Context-Specific and Multi-Prototype Character Representations

Xiaoqing Zheng, Jiangtao Feng, Mengxiao Lin, Wenqiang Zhang
School of Computer Science, Fudan University, Shanghai, China
Shanghai Key Laboratory of Intelligent Information Processing
{zhengxq, jtfeng12, linmx14, wqzhang}@fudan.edu.cn

Abstract

Unsupervised word representations have demonstrated improvements in predictive generalization on various NLP tasks. Much effort has been devoted to effectively learning word embeddings, but little attention has been given to distributed character representations, although such character-level representations could be very useful for a variety of NLP applications in intrinsically "character-based" languages (e.g. Chinese and Japanese). On the other hand, most of existing models create a single-prototype representation per word, which is problematic because many words are in fact polysemous, and a single-prototype model is incapable of capturing phenomena of homonymy and polysemy. We present a neural network architecture to jointly learn character embeddings and induce context representations from large data sets. The explicitly produced context representations are further used to learn context-specific and multiple-prototype character embeddings, particularly capturing their polysemous variants. Our character embeddings were evaluated on three NLP tasks of character similarity, word segmentation and named entity recognition, and the experimental results demonstrated the proposed method outperformed other competing ones on all the three tasks.

1 Introduction and Motivation

Much recent research has been devoted to deep learning algorithms which achieved impressive results on various natural language processing (NLP) tasks. The best results obtained on supervised learning tasks involve an unsupervised learning phase, usually in an unsupervised pre-training step to learn distributed word representations (also known as word embeddings). Such semi-supervised learning strategy has been empirically proven to be successful by using unlabeled data to supplement the supervised models for better generalization both on English [Collobert *et al.*, 2011; Socher *et al.*, 2011; dos Santos and Zadrozny, 2014] and Chinese [Zheng *et al.*, 2013; Pei *et al.*, 2014] language processing tasks.

Due to the importance of unsupervised pre-training in deep neural network methods, many models have been proposed to improve on distributed word representations. The two main model families for learning word embeddings are: (a) predicting or scoring the current word based on its local context, such as the neural probabilistic language model (NNLM) [Bengio *et al.*, 2003], C&W [Collobert *et al.*, 2011], and continuous bag-of-words (CBOW) [Mikolov *et al.*, 2013]; (b) using the current word to predict its surrounding words, such as the Skip-gram [Mikolov *et al.*, 2013] as well as its extensions, multi-sense Skip-gram (MSSG) [Neelakantan *et al.*, 2014] and proximity-ambiguity sensitive (PAS) Skip-gram [Qiu *et al.*, 2014]. When employing these models to learn the character representations, they suffer three significant drawbacks.

First, the context is often represented by a sum (or an average) of the surrounding words' feature vectors. Such context representations may not be applicable to the cases of "character-based" languages since the meaning of a character is more ambiguous than a word, and should be determined by its neighboring characters and the rules used to combine them. Each context character in general does not contribute equally to the meaning of the target character, and thus the simple sum or average operation can not capture this *compositionality*. Some other models, [Bengio *et al.*, 2003] and [Collobert *et al.*, 2011] for example, used an affine transformation (usually following a non-linear function) over the concatenation of the neighboring words' vectors to produce the context representations. These representations are obtained with more computational cost, but still are unable to capture multiple possible interpretations of a text window which correspond to different, but possible two or more word segmentation results.

Second, most of the neural network-based models poorly utilize the global statistics of the corpus since their word embeddings are separately trained on the local context windows. The training criteria of the models like NNLM and CBOW are to correctly classify the target word. There exist two main variants of such models: one with $|\mathcal{V}|$ outputs with softmax normalization, where $|\mathcal{V}|$ is the number of words in the vocabulary, and the other with a binary hierarchical encoding tree of words, where each word is associated with a bit vector and the classification can be interpreted as a series of binary stochastic decisions with respect to bits of a vector. The Huffman tree is often used to reduce the average bit length of words. The hierarchical decomposition by a proper binary tree could provide an exponential speed-up, but this encoding schema gives less information about how similar two words are to each other. Ideally, each node in the hierarchy should convey a semantic meaning being associated with a group of similar-meaning words.

Furthermore, most of existing models on word representations create a single-prototype embedding for each word. This single-prototype representation is problematic because many words are intrinsically polysemous, and a single-prototype model is incapable of capturing phenomena of homonymy and polysemy. It is also important to be reminded that languages differ in the degree of polysemy they exhibit. For example, Chinese with its (almost) complete lack of morphological marking for parts of speech certainly exhibits a higher degree of polysemy than English.

Recent models on learning multiple representations per word [Reisinger and Mooney, 2010; Huang *et al.*, 2012] generally work as follows: for each word, first cluster its contexts into a set of clusters, and then derive multiple representations (each for a cluster) of the word from the clusters of similar contexts. In these models, each context is represented by a vector composed of the occurrences of its neighboring words or a weighted average of the surrounding words' vector. Such context definitions neglect the relative order of words in the context window. The order of words does matter to the meaning of those they form, and it impairs the quality of the multiple-prototype representations derived by the clustering based on such context representations. Besides, those models set the equal number of prototypes for each word, although different numbers were tested to determine the optimal number of prototypes. We argue that the degree of polysemy in polysemous words depends on the number of distinct contexts in which they occur. When a word appears in more different linguistic contexts, it may carry more meanings, and greater number of prototypes should be created for that word.

We propose a new context-specific language model that can learn multiple-prototype Chinese character representations, which accounts for characters with multiple senses. To verify whether our model's character representations render learning deep architectures more effective, we evaluate them on various NLP tasks with several competing models. There are three main contributions in this study. (1) We propose a novel neural network architecture to learn character embeddings from large data sets by modeling the contexts, capturing the meanings of its constituents and the rules to combine them; (2) We present a multi-prototype character representation model that is capable of capturing character's syntactic and semantic information, particularly their polysemous variants, by taking advantage of the better learned context representations from our context-specific language model; (3) We show that the character representations learned by our model outperforms those of others on the three NLP tasks of character similarity, word segmentation, and NER by transferring the unsupervised internal representations of characters into the supervised models.

2 Related Work

Over the last decade, there has been increasing interest in learning word representations from a large collection of unlabeled data, and using these word representations as a feature set to augment the supervised learners. Generally, related work on learning word representations can be divided into three categories: *clustering-based* [Brown *et al.*, 1992; Li and McCallum, 2005], *distributional* [Blei *et al.*, 2003; Teh *et al.*, 2006; Řehůřek and Sojka, 2010], and *distributed* representations [Bengio *et al.*, 2003; Mnih and Hinton, 2007; Collobert and Weston, 2008].

We focus on the distributed representation methods here. Distributed representations are dense, low-dimensional, and real-valued, which are typically induced using neural language models. Distributed word representations are also called *word embeddings*, each dimension of which represents latent feature about word syntactic and semantic information. The goal of [Bengio *et al.*, 2003] is to estimate the probability of a word given the previous words in a sentence using the cross-entropy criterion. Because the size of dictionary is usually large, the computation of exponentiation and normalization can be extremely demanding, and sophisticated approximations are required. Many approaches have been proposed to eliminate the linear dependency on the dictionary size. The model of [Mnih and Hinton, 2007] learns a linear model to predict the embedding of the last word given the concatenation of the embeddings of previous words. Collobert and Weston [Collobert and Weston, 2008] introduced a neural language model that evaluates the acceptability of a piece of text.

The related studies closest to ours in term of handling multi-prototype word representations are [Reisinger and Mooney, 2010; Huang *et al.*, 2012; Qiu *et al.*, 2014; Neelakantan *et al.*, 2014]. Reisinger and Mooney [2010] introduced a multi-prototype vector-space model, where multiple prototypes for each word are generated by clustering contexts of the word occurrence and collecting the resulting cluster centroids. The dimensionality of their word vectors is normally large, which corresponds to the size of vocabulary.

Huang et al [2012] presented a neural language model that learns the word embeddings by incorporating both local (previous words) and topic (document vectors) information. The single-prototype embeddings learned in advance are used to represent the word context, and these context representations are then clustered by spherical k-means using cosine distance. Multiple prototypes of a word are induced from its associated clusters. For multi-prototype variants, they fix the number of prototypes to be ten. It shows that the performance is sensitive to the number of prototypes, and the degree of polysemy of words relates to the number of distinct contexts in which they occur. It is unnecessary for rare words to learn several prototypes, whereas for most common words, if the number of word senses is greater than that of clusters, and certain cluster could contain the contexts of different meanings, the prototype learned by that cluster might not represent any one of the meanings well as it is influenced by several (and different) meanings of that word.

Qiu et al [2014] addressed the word-sense discrimination problem by creating multiple representations per word, each for a POS type. However, many words still may carry more than one meaning even though they are constrained to a certain POS type. A POS tagger can effectively process the texts from the same domain as the training texts, but the performance may deteriorate significantly when the tagger is used to the large texts from others, which is not conducive to the word-sense discrimination. They used a weighted average of the context word vectors to represent the context. In comparison, we finely model the contexts by a convolutional layer (or a tensor) with several feature maps, particularly to capture the multiple possible semantic interpretations and compositions of the context words.

Neelakantan et al [2014] extended the Skip-gram model to learn multi-prototype word embeddings from large unlabeled texts by adding a word-sense disambiguation layer to the network. The sense disambiguation works by maintaining multiple sense vectors per word and identifying the sense whose vector is the closest to the current context. The word sense vectors are initialized randomly and updated together with the word embeddings during the training. In their model, the word context is represented as an average of its surrounding words' vector, which might not be well enough to represent the meaning (i.e. compositionality) of the context words and would lead to poor clustering results.

3 A Neural Language Model

Various types of neural network-based models were proposed for computing distributed vector representations of words. We present here a novel neural network architecture that can jointly learn character embeddings and context representations. A convolutional layer with multiple feature maps is designed to produce the refined context representations (or vectors) that reflect the order of their constituents and the rules to combine them. The better learned context representations are then used to learn context-specific and multiple-prototype character embeddings by the sense induction with the clustering over the context vectors.

3.1 The Neural Network Architecture

The network architecture is shown in Figure 1. The input to the network is a context window, and it induces the vector for the context by a sum pooling of multiple feature maps yielded by a convolutional layer. The networks are trained to correctly classify the target words by using the context feature vectors.

We use a window approach that assumes the meaning of a character depends mainly on its surrounding characters. The characters (except the target) in the window of size w (a hyper parameter) 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-size character dictionary $\mathcal{V}$. The vector representations are stored in a character embedding matrix $\mathcal{M} \in \mathbb{R}^{d \times |\mathcal{V}|}$, where d is the dimensionality of the vector space (a hyper-parameter to be chosen) and $|\mathcal{V}|$ is the size of the dictionary.

The context feature window produced by the first lookup table is a matrix $\mathcal{H} \in \mathbb{R}^{d \times w}$, where each column of the matrix $\mathcal{H}$ is the character feature vector in the window. A one-dimensional convolution is used to yield another feature vector by taking the dot product of filter vectors with the rows of

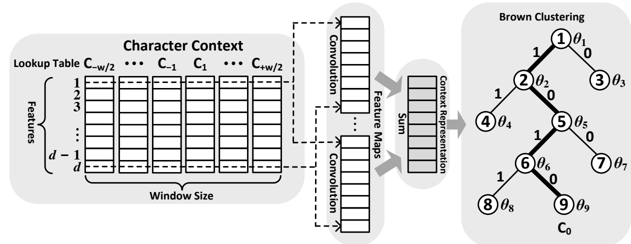

Figure 1: The neural network architecture.

the matrix $\mathcal{H}$ at the same dimension:

$$\mathcal{F} = \mathcal{H} \odot \mathcal{W} \tag{1}$$

where the weights in the matrix $\mathcal{W} \in \mathbb{R}^{w \times d}$ are the parameters to be trained, and $\mathcal{F} \in \mathbb{R}^{d}$ is a vector. The trained weights in $\mathcal{W}$ can be viewed as a linguistic feature detectors that learn to recognize a specific class of n-gram.

The convolution can be repeated to produce feature maps, hopefully capturing possible multiple, but different interpretations of character contexts. Multiple feature maps $\mathcal{F}_1, \dots, \mathcal{F}_m$ can be computed in a parallel fashion at the same layer. We add a simple sum pooling layer on the top of the convolution layer, and take the pooling results as the context representation instead.

A tensor-based transformation is also tried to model contexts with its advantage of explicitly modelling multiple interactions between feature vectors. We use a 3-way tensor $V^{[1:d]}$ to produce the context vector representation $\mathcal{Z} \in \mathbb{R}^{d}$:

$$\mathcal{Z} = \mathcal{F}_l^{\top} V^{[1:d]} \mathcal{F}_r; \mathcal{Z}_i = \mathcal{F}_l^{\top} V^{[i]} \mathcal{F}_r \tag{2}$$

where $\mathcal{F}_l$ (or $\mathcal{F}_r$) denotes the left (or right) context vector of the target character which is produced by adding up the feature vectors of the characters in the left (or right) context window, and each dimension $\mathcal{Z}_i$ is yielded by the bilinear form defined by each tensor slice $V^{[i]} \in \mathbb{R}^{d \times d}$. To speed up tensor operation, each slice is factorized into two low rank matrices: $V^{[i]} = L^{[i]} R^{[i]}$, $L^{[i]} \in \mathbb{R}^{d \times 1}$ and $R^{[i]} \in \mathbb{R}^{1 \times d}$.

3.2 Training

Once the context representation is produced, we attempt to predict the target character based on the representation. A binary hierarchical encoding tree of characters is used to speed up the training process like [Mikolov *et al.*, 2013], where each character is associated with a bit vector and the prediction can be viewed as a series of binary classification decisions with respect to bits of the vector (see Figure 1). We choose to use a binary tree based on the Brown clustering [Brown *et al.*, 1992] instead of the Huffman tree because the Brown clustering is able to assign characters to classes based on the global co-occurrence with others, which have the flavor of both semantically and syntactically based groupings. Each node in the tree of Brown clustering conveys a semantic meaning being associated with a group of similar-meaning characters, which makes the semantically similar characters to be closer to each other than those under the different nodes. Such binary hierarchical encoding tree provides some kine of global information that is complement to the local context windows.

The neural networks are trained by maximizing the conditional likelihood of the target characters given their contexts using the gradient ascent algorithm. The log-likelihood function we consider takes the following form:

$$l(\theta) = \sum_{c \in \mathcal{V}} \sum_{t \in \mathcal{T}_c} \log p_\theta(c|t) \tag{3}$$

where $\mathcal{V}$ is the dictionary of characters, $\mathcal{T}_c$ is the set of all possible character c's context windows (the target character is removed) from the data sets, and θ are the parameters needed to be trained. The distribution $p_\theta(c|t)$ can be factorized with respect to the c's bit vector as follows.

$$\begin{gathered} p_\theta(c|t) = \prod_{i=1}^{d_c} p_\theta(y_c^i|x_c, \beta_c^i) \\ p_\theta(y_c^i|x_c, \beta_c^i) = \begin{cases} \phi(x_c^\top \beta_c^i), & \text{if } y_c^i = 1; \\ 1 - \phi(x_c^\top \beta_c^i), & \text{if } y_c^i = 0. \end{cases} \end{gathered} \tag{4}$$

where x_c is the c's context vector representation produced by the networks, d_c is the depth of node c in the binary encoding tree. Each non-leaf node of the tree is associated with a binomial classifier based on logistic regression, and $\phi(\cdot)$ is a sigmoidal function. The vector of coefficients of a regression model is denoted as β, and $y_c^i \in \{0, 1\}$ are bits of the c's encoding. These lead us to maximize the following objective function:

$$\begin{aligned} l(\theta) = \sum_{c \in \mathcal{V}} \sum_{t \in \mathcal{T}_c} \sum_{i=1}^{d_c} \quad & \{y_c^i \cdot \log[\phi(x_c^\top \beta_c^i)] + \\ & (1 - y_c^i) \cdot \log[1 - \phi(x_c^\top \beta_c^i)]\} \end{aligned} \tag{5}$$

When maximizing this log-likelihood, the error will backpropagate and update both the network parameters and character embeddings.

4 Multi-Prototype Language Model

In order to learn multiple prototypes, we first construct a regression function to predict the number of character senses given the quantity of its distinct contexts. We then gather the fixed-size context windows of all occurrences of a character, and the context vector representations are extracted by collecting the outputs of the sum pooling layer (see the shaded ellipses in Figure 1). The spherical k-means algorithm [Dhillon and Modha, 2001] is used to cluster these context representations. The number of clusters for a character is given by the regression function constructed in the first step. Finally, each character in a large raw corpus is labeled with its corresponding cluster, and the sense-labeled corpus is used to train the character representations by neural network language models.

4.1 Determining the number of prototypes

As stated in the introduction, we assume that the number of prototypes of a character depends upon the quantity of distinct contexts in which it can occur. Obviously, it is desirable to choose the number of prototypes that corresponds to the character senses. To address this issue, we first want to know how many character types have a certain number of meanings, and then gain some understanding of the relationships between the number of senses in which a character can be used and the quantity of distinct contexts it fits into.

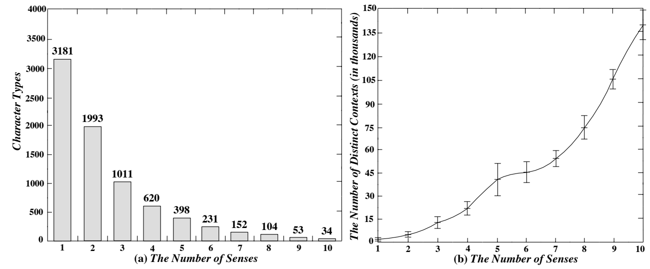

Figure 2: (a) Frequency of character types with respect to number of senses. (b) Average and standard deviation of character's contexts versus number of their senses.

To answer the first question, we downloaded a dictionary[1] that contains 7868 Chinese characters and their lexical entries, which are shown in Figure 2(a). The statistics tell us that Chinese characters on average have about 2.54 senses, but character types have a very uneven distribution. Some characters can be used in more than ten senses (there are 91 such characters, not shown in Figure 2(a), accounting for 1.16% of the character types). On the other extreme, 65.76% of the character types have only one or two meanings in the dictionary. The vast majority of word types (98.84%) have ten or fewer senses. Therefore, it is necessary to individually determine how many prototypes need to be created for each character according to its sense number.

In order to predict the number of the senses for the characters missing in the dictionary, we constructed a regression function to make such prediction by Bayesian linear regression model [Bishop, 2006]. The training data set comprises the above 7868 characters together with corresponding contexts. The contexts of a character were collected from each occurrence in a large unlabeled corpus (the duplicates are removed), and each context was composed of all unigram occurring in an 8-character window around the target character. The corpus has 268.28 million characters (6.08 million sentences) extracted from the Chinese Wikipedia. The constructed function is used to determine the appropriate number of prototypes for each character (including the missing ones in the dictionary). In Figure 2(b), we reported the average and standard deviation of character's contexts with respect to the number of senses in the data set.

4.2 Context Clustering

We use the clusters of contexts to capture meaningful variation in character usage. The contexts of characters are produced by the trained networks as mentioned above. The context vectors are then clustered into groups, each of which corresponds to a sense of the target character. To cluster these context vectors, we choose to use the spherical k-means algorithm, which has been demonstrated to model semantic relatedness well [Dhillon and Modha, 2001]. We found the spherical k-means algorithm performed better than other clustering

[1]Available at http://www.zdic.net/

methods, such as standard k-means and movMF [Banerjee *et al.*, 2005] by preliminary experiments.

The clustering algorithm is run separately for each character type, and the number of clusters will be chosen by the regression function mentioned previously. After clustering is completed, each character occurrence in the corpus is labeled to the corresponding cluster according to its context. In our implementation, the size of character dictionary grew to 16399 in the sense-labeled corpus, and the combination of character and sense label is considerred as one character. Finally, the sense-labeled corpus is used to train the multi-prototype character representations. For both word segmentation and NER tasks, the characters in the training and testing corpora are also relabeled to their associated clusters.

5 Experiments

We conducted two sets of experiments. The goal of the first one is to systematically compare different models by evaluating them on the character similarity task. The second one is to see how well the character representations learned from large unlabeled texts to enhance the supervised learning on two standard NLP tasks: word segmentation and named entity recognition, and whether the performance can be further improved by their multi-prototype versions. We used the same training data sets for all the models to be compared.

5.1 Evaluation Tasks

This section describes three NLP tasks on which different character representations are compared: character similarity, word segmentation, and NER. We wish to find out whether some representations are preferable for certain tasks. For the word similarity task, the Spearman's rank correlation coefficient is used to compare correlations between the similarity scores given by the models and those rated by humans. The other two tasks are evaluated by computing the standard F1-score, which is the harmonic mean of precision and recall.

- Character similarity focuses on ranking the character pairs in Chinese by their similarities. To the best of our knowledge, there is no such data set available. We constructed a Csim-305 data set, which contains three hundred and five different character pairs. Each pair is assigned the similarity score by twenty three native speakers, and the average of those human judgements as the final score. The scores range from 0 to 10, and the higher score, the more similar two characters will be. An average Spearman's rank correlation coefficient achieved by human annotators is 0.85, and the hightes score is 0.91.
- Word segmentation is used to reconstruct the word boundaries of texts in those languages that are written without using whitespace to delimit words. We picked Penn Chinese Treebank (CTB-7) as our data set [Xue *et al.*, 2005].
- Name entity recognition seeks to locate and classify elements in the sentence into pre-defined categories such as the names of persons, organizations, locations, etc. For the NER task, we choose to use MSRA data set from Bakeoff-3 with standard train-test splits [Levow, 2006].

5.2 Supervised NLP system

For word segmentation and NER tasks, we assume that one can take an existing, close to state-of-the-art, supervised NLP system, and its performance can be further improved by transferring the unsupervised character representations into the system. We used the neural network of [Zheng *et al.*, 2013] as such supervised component, which was designed to perform the sequence tagging tasks particularly for "character-base" languages like Chinese.

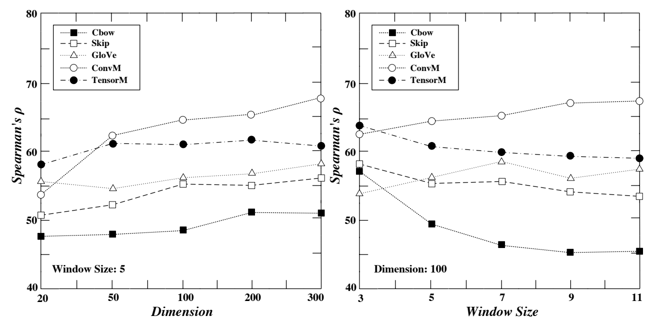

Figure 3: (a) Average Spearman's ρ versus dimension. (b) Average Spearman's ρ versus window size. In (a), the window size is 5. In (b), the dimension is 100.

5.3 Word Similarity

We reported in Figure 3 the Spearman's ρ versus window size and versus vector size for our models and other competitors on the CSim-305. The proposed convolution-based model is denoted as ConvM, and the tensor-based one as TensorM. The number of feature maps used by the ConvM was set to five. We compared the performance of ours against three state-of-the-art models: GloVe [Pennington *et al.*, 2014], CBOW and SKIP [Mikolov *et al.*, 2013]. All the results reported have been averaged over five runs.

Table 1: Nearest characters.

Character	CBOW	SKIP	GloVe	ConvM	TensorM
猫(cat)	狗(dog) 鬼(ghost) 兽(beast)	狗(dog) 熊(bear) 怪(monster)	狗(dog) 熊(bear) 狐(fox)	狗(dog) 兔(rabbit) 犬(canine)	狗(dog) 獾(badger) 鹰(hawk)
蓝(blue)	绿(green) 彩(colorful) 橙(orange)	绿(green) 红(red) 色(color)	绿(green) 红(red) 色(color)	黄(yellow) 红(red) 黑(black)	橙(orange) 绿(green) 棕(brown)
五(five)	六(six) 七(seven) 三(three)	七(seven) 八(eight) 六(six)	六(six) 七(seven) 十(ten)	三(three) 七(seven) 四(four)	七(seven) 六(six) 八(eight)
河(river)	湖(lake) 溪(creek) 闽(Min)	流(flow) 湖(lake) 溪(creek)	湖(lake) 岸(bank) 北(north)	湖(lake) 溪(creek) 岭(ridge)	湖(lake) 溪(creek) 渭(Wei)
我(I)	他(he) 牠(it) 你(you)	你(you) 谁(who) 您(you)	你(you) 么(huh) 们(pl. suffix)	它(it) 他(he) 你(you)	他(he) 你(you) 它(it)
喝(drink)	吧(bar) 窖(cellar) 酗(drunken)	吃(eat) 醉(drunk) 吧(bar)	吆(whew) 吃(eat) 酒(wine)	吃(eat) 有(have) 咬(bite)	吃(eat) 酗(drunken) 炒(fry)

It can be seen from Figure 3(a) that generally, the larger the dimension, the higher the Spearman's ρ we will have. The ConvM achieved the highest Spearman's ρ (67.58%) when the window size was set to 5 and the dimension to 300. Figure 3(b) shows that the performance drops smoothly when the window size is larger than 3 except the ConvM. A reasonable explanation for this is that most Chinese words contain no more than 3 characters, and the neighboring characters outside word boundaries become "noise" when we try to capture the semantic and syntactic information about the central character. The increasing point of entropy of successive characters is the location of a word boundary [Jin and Tanaka-Ishii, 2006]. The models will suffer from such phenomenon if they do not model the character contexts properly. The ConvM and TensorM were designed to capture this complicated, but critical "compositionality" of the surrounding characters. The TensorM, therefore, is less affected by the variation in the window size while the ConvM can even benefit from the larger ones.

The reason of such different behavior is that a convolutional layer with multiple feature maps is used to produce the refined context representations, which reflect the order of their constituents and the rules to combine them (i.e. the weights of context characters). The preliminary experiments showed that if the Huffman tree was used instead of the Brown clustering, the performance of our models drops about 3$\sim$5%. Table 1 shows the nearest neighbors of six characters (randomly chosen) for the models mentioned above. We see that the nearest neighbors discovered by our models are more semantically coherent than those by the others.

5.4 Word Segmentation and NER

Table 2 shows the final word segmentation results, and Table 3 the NER results. For the models to be evaluated here, the dimension was set to 100, and the window size to 9, because we prefer to "observe" a character within a slightly larger window to better discover its syntactic and semantic information. To train the multi-prototype character representations, the learning algorithms were fed with the sense-labeled corpus in which each character occurrence is labeled with its corresponding cluster. The results obtained using multi-prototype representations were indicated by "multiple". Unlike ours, there are no direct way to obtain the context representations from the results of the CBOW, SKIP, and GloVe, their context vector representations are simply collected by the sum of the surrounding characters' feature vectors, and then fed into the clustering algorithm for multi-prototype character representation learning like [Huang *et al.*, 2012].

The results shown in Table 2 suggest that unsupervised pretraining gives consistently better generalization comparing to the baseline started with randomly initialization (without pretraining embeddings), and our models performed slightly better than all the competitors. It is worth noting that our models' standard deviations of the F1-scores are much less than the others, which means that it is more likely for them to obtain the character embeddings with good enough performance for just one-run. The results also show that all the multi-prototype enhancements can further improve the performance of the task. The ConvM achieved higher increase in F1-score with 0.85% difference. The results for the NER task is reported in Table 3, and we found the similar trends as that for the word segmentation task. The results for both benchmark datasets show that our models achieved consistently higher performance over the three competitors.

Table 2: Word segmentation results.

Approach		Precision	Recall	F1-score
Baseline		92.31	90.22	91.25 ± 0.0089
Single	CBOW	93.06	91.40	92.22 ± 0.0076
	SKIP	93.40	92.28	92.84 ± 0.0024
	GloVe	93.12	91.39	92.25 ± 0.0067
	ConvM	93.64	92.45	93.04 ± 0.0015
	TensorM	93.89	93.03	93.46 ± 0.0017
Multiple	CBOW	93.29	91.80	92.54 ± 0.0045
	SKIP	93.88	92.60	93.23 ± 0.0056
	GloVe	92.85	91.78	92.31 ± 0.0032
	ConvM	94.50	93.29	93.89 ± 0.0018
	TensorM	94.01	93.59	93.80 ± 0.0026

Table 3: Name entity recognition results.

Approach		Precision	Recall	F1-score
Baseline		82.02	80.30	81.15 ± 0.0286
Single	CBOW	82.64	80.46	81.53 ± 0.0020
	SKIP	85.50	82.60	84.02 ± 0.0154
	GloVe	82.90	80.30	81.58 ± 0.0289
	ConvM	85.81	83.07	84.42 ± 0.0054
	TensorM	86.10	82.97	84.51 ± 0.0071
Multiple	CBOW	83.32	80.55	81.91 ± 0.0092
	SKIP	85.77	83.58	84.66 ± 0.0074
	GloVe	83.21	80.61	81.89 ± 0.0081
	ConvM	86.70	83.49	85.06 ± 0.0065
	TensorM	86.85	83.20	84.98 ± 0.0072

6 Conclusion

Word or character features can be learned in advance in an unsupervised manner. These features, once learned, are easily disseminated with other researchers, and easily integrated into existing supervised NLP systems. We presented a new neural network architecture as well as a context-specific language model that can learn multi-prototype character representations, which are capable of capturing character-level syntactic and semantic information, particularly their polysemous variants. Experiment results with different datasets showed that the proposed ConvM and TensorM outperformed the three state-of-the-art embedding learning methods on the three NLP tasks. The ConvM achieved the best results on all the three evaluation tasks. In the future, it would be interesting to see how well our models can be used to learn character or word representations for other languages.

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, and No. 15511104303).

References

[Banerjee *et al.*, 2005] Arindam Banerjee, Inderjit S. Dhillon, Joydeep Ghosh, and Suvrit Sra. Clustering on the unit hypersphere using von mises-fisher distributions. *Journal of Machine Learning Research*, 6:1345–1382, 2005.

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

[Bishop, 2006] Christopher M. Bishop. *Pattern recognition and machine learning*, chapter Linear models for regression, pages 152–160. Springer, New York, 2006.

[Blei *et al.*, 2003] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[Brown *et al.*, 1992] P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai. Class-based n-gram models of natural language. *Computational Linguistics*, 18:467–479, 1992.

[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.

[Dhillon and Modha, 2001] Inderjit S. Dhillon and Dharmendra S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1):143–175, 2001.

[dos Santos and Zadrozny, 2014] Cícero Nogueira dos Santos and Bianca Zadrozny. Learning character-level representations for part-of-speech tagging. In *Proceedings of the International Conference on Machine learning (ICML'14)*, 2014.

[Huang *et al.*, 2012] Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL'12)*, 2012.

[Jin and Tanaka-Ishii, 2006] Z. Jin and K. Tanaka-Ishii. 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)*, pages 428–435, Sidney, Australia, 2006.

[Levow, 2006] Gina A. Levow. 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)*, 2006.

[Li and McCallum, 2005] W. Li and A. McCallum. Semi-supervised sequence modeling with syntactic topic models. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI'05)*, 2005.

[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.

[Mnih and Hinton, 2007] A. Mnih and G. Hinton. Three new graphical models for statistical language modelling. In *Proceedings of the International Conference on Machine learning (ICML'07)*, 2007.

[Neelakantan *et al.*, 2014] Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of the International Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*, 2014.

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

[Pennington *et al.*, 2014] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: global vectors for word representation. In *Proceedings of the International Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*, 2014.

[Qiu *et al.*, 2014] Lin Qiu, Yong Cao, Zaiqing Nie, and Yong Rui. Learning word representation considering proximity and ambiguity. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL'14)*, 2014.

[Reisinger and Mooney, 2010] Joseph Reisinger and Raymond J. Mooney. Multi-prototype vector-space models of word meaning. In *Proceedings of the 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL'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.

[Teh *et al.*, 2006] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

[Řehůřek and Sojka, 2010] R. Řehůřek and P. Sojka. Software framework for topic modelling with large corpora. In *Proceedings of the Language Resources and Evaluation Conference (LREC'10)*, 2010.

[Xue *et al.*, 2005] Nianwen Xue, Fei Xia, Fudong Chiou, and Martha Palmer. The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(2):207–238, 2005.

[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.