Character-Based Parsing with Convolutional Neural Network

Xiaoqing Zheng, Haoyuan Peng, Yi Chen, Pengjing Zhang, Wenqiang Zhang
School of Computer Science, Fudan University, Shanghai, China
Shanghai Key Laboratory of Intelligent Information Processing
{zhengxq, penghy11, yichen11, pengjingzhang13, wqzhang}@fudan.edu.cn

Abstract

We describe a novel convolutional neural network architecture with k-max pooling layer that is able to successfully recover the structure of Chinese sentences. This network can capture active features for unseen segments of a sentence to measure how likely the segments are merged to be the constituents. Given an input sentence, after all the scores of possible segments are computed, an efficient dynamic programming parsing algorithm is used to find the globally optimal parse tree. A similar network is then applied to predict syntactic categories for every node in the parse tree. Our networks archived competitive performance to existing benchmark parsers on the CTB-5 dataset without any task-specific feature engineering.

1 Introduction

Syntactic parsing of natural langauge sentences is one of the most fundamental NLP tasks because of its importance in mediating between linguistic meaning and expression. Chinese parsing has also received steady attention since the release of Penn Chinese Treebank [Xue *et al.*, 2005]. Most of the previous work has focused on finding relevant features for the model component, and on finding effective statistical techniques for parameter estimation. Although such performance improvements can be useful in practice, there is a great temptation to optimize the system's performance for a specific benchmark. Furthermore, such systems, especially for those that use joint solution, usually involve a great number of features, which makes engineering effective task-specific features and structured learning of parameters very hard.

We address the Chinese parsing task by avoiding feature engineering, which falls in line with the recent efforts to surpass the state-of-the-art for many NLP tasks by "deep" learning. We introduce a convolutional neural network with k-max pooling layer, called KMCNN, which can be used to produce a score for each segment in an input sentence. Each produced score measures how likely the segment is to become a constituent in the parse tree. The k-max pooling operation is applied on the output of the convolutional layer, and pools the k most active features in each dimension to capture syntactic and semantic information of a sentence segment, which preserves the relative positions of the most relevant features, and thus it is sensitive to the order of the words in the input sentence. Experimental results show that the KMCNN can successfully recover the structure of Chinese sentences.

The KMCNN is different from an adapted *Graph Transformer Network* with single local max pooling operation (GTN) introduced by [Collobert, 2011] for discriminative parsing. The advantage of the GTN is that it does not depend on external language-specific features such as those derived from the constituency or dependency parse trees. However, its single max pooling operation cannot discriminate whether a relevant feature occurs just one or several times, and also neglects the relative order in which the features occur. The order of the words does matter to the meaning or the structure of the sentences, which is very useful for parsing or semantic analysis. Another limitation of the convolutional neural network with single max pooling operator is that if most or all of the highest values occur in a segment, the outputs of the max pooling layer for larger segments containing this segment are very close or equivalent to each other, and those segments cannot be distinguished in such cases.

Kalchbrenner et al [2014] proposed a more sophisticated network, called *Dynamic Convolutional Neural Network* (DCNN), in which multiple convolutional layers interleaved with k-max pooling layers are stacked to derive higher level features. In the DCNN, the k is a variable that depends on the length of an input sentence and the depth of the network. The DCNN is effective in modeling whole sentences for many sentence-level tasks including sentiment prediction, and question type classification, but its benefit may not be applicable to the case of sentence segment, much smaller units than sentence. The KMCNN is also different from the DCNN in how the convolution is computed. The DCNN used one-dimensional convolution that is to take the dot product of the filter vector with each n-gram in the sentence at the same dimension to obtain another feature vector, whereas the convolution in the KMCNN is calculated by performing affine transformations over each feature window (concatenation of multiple consecutive feature vectors) to extract local features.

There are two major contributions in this paper. (1) We describe a convolutional neural network architecture with k-max pooling layer (KMCNN), which is able to capture the most active features of sentence segments of varying length, and can be used to recover the structure of sentences in com-

bination with a dynamic programming decoder; (2) We applied KMCNN to parse Chinese sentences, and achieved the state-of-art performance in constituent parsing by transferring intermediate representations learned on large unlabeled data without task-specific feature engineering.

2 Parse Tree Representation

Parsing can be viewed as a "chunking" process which recursively finds chunks of phrases or constituents in a sentence. Words are first merged into phrases, and then the phrases are further merged into larger constituents in a syntactically and semantically meaningful order until the full parse tree is deduced. The search space is proportional to the number of possible parse trees. Even in grammars without unary chains or empty elements, the number of parses is generally exponential in the length of the sentence. For reducing the search space, we focus on binary trees, which also allow us to use efficient dynamic programming algorithms for decoding.

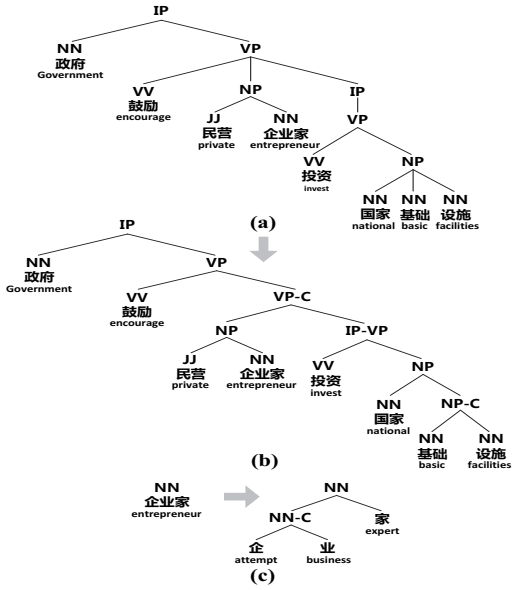

Figure 1: Parse tree representations. (a) Parse tree as in the CTB. (b) Transformed binary tree after concatenating nodes spanning the same constituents. (c) Expanding the word node to its character-level tree through morphological analysis.

A parse tree in the Penn Chinese Treebank (CTB) is shown in Figure 1 (a). The root spans all words of the sentence, and it is recursively decomposed into smaller constituents with labels like VP (verb phrase), NP (noun phrase), etc. We transformed the parse trees into the equivalent binary forms like Figure 1 (b) for training. The following three standard preprocessing steps have been applied on the original trees:

- Multi-branching trees are converted into arbitrary binary-branching nodes by inserting temporary nodes. The labels of temporary nodes are generated by appending "-C" (component) to the labels of their parent nodes.
- Tree nodes spanning the same constituents for several consecutive levels are replaced by one node. The label of this new node is the concatenation of replaced node labels, such as "IP-VP" in Figure 1 (b).
- Functional labels as well as traces are removed.

This transformation added 61 extra tags to 22 already had in the CTB. The first step breaks multi-branching nodes down into binary-branching nodes by inserting temporary nodes. Temporary nodes are collapsed and removed when we transform a binary tree back into the multi-branching tree for testing. The inverse operation is also performed on the nodes having concatenated labels at test time.

Previous studies show that Chinese syntactic parsing by using character-level information and joint solution (i.e., performing word segmentation, POS tagging, and parsing at the same time) usually leads to the improvement in accuracy over pipelined systems because the error propagation is avoided and the various information normally used in the different steps of pipelined systems can be integrated [Qian and Liu, 2012; Wang *et al.*, 2013; Zhang *et al.*, 2013]. Characters, in fact, play an important role in the character-based languages, such as Chinese and Japanese. They act as basic morphological, syntactic and semantic units in a sentence.

The internal structures of words can be recovered through morphological analysis. For example, 企业家'entrepreneur' is a word derived by adding the suffix 家'expert noun plural suffix' to the base 企业'enterprize', as shown in Figure 1 (c). Like [Zhang *et al.*, 2013], we extended the notation of phrase-structure trees, and expanded word nodes into their internal structures. Using these extended annotations, the joint word segmentation, POS tagging, and parsing can be performed by starting with the character-level in a unified manner.

3 The Network Architecture

We choose to use a variant of convolutional neural network with k-max pooling layer to find the structure of sentences. The network architecture is shown in Figure 2. The k-max pooling operations are applied in the network after the convolutional layers, which are used to pool the k most active features at low levels. This network preserves the relative positions of the most relevant features, and thus it is sensitive to the order of the words in input sentences. Convolutional layers are often stacked, interleaved with a non-linearity function, to extract higher level features (only two convolutional layers are drawn in Figure 2 for clarity). The topmost k-max pooling layer also guarantees that the input to the next layer of the network is independent of the length of an input sentence, in order to apply the same subsequent layers.

The input to the KMCNN is a sentence segment, and the network induces a score for the segment which measures how well the characters in the segment can be combined into a constituent or a phrase. After all the scores of possible segments are computed (the scores for different segments can be calculated in a parallel fashion by sharing the intermediate values of their overlapping parts), a dynamic programming parsing algorithm is used to find the globally optimal tree. For every node in the optimal parse tree, the network predicts syntactic labels by adding to the network a simple softmax layer (after removing the last scoring layer).

3.1 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 $\mathcal{D}$. 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.

Formally, assume we are given a 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, a d-dimensional feature vector representation is retrieved by the lookup table layer $\mathcal{Z}_{\mathcal{D}}(\cdot) \in \mathbb{R}^d$:

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

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 can be initialized randomly or pre-trained on unlabeled corpora. The character representations can be tuned by back propagation to be relevant to the task.

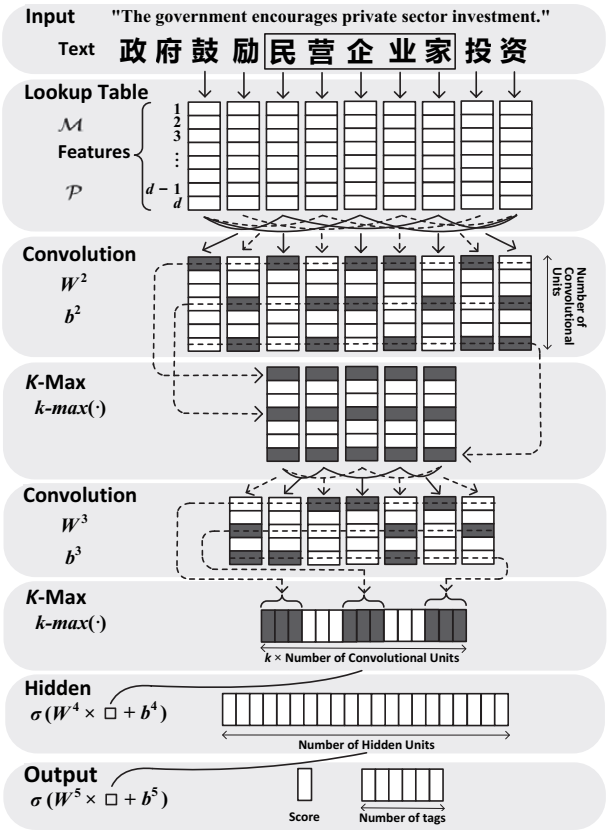

Figure 2: The neural network architecture.

3.2 Convolutional Layer

The lookup table layer extracts features for each single character, but the features of a character in context will be influenced by its surrounding characters. We assume that the features of a particular character depend mainly on its neighboring characters, and extract these features from a fixed size window w (a hyper-parameter). More precisely, given an input sentence $c_{[1:n]}$, the character feature window produced by the first lookup table layer at position c_i can be written as:

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

where f_θ^{win} is a function with trainable parameters θ. The idea behind the convolution is to perform affine transformations over each character window to extract local features:

$$f_\theta^{con}(c_i) = (W f_\theta^{win}(c_i) + b) \tag{3}$$

where the matrices $W \in \mathbb{R}^{V \times (wd)}$ and $b \in \mathbb{R}^V$ are the parameters to be optimized by training. A hyper-parameter V is called as the *number of convolutional units*. The trained weights in W and b can be viewed as a linguistic feature detector that learns to recognize a specific class of n-grams.

A wide type of convolution is used as [Kalchbrenner *et al.*, 2014], and yields a matrix $f_\theta^{con}(c_{[i:j]}) \in \mathbb{R}^{V \times (j-i+w)}$ for the segment $c_{[i:j]}$ from character c_i to c_j. Out-of-range input values are taken to be zeros. The narrow convolution gives largely equal importance to the features for every character in a segment, with the exception of characters at the margins which are considered fewer times in the computation of the convolution. In our case, the characters at the margins are more important than others since they define the boundaries of segments, and cannot be underestimated. The wide convolution ensures all weights in the filter reach the entire segment, including the characters at the margins (see Figure 2). This is particularly significant when the window size w is set to a relatively large value.

3.3 *k*-Max Pooling Layer

We applied a k-max pooling operation to the output of the convolutional layer, which is a generalization of max pooling over time dimension used in [Collobert and Weston, 2008; Collobert *et al.*, 2011]. Given a number k and a sequence $\mathcal{Q} \in \mathbb{R}^p (k \leq p)$, k-max pooling selects the subsequence $\mathcal{Q}_{max}^k$ of k-highest values in $\mathcal{Q}$. The selected values of $\mathcal{Q}_{max}^k$ preserve their relative order in $\mathcal{Q}$. Given a output matrix f_θ^{con}, the k-max pooling layer yields another matrix:

$$f_\theta^{max}(f_\theta^{con}) = \begin{pmatrix} k\text{-max}([f_\theta^{con}]_{1,1} \cdots [f_\theta^{con}]_{1,j-i+w}) \\ \vdots \quad \vdots \quad \vdots \\ k\text{-max}([f_\theta^{con}]_{V,1} \cdots [f_\theta^{con}]_{V,j-i+w}) \end{pmatrix} \tag{4}$$

where $[f_\theta^{con}]_{i,j}$ is the element in the i-th row and j-th column of matrix f_θ^{con}.

Multiple convolutional layers are often stacked to extract higher level features. Different convolutional layers may have their own window size w and value k. Every column vector of the matrix produced by the topmost k-max pooling layer is concatenated to be fed to further neural network layers.

3.4 Segment Scoring and Labeling

The fixed-sized vector produced by the topmost k-max pooling layer is fed to two standard *Linear Layers* that successively perform affine transformations over the vector, interleaved with some non-linearity function $\sigma(\cdot)$, to extract highly non-linear features. As non-linear function, we chose the sigmoidal function.

For each possible segment of an input sentence, the network outputs a score for being a correct constituent in the parse tree of the sentence. Training aims to increase scores of good segments (correspond to correct spans in the parse tree) and decrease scores of incorrect ones. Given an input sentence, after all the scores of possible segments are calculated, an efficient dynamic programming decoding algorithm is applied to find the globally optimal parse tree.

We add a simple softmax layer to the network (after removing the last scoring layer) to predict part-of-speech or syntactic labels for each segment. Labeling is trained by minimizing the cross-entropy error of the softmax layer using backpropagation. The network performs the structure prediction and labeling jointly. The two tasks shared the several layers (from the input to hidden layers) of the network. When minimizing the cross-entropy error of the softmax layer, the error will also backpropagate and influence both the network parameters and the character representations.

3.5 Dynamic Programming Decoding

In our discriminative parsing task, we aim to learn a function $f : \mathcal{X} \rightarrow \mathcal{Y}$, where $\mathcal{X}$ is a set of sentences, and $\mathcal{Y}$ is a set of valid parse trees. We denote the set of all possible parse trees that can be constructed from an input $x \in \mathcal{X}$ as $\mathcal{T}(x)$. Generally, we have to enumerate all $|\mathcal{T}(x)|$ trees and score them in order to find the best one, where $|\mathcal{T}(x)|$ is the number of parse trees that is usually exponential in the length of the sentence. However, our grammars and representations are generally structured to enable efficient inference.

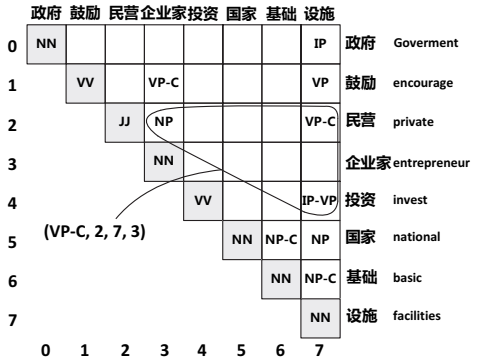

Figure 3: Grid of labeled segments.

We assign scores to local segments of a parse tree with the KMCNN, and the final score of the tree that we need for structure prediction is the sum of all the local segments (i.e., non-terminal nodes of the tree). Such model has the overlapping optimal substructure property which permits dynamic programming decompositions. The similar idea of this decomposition has previously been used for max-margin parsing [Taskar *et al.*, 2004], and in present extension to parsing langauge with recursive neural networks [Socher *et al.*, 2011].

We represent each parse tree as a set of context-free-rule tuples $(A \rightarrow BC, b, e, m)$, where A, B and C are non-terminal symbols, b is start-point, e is end-point, and m is split point. Figure 3 is the grid representation of the binary parse tree shown in Figure 1 (b). Although we illustrate the example here by taking the words as basic input units for clarity, the same dynamic programming algorithm still works when starting with the character-level. The cells in the grid except for those on the diagonal correspond to the possible segments in an input sentence. Formally, $\mathcal{R}(x, \hat{y})$ denotes a countable set of non-terminal nodes in a parse $\hat{y}$ for input x, the score of the parse tree $\hat{y}$ can be decomposed into a sum of the scores of its parts:

$$s(\theta, x, \hat{y}) = \sum_{d(b,e) \in \mathcal{R}(x,\hat{y})} s(\theta, x, d(b, e)) \tag{5}$$

where θ are all the parameters needed to compute a score s with the KMCNN, and $d(b, e)$ is a non-terminal node spanning from b to e.

3.6 Training

Given a training example (x, y), we defined a structured margin $\Delta(x, y, \hat{y})$ loss for proposing a parse $\hat{y}$ for sentence x when y is the true parse. This penalty is proportional to the number of labeled spans on which the two parse trees do not agree. In general, $\Delta(x, y, \hat{y})$ is equal to 0 if $y = \hat{y}$. The loss function is defined as a penalization of incorrect spans:

$$\Delta(x, y, \hat{y}) = \sum_{d(b,e) \in \mathcal{R}(x,\hat{y})} \kappa \mathbf{1}\{d(b, e) \notin \mathcal{R}(x, y)\} \tag{6}$$

where κ is a penalization term to each incorrect span, and $\mathcal{R}(x, y)$ is the truth parse for input x.

For a training set, we seek a function f with small expected loss on unseen sentences. The function we consider take the following form:

$$f_\theta(x) = \arg\max_{\hat{y} \in \mathcal{T}(x)} s(\theta, x, \hat{y}) \tag{7}$$

where θ are all the parameters needed to be trained. The score of a tree $\hat{y}$ is higher if the algorithm is more confident that the structure of the tree is correct. In the max-margin estimation framework, we want to ensure that the highest scoring tree is the true parse for all training instances $(x_i, y_i), i = 1, \cdots, n$, and it's score to be larger up to a margin defined by the loss. For all i in the training data:

$$s(\theta, x_i, y_i) \geq s(\theta, x_i, \hat{y}) + \Delta(x_i, y_i, \hat{y}) \tag{8}$$

These lead us to minimize the following regularized objective for n training instances:

$$\begin{aligned} J(\theta) &= \frac{1}{n} \sum_{i=1}^{n} E_i(\theta) + \frac{\lambda}{2} ||\theta||^2, \text{where} \\ E_i(\theta) &= \max_{\hat{y} \in \mathcal{T}(x_i)} (s(\theta, x_i, \hat{y}) + \Delta(x_i, y_i, \hat{y})) \\ &\quad - s(\theta, x_i, y_i) \end{aligned} \tag{9}$$

where the coefficient λ governs the relative importance of the regularization term compared with the error. Not that the correct parse tree of sentence x_i is not necessarily unique, and y_i could be one with the highest score among the multiple correct parses (See Figure 3). The loss penalizes trees more when they deviate from the correct one. Minimizing this objective maximizes the score of the correct tree, and minimizes that of the highest scoring but incorrect tree. The objective is not differentiable due to the hinge loss. we use the subgradient method to compute a gradient-like direction for minimizing the objective function.

4 Experiments

We conducted two sets of experiments. The goal of the first one is to test several variants of the network configurations on the development set, to tune hyper-parameters. The second one is to see how far we can go for Chinese constituent parsing without task-specific feature engineering. We compared the performance of the KMCNN with the existing state-of-the-art systems on the Penn Chinese Treebank 5 (CTB-5) using the standard split of data[1].

We transformed the parse trees of the CTB-5 into their binary forms for training by the method described in Section 2. At test time, the inversion operation was performed on the predicted parse trees. The standard $F1$-score was used to evaluate the performance, which is the harmonic mean of precision p and recall r, defined as $2pr/(p+r)$.

4.1 The Choice of Hyper-parameters

We tuned the hyper-parameters by trying only a few different networks. All the test results were obtained over 10 runs with different random initialization for each setting of the network. In our experiments, we chose to use the networks with two convolutional layers. Each convolutional layer is followed by a non-linearity, and a k-max pooling layer. The window sizes of the convolutional filters are 5 and 3, respectively. The value of k at the first max pooling layer is 5, and that at the top is 3.

Hyper-parameter	Value
Window sizes of two convolutional layers	$5/3$
Values k in two k-max pooling layers	$5/3$
Number of convolutional units	200
Number of hidden units	300
Character feature dimension	50
Learning rate	0.02
Penalization term κ	0.05
Regularization parameter λ	0.0001

Table 1: Hyper-parameters of the network.

Generally, the numbers of convolutional and hidden units, provided they are large enough, have a limited impact on the generalization performance. The dimension of character was set to 50, which achieved a good trade-off between training speed and performance. We also found that the performance was improved marginally when the value k of max pooling at the first layer is larger than 5, but the training time increases drastically. Training can be faster with a larger learning rate, but we preferred to stick to a small one which works, rather than finding the optimal one for speed. The network hyper-parameters used in the experiments are shown in Table 1.

4.2 Pre-training Character Embeddings

Previous work showed that the performance can be improved by using word or character embeddings learned from large-scale unlabeled data in many NLP tasks both in English [Collobert *et al.*, 2011; Socher *et al.*, 2011] and Chinese [Zheng *et al.*, 2013; Pei *et al.*, 2014]. Unsupervised pretraining guides the learning towards basins of attraction of minima that support better generalization [Erhan *et al.*, 2010]. We leveraged large unlabeled corpus to learn character embeddings, and then used these improved embeddings to initialize the character lookup tables of the networks. A Chinese Wikipedia corpus[2] containing about 667MB data was used to train the character embeddings by Word2Vec tool[3] [Mikolov *et al.*, 2013].

System	*F*1 Score
Socher et al [2011]	71.12 (RNN model)
Wang et al [2006]	77.50 (stacked classifier model)
Li [2011]	79.07 (annotated word structures)
Qian and Liu [2012]	82.85 (joint linear model)
Wang [2013]	83.42 (joint latticed-based model)
Zhang et al [2013]	84.43 (annotated word structures)
KMCNN	81.78 (pipeline)
KMCNN	84.22 (annotated word structures)

Table 2: Comparison with state-of-the-art systems

4.3 Experimental Results

The experimental results are reported in Table 2, in which our model is indicated by "KMCNN". We implemented a variant of the RNN [Socher *et al.*, 2011] which achieved an $F1$ of 71.12%. In this implementation, we start with the character-level; combine the characters into words, words into phrases, until a parse tree is generated for an input sentence. To see how well the joint solution can improve the performance, we also evaluated the parsing performance on automatically tagged data (i.e. pipeline solution). We implemented a neural network-based system of [Zheng *et al.*, 2013], which can perform joint word segmentation and POS tagging task. The $F1$-scores of this system in word segmentation and POS tagging are about 95.2% and 91.8% respectively.

The results reported in Table 2 show that the KMCNN achieved the state-of-art performance (just 0.21% difference in $F1$ to the best parser [Zhang *et al.*, 2013]) on the CTB-5 data set. The pipeline version of our network also outperformed previous pipelined systems while less dataset-specific features were used. Many other systems often used some extra heuristics or task-specific features carefully optimized to improve their performances. For example, Li [2011] proposed an adapted chart parser of [Collins, 2003], which used

[1] Sections 1-270, 400-931, 1001-1151 are used for training, sections 301-325 for development, and sections 271-300 for testing.

[2] Available at https://www.wikipedia.org/

[3] Available at http://code.google.com/p/word2vec/

the head-finding rules described by [Sun and Jurafsky, 2004]. The reported best parser of [Zhang *et al.*, 2013] integrated various contextual and structural features at the morpheme-, character-, and word-levels, carefully optimized for the task. Unlike many current parsers, our networks were not tailored to the intricacies of the CTB-5 dataset.

5 Related Work

More than a decade of parsing research for Chinese has seen the parsing performance in $F1$ improved from 73.04% [Bikel and Chiang, 2000] to 84.43% [Zhang *et al.*, 2013], thanks to the release of Penn Chinese Treebank. Bikel and Chiang [2000] constructed two word-based statistical parsers on the first release of the CTB. One is adapted from a lexicalized probabilistic context-free grammar [Miller *et al.*, 1998]; The other is based on the statistical tree-adjoining grammar [Chiang, 2000]. On sentences with $\leq$100 words, the former performed at 70.37% in $F1$, and the latter at 73.04% in $F1$. Bikel and Chiang [2002] then proposed an unsupervised learning algorithm to augment the head-finding rules, and they achieved an improved F1 of 79.93%.

Fung et al [2004] took the maximum-entropy model augmented by transformation-based learning. They achieved an $F1$ of 79.56% when tested with gold standard segmentation. Wang et al [2006] proposed a classifier-based deterministic parser for Chinese, and compared four different classifiers in making shift-reduce decisions when building parsing trees from bottom to up in one pass. They found that the classifier based on support vector machine outperformed the three others: maximum entropy, decision tree, and memory-based learning. Their best model using stacked classifiers performed at $F1$ of 77.5% on the test set for sentences $\leq$100 words with automatically generated POS tags.

The character-based parsing solution was first proposed in [Luo, 2003]. The benefit of those solutions is that they can start with the character-level, and Chinese word segmentation, POS tagging and syntax parsing can be done in a joint framework, which improves the accuracies of three sub-tasks and does not suffer from the error propagation. Luo [2003] used a maximum entropy frame for joint word segmentation, POS tagging and constituent parsing task, and he achieved an $F1$ score of 81.4% in parsing. Qian and Liu [2012] proposed a decoding algorithm for joint word segmentation, POS tagging and parsing. They trained the three individual models separately, and incorporated them together during decoding. An $F1$ score of 82.85% was achieved on the CTB-5.

Wang, Zong and Xue [2013] took a lattice-based framework for joint Chinese word segmentation, POS tagging and parsing, in which a sentence is first segmented into a word lattice, and then a POS tagger and a parser are performed on the word lattice at the same time. The experimental results on the CTB-5 show that the lattice-based framework improves the accuracy of the three sub-tasks, and has an $F1$ score of 83.42% in parsing. Zhang et al [2013] constructed a variant of transition-based model of [Zhang and Clark, 2009] for the same joint task. They manually annotated the internal structures of all Chinese words in the CTB-5, and their character-based parsing model is augmented by the features derived from the word-structure formation. Their joint model outperformed the state-of-the-art word-based parser, and achieved an improved $F1$ of 84.43% on the CTB-5.

Most of the existing work in parsing has focused on finding effective features for the model component, and on finding effective statistical techniques for parameter estimation. This approach is effective because researchers can leverage various linguistic knowledge once this knowledge is converted into the features. Although such performance improvements can be very useful in practice, there is a great temptation to optimize the performance of a system for a specific benchmark. 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.

Some studies have investigated how to use neural networks for parsing English sentences. Collobert [2011] proposed a recurrent convolutional GTN for parsing. Assuming a parse tree can be decomposed into a stack of levels, the network predicts next level of the tree taking into account predictions of previous levels. His discriminative parser achieved an $F1$ of 87.9% on the full Penn Treebank dataset with gold POS tags. According to the observation that the parse trees of natural language sentences can be seen as recursive structures, Socher et al [2010; 2011] introduced recursive neural network to recover such structures, and they achieved an unlabeled bracketing $F1$-measure of 92.1% on the Wall street Journal dataset for sentences $\leq$15 words. The greedy search algorithm was used in the RNN that recursively selects the pair which receives the highest score to be merged until the full parse tree is deduced. However, the greedy strategy does not in general produce an optimal solution.

In most cases, words are fed to the neural networks as input, while our neural network is different in that the input to the network is a sequence of characters, more raw units than words. The character is a more natural form of input in Chinese, and we can leverage large-scale unlabeled data to obtain the character representations with more syntactic and semantic information, which is impossible for Chinese words due to the lack of large high-quality pre-segmented corpora. We also explored the feasibility of estimating how likely each possible segment in a sentence is to be a constituent by a variant of convolutional neural network, and employing a dynamic programming algorithm to find the globally optimal tree. To the best of our knowledge, this study is among the first ones to perform Chinese parsing by deep learning.

6 Conclusion

We have described a novel convolutional neural network architecture with k-max pooling operation, which can capture the most active features of sentence segments based on deep learned semantic transformations of their original character-level features. Using only the character embeddings learned from large unlabeled texts, our neural networks achieved the state-of-the-art performance without labor-intensive feature engineering on the Chinese parsing task. In particular, the dynamic programming decoder allows us to efficiently learn a model which discerns among the entire space of parse trees, as opposed to evaluating the top few candidates.

7 Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments. This work was supported by a grant from Shanghai Municipal Natural Science Foundation (No. 13ZR1403800).

References

[Bikel and Chiang, 2000] Daniel M. Bikel and David Chiang. Two statistical parsing models applied to the Chinese treebank. In *Proceedings of the Second Chinese Language Processing Workshop*, 2000.

[Chiang and Bikel, 2002] David Chiang and Daniel M. Bikel. Recovering latent information in treebanks. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING'02)*, 2002.

[Chiang, 2000] David Chiang. Statistical parsing with an automatically extracted tree adjoining grammar. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL'00)*, 2000.

[Collins, 2003] Michael Collins. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637, 2003.

[Collobert and Weston, 2008] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of the International Conference on Machine learning (ICML'08)*, 2008.

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

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

[Erhan *et al.*, 2010] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, and Pascal Vincent. Why does unsupervised pre-training help deep learning. *Journal of Machine Learning Research*, 11:625–660, 2010.

[Fung *et al.*, 2004] Pascale Fung, Grace Ngai, Yongsheng Yang, and Benfeng Cheng. A maximum-entropy Chinese parser augmented by transformation-based learning. *ACM Transactions on Asian Language Information Processing*, 3(2):159–168, 2004.

[Kalchbrenner *et al.*, 2014] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL'14)*, 2014.

[Li, 2011] Zhongguo Li. Parsing the internal structure of words: A new paradigm for Chinese word segmentation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL'11)*, 2011.

[Luo, 2003] Xiaoqiang Luo. Maximum entropy Chinese character-based parser. In *Proceedings of the International Conference on Empirical Methods in Natural Language Processing (EMNLP'03)*, 2003.

[Mikolov *et al.*, 2013] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[Miller *et al.*, 1998] Scott Miller, Heidi Fox, Lance Ramshaw, and Ralph Weischedel. Sift - statistically-derived information from text. In *Proceedings of the 7th Message Understanding Conference (MUC'98)*, 1998.

[Pei *et al.*, 2014] Wenzhe Pei, Tao Ge, and Baobao Chang. Max-margin tensor neural network for chinsese word segmentation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL'14)*, 2014.

[Qian and Liu, 2012] Xian Qian and Yang Liu. Joint Chinese word segmentation, pos tagging and parsing. In *Proceedings of the International Conference on Empirical Methods in Natural Language Processing (EMNLP'12)*, 2012.

[Socher *et al.*, 2010] Richard Socher, Christopher D. Manning, and Andrew Y. Ng. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the Deep Learning and Unsupervised Feature Learning Workshop, the Conference on Neural Information Processing Systems (NIPS'10)*, 2010.

[Socher *et al.*, 2011] Richard Socher, Cliff C-Y. Lin, Andrew Y. Ng, and Christopher D. Manning. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the International Conference on Machine learning (ICML'11)*, 2011.

[Sun and Jurafsky, 2004] Honglin Sun and Daniel Jurafsky. Shallow semantic parsing of Chinese. In *Proceedings of the International Conference on Human Language Technology / North American chapter of the Association for Computational Linguistics (HLT-NAACL'04)*, 2004.

[Taskar *et al.*, 2004] Ben Taskar, Dan Klein, Michael Collins, Daphne Koller, and Christopher D. Manning. Max-margin parsing. In *Proceedings of the International Conference on Empirical Methods in Natural Language Processing (EMNLP'04)*, 2004.

[Wang *et al.*, 2006] Mengqiu Wang, Kenji Sagae, and Teruko Mitamura. A fast, accurate deterministic parser for Chinese. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (ACL'06)*, 2006.

[Wang *et al.*, 2013] Zhiguo Wang, Chengqing Zong, and Nianwen Xue. A lattice-based framework for joint Chinese word segmentation, pos tagging and parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL'13)*, 2013.

[Xue *et al.*, 2005] Nanwen Xue, Fei Xia, Fudong Chiou, and Marta Palmer. The Penn Chinese Treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(2):207–238, 2005.

[Zhang and Clark, 2009] Yue Zhang and Stephen Clark. Transition-based parsing of the Chinese treebank using a global discriminative model. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, 2009.

[Zhang *et al.*, 2013] Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. Chinese parsing exploiting characters. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL'13)*, 2013.

[Zheng *et al.*, 2013] Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. Deep learning for Chinese word segmentation and pos tagging. In *Proceedings of the International Conference on Empirical Methods in Natural Language Processing (EMNLP'13)*, 2013.