Computational Linguistics*, 35(4):505–512.

Mairgup Mansur, Wenzhe Pei, and Baobao Chang. 2013. Feature-based neural language model and chinese word segmentation. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*.

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

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of NAACL-HLT*, pages 746–751.

Marc'Aurelio Ranzato, Alex Krizhevsky, and Geoffrey E Hinton. 2010. Factored 3-way restricted boltzmann machines for modeling natural images.

Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. 2007. (online) subgradient methods for structured prediction.

Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. 2007. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798. ACM.

Holger Schwenk, Anthony Rousseau, and Mohammed Attik. 2012. Large, pruned or continuous space language models on a gpu for statistical machine translation. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pages 11–19. Association for Computational Linguistics.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013a. Parsing with compositional vector grammars. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. EMNLP.

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.

Xu Sun, Yaozhong Zhang, Takuya Matsuzaki, Yoshimasa Tsuruoka, and Jun'ichi Tsujii. 2009. A discriminative latent variable chinese segmenter with hybrid word/character information. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 56–64. Association for Computational Linguistics.

Xu Sun, Houfeng Wang, and Wenjie Li. 2012. Fast online training with frequency-adaptive learning rates for chinese word segmentation and new word detection. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 253–262, Jeju Island, Korea, July. 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 22nd international conference on Machine learning*, pages 896–903. ACM.

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.

Tim Van de Cruys, Thierry Poibeau, and Anna Korhonen. 2013. A tensor-based factorization model of semantic compositionality. In *Proceedings of NAACL-HLT*, pages 1142–1151.

Mengqiu Wang and Christopher D Manning. 2013. Effect of non-linear deep architecture in sequence labeling. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*.

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

Yue Zhang and Stephen Clark. 2007. Chinese segmentation with a word-based perceptron algorithm. In *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, volume 45, page 840.

Ruiqiang Zhang, Genichiro Kikui, and Eiichiro Sumita. 2006. Subword-based tagging by conditional random fields for chinese word segmentation. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 193–196. Association for Computational Linguistics.

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*, pages 311–321, Seattle, Washington, USA, October. Association for Computational Linguistics.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for Chinese word segmentation and POS tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 647–657, Seattle, Washington, USA, October. Association for Computational Linguistics.