Deep learning for Chinese word segmentation and POS tagging

Conference Paper · October 2013

CITATIONS
94

READS
655

3 authors:

Xiaoqing Zheng
Fudan University
22 PUBLICATIONS **211** CITATIONS
SEE PROFILE

Hanyang Chen
University College Dublin
1 PUBLICATION **94** CITATIONS
SEE PROFILE

Tianyu Xu
Coventry University
1 PUBLICATION **94** CITATIONS
SEE PROFILE

Deep Learning for Chinese Word Segmentation and POS Tagging

Xiaoqing Zheng
Fudan University
220 Handan Road
Shanghai, 200433, China
zhengxq@fudan.edu.cn

Hanyang Chen
Fudan University
220 Handan Road
Shanghai, 200433, China
chenhy12345@gmail.com

Tianyu Xu
Fudan University
220 Handan Road
Shanghai, 200433, China
xty213@gmail.com

Abstract

This study explores the feasibility of performing Chinese word segmentation (CWS) and POS tagging by deep learning. We try to avoid task-specific feature engineering, and use deep layers of neural networks to discover relevant features to the tasks. We leverage large-scale unlabeled data to improve internal representation of Chinese characters, and use these improved representations to enhance supervised word segmentation and POS tagging models. Our networks achieved close to state-of-the-art performance with minimal computational cost. We also describe a perceptron-style algorithm for training the neural networks, as an alternative to maximum-likelihood method, to speed up the training process and make the learning algorithm easier to be implemented.

1 Introduction

Word segmentation has been a long-standing challenge for the Chinese NLP community. It has received steady attention over the past two decades. Previous studies show that joint solutions usually lead to the improvement in accuracy over pipelined systems by exploiting POS information to help word segmentation and avoiding error propagation. However, traditional joint approaches usually involve a great number of features, which arises four limitations. First, the size of the result models is too large for practical use due to the storage and computing constraints of certain real-world applications. Second, the number of parameters is so large that the trained model is apt to overfit on training corpus. Third, a longer training time is required. Last but not the least, the decoding by dynamic programming technique might be intractable since a large search space is faced by the decoder.

The choice of features, therefore, is a critical success factor for these systems. Most of the state-of-the-art systems address their tasks by applying linear statistical models to the features carefully optimized for the tasks. This approach is effective because researchers can incorporate a large body of linguistic knowledge into the models. However, the approach does not scale well when it is used to perform more complex joint tasks, for example, the task of joint word segmentation, POS tagging, parsing, and semantic role labeling. A challenge for such a joint model is the large combined search space, which makes engineering effective task-specific features and structured learning of parameters very hard. Instead, we use multilayer neural networks to discover the useful features from the input sentences.

There are two main contributions in this paper. (1) We describe a perceptron-style algorithm for training the neural networks, which not only speeds up the training of the networks with negligible loss in performance, but also can be implemented more easily; (2) We show that the tasks of Chinese word segmentation and POS tagging can be effectively performed by the deep learning. Our networks achieved close to state-of-the-art performance by transferring the unsupervised internal representations of Chinese characters into the supervised models.

Section 2 presents the general architecture of neural networks, and our perceptron-style training algorithm for tagging. Section 3 describes how to

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 647–657,
Seattle, Washington, USA, 18-21 October 2013.

leverage large unlabeled data to obtain more useful character embeddings, and reports the experimental results of our systems. Section 4 presents a brief overview of related work. The conclusions are given in section 5.

2 The Neural Network Architecture

Chinese word segmentation and part-of-speech tagging tasks can be formulated as assigning labels to characters of an input sentence. The performance of the traditional tagging approaches is heavily dependent on the choice of features, for example, conditional random fields (CRFs), often with a set of feature templates. For that reason, much of the effort in designing such systems goes into the feature engineering, which is important but labor-intensive, mainly based on human ingenuity and linguistic intuition.

In order to make learning algorithms less dependent on the feature engineering, we chose to use a variant of the neural network architecture first proposed by (Bengio et al., 2003) for probabilistic language model, and reintroduced later by (Collobert et al., 2011) for multiple NLP tasks. The network takes the input sentence and discovers multiple levels of feature extraction from the inputs, with higher levels representing more abstract aspects of the inputs. The architecture is shown in Figure 1. The first layer extracts features for each Chinese character. The next layer extracts features from a window of characters. The following layers are classical neural network layers. The output of the network is a graph over which tag inference is achieved with a Viterbi algorithm.

2.1 Mapping Characters into Feature Vectors

The characters are fed into the network as indices that are used by a lookup operation to transform characters into their feature vectors. We consider a fixed-sized character dictionary $\mathcal{D}$[1]. The vector representations are stored in a character embedding matrix $\mathcal{M} \in \mathbb{R}^{d \times |\mathcal{D}|}$, where d is the dimensionality of the vector space (a hyper-parameter to be chosen) and $|\mathcal{D}|$ is the size of the dictionary.

[1]Unless otherwise specified, the character dictionary is extracted from the training set. Unknown characters are mapped to a special symbol that is not used elsewhere.

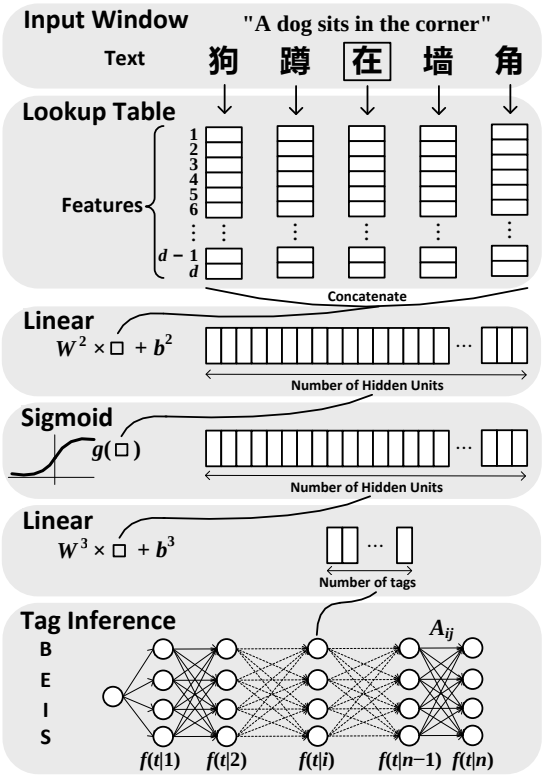

Figure 1: The neural network architecture.

Formally, assume we are given a Chinese sentence $c_{[1:n]}$ that is a sequence of n characters $c_i, 1 \leq i \leq n$. For each character $c_i \in \mathcal{D}$ that has an associated index k_i into the column of the embedding matrix, an d-dimensional feature vector representation is retrieved by the lookup table layer $\mathcal{Z}_{\mathcal{D}}(\cdot) \in \mathbb{R}^d$:

$$\mathcal{Z}_{\mathcal{D}}(c_i) = \mathcal{M} e_{k_i} \quad (1)$$

where we use a binary vector $e_{k_i} \in \mathbb{R}^{|\mathcal{D}| \times 1}$ which is zero in all positions except at the k_i-th index. The lookup operation can be seen as a simple projection layer. The feature vector of each character, starting from a random initialization, can be automatically trained by back propagation to be relevant to the task of interest.

In practice, it is common that one might want to provide other additional features that is thought to be helpful for the task. For example, for the name entity recognition task, one could provide a feature which says if a character is in a list of the common Chinese surnames or not. Another common practice is to introduce some statistics-based measures, such

as *boundary entropy* (Jin and Tanaka-Ishii, 2006) and *accessor variety* (Feng et al., 2004), which are commonly used in unsupervised CWS models. We associate a lookup table to each additional feature, and the character feature vector becomes the concatenation of the outputs of all these lookup tables.

2.2 Tag scoring

A neural network can be considered as a function $f_\theta(\cdot)$ with parameters θ. Any feed-forward neural network with L layers can be seen as a composition of functions $f_\theta^l(\cdot)$ defined for each layer l:

$$f_\theta(\cdot) = f_\theta^L(f_\theta^{L-1}(\dots f_\theta^1(\cdot)\dots)) \tag{2}$$

For each character in a sentence, a score is produced for every tag by applying several layers of the neural network over the feature vectors produced by the lookup table layer. We use a window approach to handle the sequences of variable sentence length. The window approach assumes that the tag of a character depends mainly on its neighboring characters. More precisely, given an input sentence $c_{[1:n]}$, we consider all successive windows of size w (a hyper-parameter), siding over the sentence, from character c_1 to c_n. At position c_i, the character feature window produced by the first lookup table layer can be written as:

$$f_\theta^1(c_i) = \begin{pmatrix} \mathcal{Z}_\mathcal{D}(c_{i-w/2}) \\ \vdots \\ \mathcal{Z}_\mathcal{D}(c_i) \\ \vdots \\ \mathcal{Z}_\mathcal{D}(c_{i+w/2}) \end{pmatrix} \tag{3}$$

The characters with indices exceeding the sentence boundaries are mapped to one of two special symbols, namely *"start"* and *"stop"* symbols.

The fixed-sized vector f_θ^1 is fed to two standard *Linear Layers* that successively perform affine transformations over f_θ^1, interleaved with some non-linearity function $g(\cdot)$, to extract highly non-linear features. Given a set of tags $\mathcal{T}$ for the task of interest, the network outputs a vector of size $|\mathcal{T}|$ for each character at position i, interpreted as a score for each tag in $\mathcal{T}$ and each character c_i in the sentence:

$$\begin{aligned} f_\theta(c_i) &= f_\theta^3(g(f_\theta^2(f_\theta^1(c_i)))) \\ &= W^3 g(W^2 f_\theta^1(c_i) + b^2) + b^3 \end{aligned} \tag{4}$$

where the matrices $W^2 \in \mathbb{R}^{H\times(wd)}, b^2 \in \mathbb{R}^H$, $W^3 \in \mathbb{R}^{|\mathcal{T}|\times H}$ and $b^3 \in \mathbb{R}^{|\mathcal{T}|}$ are the parameters to be trained. The hyper-parameter H is usually called the *number of hidden units*. As non-linear function, we chose a sigmoidal function[2]:

$$g(x) = 1/(1+e^{-x}) \tag{5}$$

2.3 Tag Inference

There are strong dependencies between character tags in a sentence for the tasks like word segmentation and POS tagging. The tags are organized in chunks, and it is impossible for some tags to follow other tags. We introduce a transition score A_{ij} for jumping from $i \in \mathcal{T}$ to $j \in \mathcal{T}$ tags in successive characters, and an initial scores A_{0i} for starting from the i-th tag for taking into account the sentence structure. We want the valid paths of tags to be encouraged, while discouraging all other paths.

Given an input sentence $c_{[1:n]}$, the network outputs the matrix of scores $f_\theta(c_{[1:n]})$. We use a notation $f_\theta(t|i)$ to indicate the score output by the network with parameters θ, for the sentence $c_{[1:n]}$ and for the t-th tag, at the i-th character. The score of a sentence $c_{[1:n]}$ along a path of tags $t_{[1:n]}$ is then given by the sum of transition and network scores:

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

Given a sentence $c_{[1:n]}$, we can find the best tag path $t^*_{[1:n]}$ by maximizing the sentence score:

$$t^*_{[1:n]} = \arg\max_{\forall t'_{[1:n]}} s(c_{[1:n]}, t'_{[1:n]}, \theta) \tag{7}$$

The Viterbi algorithm can be used for this inference. Now we are prepared to show how to train the parameters of the network in an end-to-end fashion.

2.4 Training

The training problem is to determine all the parameters of the network $\theta = (\mathcal{M}, W^2, b^2, W^3, b^3, A)$ from training data. The network generally is trained

[2]In our experiments, the sigmoidal function performs slightly better than the "hard" version of the hyperbolic tangent used by (Collobert, 2011).

by maximizing a likelihood over all the sentences in the training set $\mathcal{R}$ with respect to θ:

$$\theta \mapsto \sum_{\forall (c,t) \in \mathcal{R}} \log p(t|c,\theta) \tag{8}$$

where c represents a sentence and its associated features, and t denotes the corresponding tag sequence. We drop the subscript $[1:n]$ from now for notation simplification. The probability $p(\cdot)$ is calculated from the outputs of the neural network. We will present in the following section how to interpret neural network outputs as probabilities.

Maximizing the log-likelihood (8) with the gradient ascent algorithm[3] is achieved by iteratively selecting a example (c,t) and applying the following gradient update rule:

$$\theta \leftarrow \theta + \lambda \frac{\partial \log p(t|c,\theta)}{\partial \theta} \tag{9}$$

where λ is the learning rate (a hyper-parameter). The gradient in (9) can be computed by a classical back propagation: the differentiation chain rule is applied through the network, until the character embedding layer.

2.4.1 Sentence-Level Log-Likelihood

The score of a sentence (6) is interpreted as a conditional tag path probability by taking it to the exponential (making the score positive) and normalizing it over all possible tag paths (summing to 1 over all paths). Taking the log, the conditional probability of the true path t is given by[4]:

$$\log p(t|c,\theta) = s(c,t,\theta) - \log \sum_{\forall t'} \exp\{s(c,t',\theta)\} \tag{10}$$

[3]We did not use the stochastic gradient ascent algorithm (Bottou, 1991) to train the network as (Collobert et al., 2011). The gradient ascent algorithm was used instead for fairly comparing our algorithm with the sentence-level maximum-likelihood method (see Section 2.4.1). The gradient ascent algorithm requires a loop over all the examples to compute the gradient of the cost function, which will not cause a problem since all the training sets used in this article are finite.

[4]The cost functions are differentiable everywhere thanks to the differentiability of sigmoidal function chosen as non-linearity instead of a "hard" version of the hyperbolic tangent. For details about gradient computations, see Appendix A of (Collobert et al., 2011).

The number of terms in (10) grows exponentially with the length of the input sentence. Although one can compute it in linear time with the Viterbi algorithm, it is quite computationally expensive to compute the conditional probability of the true path, and its derivatives with respect to $f_\theta(t|i)$ and A_{ij}. The gradients with respect to the trainable parameters other than $f_\theta(t|i)$ and A_{ij} can all be computed using the derivatives with respect to $f_\theta(t|i)$ by applying the differentiation chain rule. We will see in the next section our training algorithm that has the advantage of being much cheaper to compute the gradients.

2.5 A New Training Method

The log-likelihood (10) can be seen as the difference between the forward score constrained over the valid path and the sum of the scores of all possible paths. While this training criterion is used, the neural networks are trained by maximizing the likelihood of training data. In fact, a CRF maximizes the same log-likelihood (Lafferty et al., 2001) by using a linear model in stead of a nonlinear neural network.

As an alternative to maximum-likelihood method, we propose the following training algorithm inspired by the work of (Collins, 2002). Given a training example (c,t), the network outputs the matrix of scores $f_\theta(c)$ under the current parameter settings. The highest scoring sequence of tags for the input sentence c then can be found using the Viterbi algorithm: this tagged sequence is denoted by t'. For every character c_i where $t_i \neq t'_i$, we simply set

$$\frac{\partial L_\theta(t,t'|c)}{\partial f_\theta(t_i|i)} {+}{+}, \ \frac{\partial L_\theta(t,t'|c)}{\partial f_\theta(t'_i|i)} {-}{-} \tag{11}$$

and for every transition where $t_{i-1} \neq t'_{i-1}$ or $t_i \neq t'_i$, we set

$$\frac{\partial L_\theta(t,t'|c)}{\partial A_{t_{i-1}t_i}} {+}{+}, \ \frac{\partial L_\theta(t,t'|c)}{\partial A_{t'_{i-1}t'_i}} {-}{-} \tag{12}$$

where "$+\,+$" (which increases a value by one) and "$-\,-$" (which decreases a value by one) are two unary operators, and $L_\theta(t,t'|c)$ is a new function which we now want to maximize over all the training pairs (c,t). The function $L_\theta(t,t'|c)$ can be viewed as the difference between the score of the correct path and that of the incorrect one (which is the highest scoring sequence produced by the network under the current parameters θ).

As an example, say the correct tag sequence of the sentence 狗蹲在墙角 'A dog sits in the corner' is

狗/S 蹲/S 在/S 墙/B 角/E

and under the current parameter settings the highest scoring tag sequence is

狗/S 蹲/B 在/E 墙/B 角/E

Then the derivatives with respect to $f_\theta(S|蹲)$, and $f_\theta(S|在)$ will be set to 1, that with respect to A_{SB}, A_{BE}, A_{EB}, $f_\theta(B|蹲)$, and $f_\theta(E|在)$ to -1, and that with respect to A_{SS} to 2 respectively[5]. Intuitively these assignments have the effect of updating the parameter values in a way that increases the score of the correct tag sequence and decreases the score of the incorrect one output by the network with the current parameter settings. If the tag sequence produced by the network is correct, no changes are made to the values of parameters.

Inputs:
$\mathcal{R}$: a training set.
N: a specified maximum number of iterations.
E: a desired tagging precision.

Initialization: set the initial parameters of the network with small random values.

Output: the trained parameters $\tilde{\theta}$

Algorithm:

do
  for each example $(c, t) \in \mathcal{R}$
    get the matrix $f_\theta(c)$ by the neural network under the current parameters θ
    find the highest scoring sequence of tags t' for c with $f_\theta(c)$ and A_{ij} by using the Viterbi algorithm
    if $(t \neq t')$
      compute the gradients with respect to $f_\theta(c)$ and A_{ij} as (11) and (12)
      compute the gradients with respect to the weights from output layer to character embedding layer
      update the parameters of the network by (9)
until the desired precision E achieved or maximum number of iterations N reached
return $\tilde{\theta}$

Figure 2: The training algorithm for tagging.

We propose the training algorithm in Figure 2. Note that the perceptron algorithm of (Collins, 2002) was designed for discriminatively training an HMM-style tagger, while our algorithm is used to calculate the "direction" in which the parameters are updated (i.e. the gradient of the function we want to maximize). Due to space limitations, we do not give convergence theorems justifying the training algorithm in this paper. Intuitively it can be achieved by combining the theorems of convergence for the perceptron applied to tagging problem from (Collins, 2002) with the convergence results of backpropagation algorithm from (Rumelhart et al., 1986).

3 Experiments

We conducted three sets of experiments. The goal of the first one is to test several variants for each training algorithm on the development set, to gain some understanding of how the choice of hyperparameters impacts upon the performance. We applied the network both with the sentence-level log-likelihood (SLL) and our perceptron-style training algorithm (PSA) to the two Chinese NLP problems: word segmentation, and joint CWS and POS tagging. We ran this set of experiments on the part of Chinese Treebank 4 (CTB-4)[6]. Ninety percent of the sentences (1529) were randomly chosen for training and the rest (168) were used as development set.

The second set of experiments was run on the Chinese Treebank (CTB) data sets from Bakeoff-3 (Levow, 2006), which contains a training and a test corpus for supervised word segmentation and POS tagging tasks. The results were obtained without using any extra knowledge (i.e. the closed test), and are comparable with other models in the literature.

In the third experiment, we study to see how well large unlabeled texts can be used to enhance the supervised learning. Following (Collobert et al., 2011), we first use large unlabeled data set to obtain character embeddings carrying more syntactic and semantic information, and then use these improved embeddings to initialize the character lookup tables of the networks instead of previous random values. Our corpus is the Sina news[7] that contains about 325MB data.

[5]The derivatives with respect to A_{SB} will be set to 0, because it is increased first and decreased afterwards.

[6]The data set was sections 1–43, 144–169, and 900–931 of the treebank, containing 78,023 characters, 45,135 words and 1,697 sentences. These files are double-annotated and can be regarded as golden standard files.

[7]Available at http://www.sina.com.cn/

We implemented two versions of the network: one for the sentence-level log-likelihood and one for our perceptron-style training algorithm. Both are written in Java language. All experiments were run on a computer equipped with an Intel Core i3 processor working at 2.13GHz, with 2GB RAM, running Linux and Java Development Kit 1.6. The standard F-score was used to evaluate the performance of both word segmentation and joint word segmentation and POS tagging tasks. F-score is the harmonic mean of precision p and recall r, which is defined as $2pr/(p+r)$.

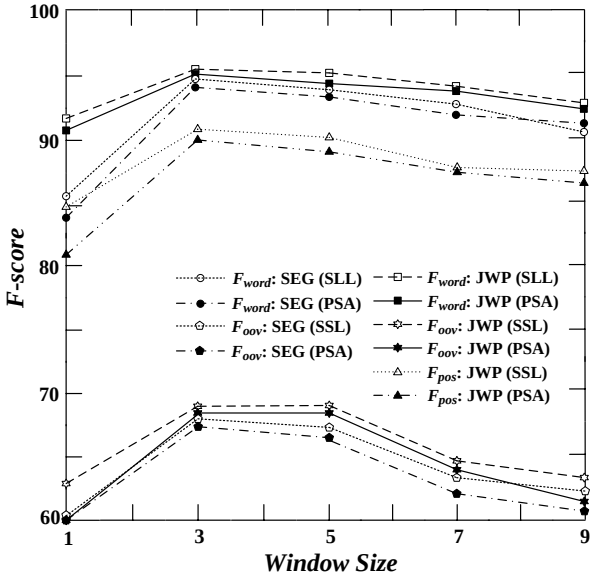

Figure 3: Average F-score versus window size.

3.1 Tagging Schemes

The network will output the scores for all the possible tags for the task of interest. For word segmentation, each character will be assigned one of four possible boundary tags: "*B*" for a character located at the beginning of a word, "*I*" for that inside of a word, "*E*" for that at the end of a word, and "*S*" for a character that is a word by itself.

Following Ng and Lou (2004) we perform joint word segmentation and POS tagging task in a labeling fashion by expanding boundary labels to include POS tags. For instance, we describe verb phases using four different tags. Tag "*S_VP*" is used to mark a verb phase containing a single character. Other tags "*B_VP*", "*I_VP*", and "*E_VP*" are used to mark the first, in-between and last characters of the verb phrase. In fact, we used the "*IOBES*" tagging scheme, and tag "*O*" is not applicable to Chinese word segmentation and POS tagging tasks.

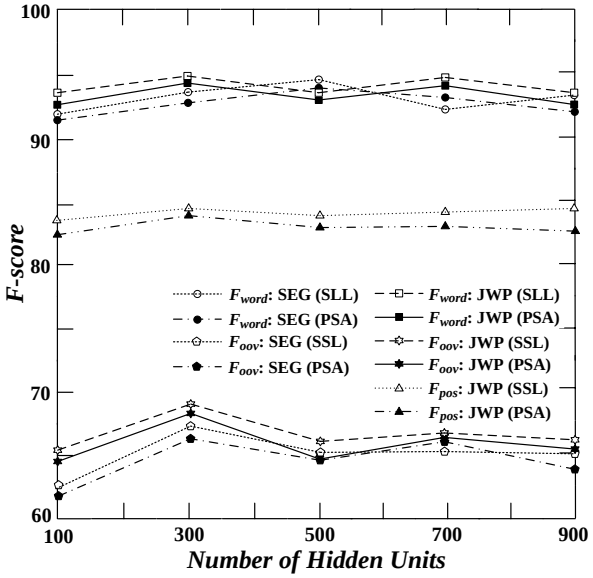

Figure 4: Average F-score versus number of hidden units.

3.2 The Choice of Hyper-parameters

We tuned the hyper-parameters by trying only a few different networks. We report in Figure 3 the F-scores on the development set versus window size for word segmentation (SEG) and joint word segmentation and POS tagging (JWP) tasks with the sentence-level log-likelihood (SLL) and our perceptron-style training algorithm (PSA), and report in Figure 4 the F-scores on the same data set versus number of hidden units. The average F-scores were obtained over 5 runs with different random initialization for each setting of the network. The F-scores of the word segmentation, out-of-vocabulary, and POS tagging are denoted by F_{word}, F_{oov} and F_{pos} respectively.

Generally, the number of hidden units has a limited impact on the performance if it is large enough, which is consistent with the findings of (Collobert et al., 2011) for English. It can be seen from Figure 3 that the performance drops smoothly when the window size is larger than 3. In particularly, the F-score of out-of-vocabulary identification decreases relatively fast beyond window size 5, which shows

that the size of window (and the number of parameters) is too large that the trained network has overfitted on training data. An explanation for this result is that most Chinese words are less than 3 characters, and the neighboring characters outside of the window (size 5) become "noise" when we perform word segmentation.

The hyper-parameters of the network used in all the following experiments are shown in Table 1. Although the top performance was obtained by the network with window size 3, we chose the architecture with window size 5 because a larger training corpus will be used in the following experiments, and the sparseness problem would be alleviated. Furthermore, in order to obtain character embeddings by using large unlabeled data, we prefer to "observe" a character within a slightly larger window to better discover its syntactic and semantic information.

Hyper-parameter	Value
Window size	5
Number of hidden units	300
Character feature dimension	50
Learning rate	0.02

Table 1: Hyper-parameters of the network.

We report in Table 2 the F-score of the first five iterations on the development set for word segmentation with SLL and PSA. The data in the fourth and fifth rows of the table shows the convergence of PSA. The difference of the F-scores between the networks with SLL and PSA can be negligible after the number of iteration is greater than 5. In our implementation, for each iteration the training time is reduced at least 10% by using PSA, compared with SLL. The training time can be reduced further for more complex tasks like POS tagging and semantic role labeling in which a larger tag set is used.

Iteration		1	2	3	4	5
SSL	F_{word}	49.89	69.56	88.91	90.19	91.24
	F_{oov}	15.92	27.54	54.37	55.89	59.74
	Time (s)	209	398	586	737	886
PSA	F_{word}	49.04	68.54	87.79	89.07	91.19
	F_{oov}	13.61	25.79	52.30	55.15	60.49
	Time (s)	184	343	497	610	754

Table 2: Word segmentation results with SLL and PSA for the first five iterations.

Many exponential sums ($\sum_i \exp(x_i)$) are required for training the networks with SLL, and in most cases the values of exponential sums will exceed the range of double-precision floating-point arithmetic defined in popular programming languages. These sums need to be estimated by analytic number theory. In comparison, a lot of the computation-intensive exponential sums are avoided in our training algorithm, which not only speed up the training of the networks but also make it easier to be implemented.

3.3 Closed Test on the SIGHAN Bakeoff

We trained the networks with PSA on the Chinese Treebank (CTB) data set from Bakeoff-3 for both SEG and JWP tasks. The results are reported in Table 3. The hyper-parameters of our networks are reported in Table 1. Although results show that our networks with PSA are behind the state-of-the-art systems, the networks perform comparatively well, considering we did not use any extra information. Many other systems used some extra heuristics or resources to improve their performance. For example, a key parameter in the system of (Wang et al., 2006) was optimized in advance by using an external segmented corpus, and a manually prepared list of characters as well as their types was used in (Zhao et al., 2006; Zhu et al., 2006; Kruengkrai et al., 2009).

It is worth noting that the comparison for joint word segmentation and POS tagging task is indirect because the different versions of CTB were used. We reported the results on CTB-3 from SIGHAN Bakeoff-3, while both (Jiang et al., 2008) and (Kruengkrai et al., 2009) used CTB-5. Both (Ng and Lou, 2004) and (Zhang and Clark, 2008) evenly partitioned the sentences in CTB3 into ten groups, and used nine groups for training and the rest for testing.

Following (Bengio et al., 2003; Collobert et al., 2011), we want semantically and syntactically similar characters to be close in the embedding space. If we knew that 狗 'dog' and 猫 'cat' were similar semantically, and similarly for 蹲 'sit' and 躺 'lie', we could generalize from 狗蹲在墙角 'A dog sits in the corner' to 猫蹲在墙角 'A cat sits in the corner', and to 猫躺在墙角 'A cat lies in the corner' in the same way. We describe the way to obtain these character embeddings by using large unlabeled data in the next section.

Approach		F_{word}	R_{oov}	F_{pos}
SEG	(Zhao et al., 2006)	93.30	70.70	–
	(Wang et al., 2006)	93.00	68.30	–
	(Zhu et al., 2006)	92.70	63.40	–
	(Zhang et al., 2006)	92.60	61.70	–
	(Feng et al., 2006)	91.70	68.00	–
	PSA	92.59	64.24	–
	PSA + LM	94.57	70.12	–
JWP	(Ng and Lou, 2004)	95.20	–	–
	(Zhang and Clark, 2008)	95.90	–	91.34
	(Jiang et al., 2008)	97.30	–	92.50
	(Kruengkrai et al., 2009)	96.11	–	90.85
	PSA	93.83	68.21	90.79
	PSA + LM	95.23	72.38	91.82

Table 3: Comparison of the F-scores on the Penn Chinese Treebank

3.4 Combined Approach

We used the corpus of Sina news to obtain character embeddings carrying more semantic and syntactic information by training a language model that evaluates the acceptability of a piece of text. This language model is again the neural network, and we also use PSA to train the language model. Following (Collobert et al., 2011), we minimize the following criterion with respect to the parameters θ:

$$\theta \mapsto \sum_{\forall h \in \mathcal{H}} \sum_{\forall c' \in \mathcal{D}} \max\{0, 1 - f_\theta(c|h) + f_\theta(c'|h)\} \tag{13}$$

where the score $f_\theta(c|h)$ is the output of the network with parameters θ for a character c at the center of a window h, $\mathcal{D}$ is the dictionary of characters, $\mathcal{H}$ is the set of all possible text windows (i.e. character sequences) from the training data, and $c'|h$ denotes the window obtained by replacing the central character of the window h by the character c'.

We used a dictionary consisting of the characters extracted from all the data sets in Bakeoff-3, which contains about eight thousand characters. The total unsupervised training time was about two weeks. Our combined approach works as initializing the lookup tables of the supervised networks with the character embeddings obtained by unsupervised learning, and then performing supervised training on CTB-3. The lookup tables will not be modified at the supervised training stage.

We reported the results in Table 3, in which our combined approach is indicated by "PSA + LM". It can be seen from Table 3 that this approach results in a performance boost for both SEG and JWP tasks. The POS tagging F-score of our approach was comparable to but still less than the model of (Jiang et al., 2008). They achieved the best score by first separately training multiple word-, character-, and POS n-gram based models, and then integrating them by cascading method. In comparison, our networks achieve the performance by automatically discovering useful features by itself and avoiding the task-specific engineering.

Table 4 compares the decoding speeds on the test data from CTB-3 for our system and for two CRFs-based word segmentation systems. Regardless of the differences in implementation, the neural networks clearly run considerably faster than the systems based on the CRFs model. They also require much more memory than our neural networks.

System	Number of parameters	Time (s)
(Tsai et al., 2006)	3.1×10^6	1669
(Zhao et al., 2006)	3.8×10^6	2382
Neural network	4.7×10^5	138

Table 4: Comparison of computational cost.

4 Related Work

Word segmentation has been pursued with considerable efforts in the Chinese NLP community, and statistical approaches are clearly dominant in the last decade. A popular statistical approach is the character-based tagging solution that treats word segmentation as a sequence tagging problem, assigning labels to the characters indicating whether a character locates at the beginning of, inside, or at the end of a word. The character-based tagging solution was first proposed in (Xue, 2003). This work caused quite a number of character position tagging based CWS studies because known and unknown words can be treated in the same way. Peng, Feng and McCallum (2004) first introduced a linear-chain CRFs model to the character tagging based word segmentation. Zhang and Clark (2007) proposed a word-based CWS approach using a discriminative perceptron learning algorithm, which allows word-level information to be added as features.

Recent years have seen a rise of joint word segmentation and POS tagging approach that improves

the accuracies of both tasks and does not suffer from the error propagation. Ng and Lou (2004) perform such joint task in a labeling fashion by expanding boundary labels to include POS tags. Zhang and Clark (2008) proposed a linear model for the same joint task, which overcomed the disadvantage of (Ng and Lou, 2004), in which it was unable to incorporate "whole word + POS tag" features. Sun (2011) described a sub-word model using stacked learning technique for the joint task, which explored the complementary strength of different character- and word-based segmenters with different views.

The majority of the state-of-the-art systems address their tasks by applying linear statistical models to ad-hoc features. The researchers first chose task-specific features which are then fed to a classification algorithm. The selected features may vary greatly because they are usually chosen in a empirical process, mainly based first on linguistic intuition, and then trial and error. It seems reasonable to assume that the number and effectiveness of features constitutes a major factor in the performance of the various systems, and might even more important than the particular statistical models they used. In comparison, we try to avoid task-specific feature engineering, and use the neural network to learn several layers of feature extraction from the inputs. To the best of our knowledge, this study is among the first ones to perform Chinese word segmentation and POS tagging by deep learning.

It was reported that supervised and unsupervised approaches can be integrated to improve on the overall performance of word segmentation by combining the strengths of both. Zhao and Kit (2011) explored the feasibility of enhancing supervised segmentation by informing the supervised learner of goodness scores obtained from large unlabeled corpus. Sun and Xu (2011) investigated how to improve on the accuracy of supervised word segmentation by leveraging the statistics-based features derived from large unlabeled in-domain corpus and the document to be segmented. The basic idea of these integration solutions is to incorporate a set of statistics-based measures into a CRFs model after these measures are derived from unlabeled data and discretized into feature values. In comparison, we use large unlabeled data to obtain the character embeddings with more syntactic and semantic information.

Several works have investigated how to use deep learning for NLP applications (Bengio et al., 2003; Collobert et al., 2011; Collobert, 2011; Socher et al., 2011). In most cases, words are fed to the neural networks as inputs, and the lookup tables map each word to a vector representation. Our network is different in that the inputs to the network are characters, more raw units than words. In many Asian languages, such as Chinese and Japanese, they are written without using whitespace to delimit words. For these languages, the character becomes a more natural form of input. Furthermore, a perceptron-style algorithm for tagging is proposed for training the networks.

5 Conclusion

We have described a perceptron-style algorithm for training the neural networks, which is much easier to be implemented, and has speed advantage over the maximum-likelihood scheme, while the loss in performance is negligible. The neural networks trained with PSA have been applied to Chinese word segmentation and POS tagging tasks, and the networks achieved close to state-of-the-art performance by using the character representations learned from large unlabeled corpus.

Although we focus on the question of how far we can go for Chinese word segmentation and POS tagging without using the extra task-specific features in this study, there are at least three ways to further improve the performance of the networks, which are worthy to be explored in the future: (1) introduce specific linguistic features (e.g. gazetteer features) that are helpful for the tasks; (2) incorporate some common techniques, such as cascading, voting, and ensemble; and (3) use the special network architecture tailored for the tasks of interest.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments. The work was supported by a grant from the National Natural Science Foundation of China (No. 60903078), a grant from Shanghai Leading Academic Discipline Project (No. B114), a grant from Shanghai Municipal Natural Science Foundation (No. 13ZR1403800), and a grant from FDUROP.

References

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

Léon Bottou. 1991. Stochastic gradient learning in neural networks. *In Proceedings of the Neuro-Nîmes*.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. *In Proceedings of the International Conference on Empirical Methods in Natural Language Processing (EMNLP'02)*.

Ronan Collobert. 2011. Deep learning for efficient discriminative parsing. *In Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS'11)*.

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

Haodi Feng, Kang Chen, Xiaotie Deng, and Weimin Zheng. 2004. Accessor variety criteria for Chinese word extraction. *Computational Linguistics*, 30(1): 75–93.

Yuanyong Feng, Sun Le, and Yuanhua Lv. 2006. Chinese word segmentation and name entity recognition based on conditional random fields models. *In Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing (SIGHAN'06)*.

Wenbin Jiang, Liang Huang, Qun Liu, Yajuan Lu. 2008. A cascaded linear model for joint Chinese word segmentation and part-of-speech tagging. *In Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL'08)*.

Zhihui Jin, and Kumiko Tanaka-Ishii. 2006. Unsupervised segmentation of Chinese text by use of branching entropy. *In Proceedings of the International Conference on Computational Linguistics and the Annual Meeting of the Association for Computational Linguistics (COLING/ACL'06)*.

Canasai Kruengkrai, Kiyotaka Uchimoto, Jun'ichi Kazama, Yiou Wang, Kentaro Torisawa, and Hitoshi Isahara. 2009. An error-driven word-character hybrid model for joint Chinese word segmentation and pos tagging. *In Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL'09)*.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *In Proceedings of the International Conference on Machine learning (ICML'01)*.

Gina A. Levow. 2006. The third international Chinese language processing bakeoff: Word segmentation and named entity recognition. *In Proceedings of the SIGHAN Workshop on Chinese Language Processing (SIGHAN'06)*.

Hwee T. Ng and Jin K. Lou. 2004. Chinese part-of-speech tagging: one-at-atime or all-at-once? word-based or character-based? *In Proceedings of the International Conference on Empirical Methods in Natural Language Processing (EMNLP'04)*.

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. *In Proceedings of the 20th International Conference on Computational Linguistics (COLING'04)*.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning internal representations by backpropagating errors. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, 1: 318–362. MIT Press.

Richard Socher, Cliff C-Y. Lin, Andrew Y. Ng, and Christopher D. Manning 2011. Parsing Natural Scenes and Natural Language with Recursive Neural Networks. *In Proceedings of the International Conference on Machine learning (ICML'11)*.

Weiwei Sun. 2011. A stacked sub-word model for joint Chinese word segmentation and part-of-speech tagging. *In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL'11)*.

Weiwei Sun and Jia Xu. 2011. Enhancing Chinese word segmentation using unlabeled data. *In Proceedings of the International Conference on Empirical Methods in Natural Language Processing (EMNLP'11)*.

Richard T.-H. Tsai, Hsieh C. Hung, Chenglung Sung, Hongjie Dai, and Wenlian Hsu. 2006. On closed task of Chinese word segmentation: An improved CRF model coupled with character clustering and automatically generated template matching. *In Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing (SIGHAN'06)*.

Xinhao Wang, Xiaojun Lin, Dianhai Yu, Hao Tian, and Xihong Wu. 2006. Chinese word segmentation with maximum entropy and n-gram language model. *In Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing (SIGHAN'06)*.

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

Min Zhang, GuoDong Zhou, LingPeng Yang, and DongHong Ji. 2006. Chinese word segmentation and named entity recognition based on a context-dependent mutual information independence Model. *In Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing (SIGHAN'06)*.

Yue Zhang and Stephen Clark. 2007. Chinese segmentation with a word-base perceptron algorithm. *In Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL'07)*.

Yue Zhang and Stephen Clark. 2008. Joint word segmentation and pos tagging using a single perceptron. *In Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL'08)*.

Hai Zhao, Chang N. Huang, and Mu Li. 2006. An improved Chinese word segmentation system with conditional random field. *In Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing (SIGHAN'06)*.

Hai Zhao and Chunyu Kit. 2011. Integrating unsupervised and supervised word segmentation: The role of goodness measures. *Information Sciences*, 181(1): 163–183.

Muhua Zhu, Yilin Wang, Zhenxing Wang, Huizhen Wang, and Jingbo Zhu. 2006. Designing special post-processing rules for SVM-based Chinese word segmentation. *In Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing (SIGHAN'06)*.