Semantic Expansion using Word Embedding Clustering and Convolutional Neural Network for Improving Short Text Classification

Peng Wang[a,], Bo Xu[a], Jiaming Xu[a], Guanhua Tian[a], Cheng-Lin Liu[a,b], Hongwei Hao[a]*

[a]Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, P.R. China
[b]National Laboratory of Pattern Recognition (NLPR), Beijing, 100190, P.R. China

Abstract

Text classification can help users to effectively handle and exploit useful information hidden in large-scale documents. However, the sparsity of data and the semantic sensitivity to context often hinder the classification performance of short texts. In order to overcome the weakness, we propose a unified framework to expand short texts based on word embedding clustering and convolutional neural network (CNN). Empirically, the semantically related words are usually close to each other in embedding spaces. Thus, we first discover semantic cliques via fast clustering. Then, by using additive composition over word embeddings from context with variable window width, the representations of multi-scale semantic units[1] in short texts are computed. In embedding spaces, the restricted nearest word embeddings (NWEs)[2] of the semantic units are chosen to constitute expanded matrices, where the semantic cliques are used as supervision information. Finally, for a short text, the projected matrix[3] and expanded matrices are combined and fed into CNN in parallel. Experimental results on two open benchmarks validate the effectiveness of the proposed method.

Keywords: Short Text, Classification, Clustering, Convolutional Neural Network, Semantic Units, Word Embeddings, Semantic Cliques

*Corresponding author

[1]Semantic units are defined as n-grams which have dominant meaning of text. With n varying, multi-scale contextual information can be exploited.

[2]In order to prevent outliers, a Euclidean distance threshold is preset between semantic cliques and semantic units, which is used as restricted condition.

[3]The projected matrix is obtained by table looking up, which encodes Unigram level features.

1. Introduction

The classification of short texts, such as search snippets, micro-blogs, product reviews, and short messages, plays important roles in user intent understanding, question answering and intelligent information retrieval [1]. Since short texts do not provide enough contextual information, the data sparsity problem is easily encountered [2]. Thus, the general methods based on bag-of-words (BoW) model cannot be directly applied to short texts [1], because the BoW model ignores the order and semantic relations between words. How to acquire effective representations of short texts to enhance the categorization performance has been an active research issue [2, 3].

Conventional text classification methods often expand short texts using latent semantics, learned by latent Dirichlet allocation (LDA) [4] and its extensions. Phan et al. [3] presented a general framework to expand short and sparse texts by appending topic names, discovered using LDA over Wikipedia. Sahami and Heilman [5] enriched text representation by web search results using the short text segment as a query. Furthermore, Yan et al. [6] presented a variant of LDA, dubbed biterm topic model (BTM), especially for short text modeling to alleviate the data sparsity problem. However, these methods still consider a text as BoW. Therefore, they are not effective in capturing fine-grained semantics for short texts modeling.

More recently, deep learning based methods have drawn much attentions in the field of natural language processing (NLP), which mainly evolved into two branches. One is to learn word embeddings by training language models [7, 8, 9, 10], and another is to perform semantic composition to obtain phrase or sentence level representation [11, 12]. Word embeddings, also known as distributed representations, typically represent words with dense, low-dimensional and real-valued vectors. Each dimension of the vectors encodes a different aspect of words. In embedding spaces, semantically close words are likely to cluster together and form semantic cliques. Moreover, the embedding spaces exhibit linear structure that the word embeddings can be meaningfully combined using simple vector addition [9].

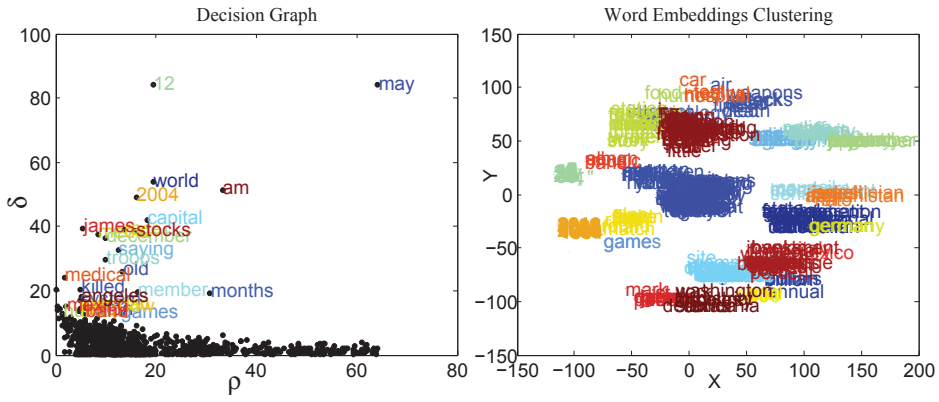

Figure 1: Fast clustering based on density peaks of embeddings.

In this paper, we aim to obtain the semantic representations of short texts and overcome the weakness of conventional methods. Similar to Li et al. [13] that cluster indicators learned by nonnegative spectral clustering are used to provide label information for structural learning, we develop a novel method to model short texts using word embeddings clustering and convolutional neural network (CNN). For concision, we abbreviate our methods to CCNN, as Clustering and CNN are employed. Particularly, the fast clustering algorithm, based on density peaks searching [14], is first utilized to discover the semantic cliques in embedding spaces, as shown in Figure 1. Then, the component-wise additive composition is performed over word embeddings, from context with variable length, to compute the representations of semantic units appearing in short texts, as shown in Figure 2. The semantic units are used to calculate Euclidean distance with each semantic clique, and their nearest word embeddings (NWEs) can

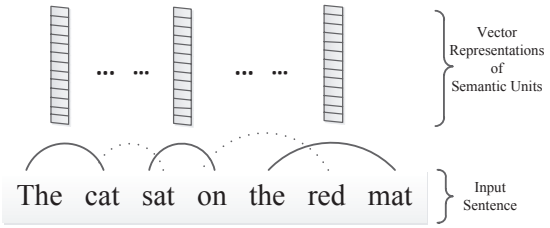

Figure 2: The detection of semantic units.

be found. In our framework, the NWEs that meet the preset threshold of Euclidean distance are chosen to constitute the expanded matrices for short texts enrichment, otherwise simply dropout. In this stage, the semantic cliques are used as supervision information to detect precise semantics. Finally, a CNN with one convolutional layer followed by a K-max pooling layer is trained under the cross entropy objective, which is optimized with mini-batches of samples iteratively by back propagation (BP).

The motivation of the proposed method is to introduce semantic knowledge and expand short texts by related word embeddings, which is pre-trained over large-scale external corpus. To preserve the semantics in original short texts, we integrate text understanding and vectorization into a joint framework. As shown in Figure 2, for the input short text "*The cat sat on the red mat*", three semantic units can be detected with different window width. These multi-scale semantic information is leveraged to expand the short text, and its context is fully exploited.

The main contributions of this paper are summarized as follows:

(1) The density peaks searching based clustering method is utilized to discover semantic cliques, which are used as supervision information to extract fine-tuned semantics.

(2) Multi-scale semantic units are defined and their representations are calculated by using a one-dimensional convolution-like operation.

(3) In embedding spaces, the restricted NWEs of semantic units are discovered to produce expanded matrices. Then, the projected matrix and the expanded matrices are simply combined and fed into a CNN to extract high-level features.

Experiments are conducted on Google snippets [3] and TREC [15] to validate the effectiveness of our method.

The rest of this paper is organized as follows. Section 2 gives a brief review of related works. Section 3 introduces the theoretical foundation of our work, including semantic composition and word embeddings clustering. Section 4 defines the relevant operators and hierarchies of the framework. Section 5 presents our experimental results. Finally, concluding remarks are offered in Section 6.

2. Related Works

In order to overcome the data sparsity problem in short texts representations, many popular solutions have been proposed. Based on external Wikipedia corpus, Phan et al. [3] proposed a method to discover hidden topics using LDA and expand short texts. Zhou et al. [16] exploited semantic information from Wikipedia to enhance the question similarity in concept space. Chen et al. [2] proved that leveraging topics at multiple granularity can model short texts more precisely.

In recent years, neural networks (NNs) relevant methods have been used to model languages with promising results, and word embeddings can be learned meanwhile [17]. Mikolov et al. [9] introduced the continuous Skip-gram model that is an efficient method for learning high quality word embeddings from large-scale unstructured text data. Furthermore, various pre-trained word embeddings are publicly available, and many composition-based methods are proposed to induce semantic representations of texts.

To obtain sentence-level representations of texts, NNs related works can be divided into two types, which are respectively used for universal tasks and special tasks. For the former, Le and Mikolov [12] proposed the paragraph vector to learn a fixed-size feature representation for documents with variable length. Kalchbrenner et al. [18] introduced the dynamic convolutional neural network (DCNN) for modeling sentences, which is the most related work to our study. In that work, dynamic k-max pooling is utilized to capture global features without relying on parse tree. Based on convolutional architecture, Kim [19] proposed a simple improvement that two input channels are used which allow the employment of dynamic-updated and static word embeddings simultaneously. These methods can be used to generate semantic representations of texts for various tasks.

For the latter, Zeng et.al [20] developed a deep convolutional neural network (DNN) to extract lexical and sentence level features, which are used for relation classification. Socher et al. [21] proposed the recursive neural network (RNN) that has proven to be effective in sentiment prediction. In order to reduce the overfitting problem of neural network, especially trained on small data set, Hinton et al. [22] used random dropout

to prevent complex co-adaptations.

Standard recurrent neural networks can take into account all of the predecessor words for modeling languages [23]. However, it is difficult to train due to the vanishing gradient problem, which can be explicitly avoided in Long Short-Term Memory (LSTM) algorithm [24]. Currently, LSTM is widely used in spoken language understanding and sequence prediction [25, 26, 27].

Although the methods discussed above can capture high-order n-grams and word order information to produce complex features, the small length of short texts still heavily affects the classification performance. In this paper, we design a novel method to detect multi-scale semantic units for expanding short texts.

3. Theoretical Foundation

3.1. Semantic Composition

In vector spaces, words can be described by real-valued vectors, such as one-hot representations and word embeddings. However, the fundamental problems appear in one-hot representations of words include data sparsity and the curse of dimensions, which make language models and other learning algorithms difficult to use [7]. Furthermore, the one-hot representations ignore the dependency among words in context and cannot be used to measure words similarity.

The recently introduced neural network language models, especially the continuous Skip-gram model [8] can be efficiently used to learn high-quality word embeddings, where each component of the vectors might have a semantic or grammatical interpretation. The training objective of continuous Skip-gram model is to learn words representations that are good at predicting their context. Thus, the co-occurrence information can be effectively used to describe each word.

Moreover, the word embeddings can capture various syntactical and semantic relationships. For example,

$$vec\,(Germany) + vec\,(Capital) \approx vec\,(Berlin) \tag{1}$$

$$vec(Athlete) + vec\,(Football) \approx vec\,(Football\_Player) \tag{2}$$

Where $vec(\cdot)$ is a special word embedding. The above examples indicate that the additive composition over word embeddings can often produce meaningful results. Thus these words that cannot be observed directly can be composed by using basic mathematical operations on word embeddings. In Equation (1), the token $'Berlin'$ can be viewed that it has a embedding offset $vec\,(Capital)$ to the token $'Germany'$ in embedding space. The embedding offsets represent the semantic relations among words. This merits make it possible to meaningfully combine words by element-wise addition.

The theoretical interpretation of the additive property can be obtained by reviewing the learning procedure of word embeddings [9]. Since it is trained with the objective that predicts the surrounding words in a context, the word embedding encodes the implicit distribution of the context. Thus, these words often appearing in the similar context will obtain approximately equal vector representations, which cluster together and constitute semantic cliques in embedding spaces, as shown in Figure 1. The values of each word embedding are logarithmically related to the probabilities output by softmax function of Skip-gram model. Thus, the sum of two word embeddings is proportional to the product of the two corresponding context distributions, which induce the joint probability distribution of the two word contexts. While the joint probability distribution implies the co-occurrence of these contexts information, which in reverse can produce the vector representation of composition result.

Semantic composition in embedding spaces has recently received much attention [9, 28, 29]. Composition based methods can be useful for discovering latent semantics and obtaining the vector representations of phrases or sentences, as shown in Figure 2. The composition results obtained from co-occurrences can be used to analyze similarities of phrases [9], and as input feature for classifiers, which help language understanding.

3.2. Word Embedding Clustering

In embedding spaces, the neighbors of each word are generally semantically related [9]. Therefore, clustering methods can be used to discover semantic cliques. However, the number of semantic cliques is unknown in advance, and the vocabulary size of word embeddings is usually large. For example, the publicly available word

embeddings pre-trained by $Word2Vec$[4] contain three million words. In order to handle these problems, we adopt the fast algorithm based on searching density peaks [14] to perform word embeddings clustering.

The clustering algorithm assumes that cluster centers are surrounded by neighbors with lower local density and they are at a relatively large distance from any points with a higher local density, which exactly meet the distributed property of word embeddings. For implementation, two quantities of data point i are computed, include: local density ρ_i and distance δ_i from points of higher density, which are defined as follows:

$$\rho_i = \sum_j \chi(d_{ij} - d_c) \tag{3}$$

where d_{ij} is the distance between data points, d_c is a cutoff distance, and

$$\chi(\cdot) = \begin{cases} 1\ ,\ if\ \ d_{ij} < d_c \\ 0\ ,\ \ \text{otherwise} \end{cases} \tag{4}$$

Thus, ρ_i is equal to the number of points that are closer than d_c to point i. Furthermore, δ_i is measured by:

$$\delta_i = \begin{cases} \min\limits_{j:\rho_j > \rho_i} (d_{ij})\ \ ,\ \ if\ \ \rho_i < \rho_{\max} \\ \max\limits_j (d_{ij})\ \ \ ,\ \ \ otherwise \end{cases} \tag{5}$$

An simple example of word embeddings clustering is illustrated in Figure 1. The decision graph shows the two quantities ρ and δ of each word embedding. According to the definitions above, these word embeddings with large ρ and δ simultaneously are chosen as cluster centers, which are labeled using the corresponding words in the decision graph of Figure 1.

4. Proposed Method

In this section, a unified framework used for short texts modeling and classification is described, as shown in Figure 3. Our method aims to introduce external knowledge by taking advantage of the well pre-trained word embeddings and exploit more contextual information of short texts to improve classification performance.

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

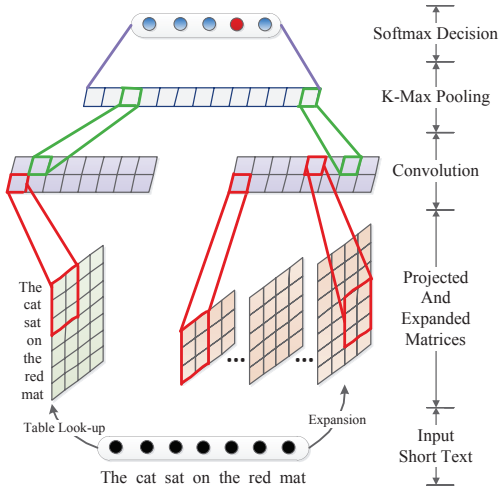

Figure 3: Neural network for short text modeling.

For a short text $S = \{w_1, w_2, \cdots, w_N\}$, the framework takes the input as a sequence of N tokens, which are contained by a finite vocabulary D. In the first layer, these tokens are transformed into real-valued word embeddings by table looking up, and the projected matrix $\mathbf{PM} \in \mathbf{R}^{d \times N}$ is obtained, which can be induced using the matrices product as follows:

$$\mathbf{PM} = \mathbf{LT} \cdot index(S) \tag{6}$$

where the lookup table $\mathbf{LT} \in \mathbf{R}^{d \times v}$ is initialized by pre-trained word embeddings that encode word-level information, d is the dimension of the embedding, v is the size of vocabulary D, and $index(\cdot)$ is the function that transform each word in S into one-hot representation, which is corresponding to the vocabulary D of the lookup table $\mathbf{LT}$.

The main functions of the second layer, exhibited in Figure 3, are to obtain multi-scale semantic units via supervised strategy and produce the expanded matrices, as shown in Figure 4. The expanded matrices are simply combined with the projected matrix and fed into convolutional layer, where high-level local features are extracted. Then, the k-max pooling layer is used to down-sample the output feature maps of the convolutional layer, and global features are generated. Subsequently, the pooling results are directly concatenated to produce the vector representation of the input short text. At last, a softmax decision function is employed as classifier.

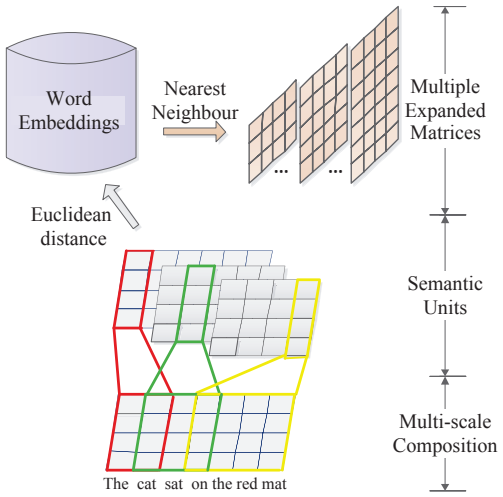

Figure 4: Expanded matrices for short texts.

The given short text is passed through the sequence of layers, and the sentence-level features are extracted. The detection for multi-scale semantic units, which are used to expand short texts, is the main novelty of this work. In the following, the details of the framework are described.

4.1. Architecture Description

4.1.1. Semantic Units Detection

For a short text S, methods to obtain the feature representation mainly have two problems: the length of S is variable; the semantic meaning of S is often determined by a few of key-phrases, however, these meaningful phrases may appear at any position of S. Thus, simply combining all words of S may introduce unnecessary divergence and hurt the effectiveness of the overall semantic representation. Therefore, the detection for the semantic units are useful, which capture salient local information, as shown in Figure 2.

The main idea of the detection for semantic units is to define a convolution-like operation to perform semantic composition over word embeddings from context, where multiple windows with variable width are used. Then sentence-wide semantic units are discovered and multi-scale contextual information can be exploited, which is helpful

to reduce the impact of ambiguous words.

Particularly, to obtain the representations of semantic units, a window matrix $E_{win} \in R^{d \times m}$ with all weights equal to one is used to convolve with the projected matrix $\mathbf{PM}$. The essence of the operation is a one-dimensional convolution, which is defined as follows:

$$[\mathbf{seu}_1, \mathbf{seu}_2, \cdots, \mathbf{seu}_{l-m+1}] = \mathbf{PM} \otimes \mathbf{E}_{win} \tag{7}$$

where,

$$\mathbf{seu}_i = \sum_{j=1}^{|\mathbf{PM}^{win,i}|} \mathbf{PM}_j^{win,i} \tag{8}$$

$\mathbf{PM}_j^{win,i}$ is the jth column from the sub-matrix $\mathbf{PM}^{win,i}$, which is windowed on projected matrix $\mathbf{PM}$ by E_{win} with the ith times sliding. m is the width of the window matrix E_{win}, and l is the length of input short text. As shown in Equation (8), the ith semantic unit $\mathbf{seu}_i \in \mathbf{R}^d$ is the component-wise summation of the columns in $\mathbf{PM}^{win,i}$, which have the same dimension with each word embedding.

Since meaning related words often close to each other and form semantic cliques in the embedding spaces, each meaningful semantic unit is assumed that it has one close embedding neighbor at least. In order to recognize precise semantic units, we compute Euclidean distance between semantic units and semantic cliques, as shown in Figure 4. A preset distance threshold is used as restricted condition to fine-tune the detection for semantic units. In particular, for a semantic unit, the nearest semantic clique center is searched firstly, and then the NWEs in semantic clique can be discovered fast. If the distance between the semantic unit and the NWEs are smaller than the threshold, the NWEs are select to constitute the expanded matrices $\mathbf{EMs}$, otherwise dropout. Therefore, the semantic cliques are used as supervision information to extract more precise features.

Corresponding to a window matrix E_{win} with certain width m, the restricted nearest embedding neighbors of semantic units are selected to constitute one expanded matrix. By increasing the window matrices with distinct width, multiple expanded matrices can be computed in parallel, which guarantees the merit of the proposed architecture that multi-scale contextual information can be used to expand the input short texts. As

described above, the width m of each window matrix is a critical factor that impacts the extraction of effective information.

4.1.2. Convolution Layer

After expansion for short texts, a convolutional layer is used to extract local features. In our framework, the projected matrix $\mathbf{PM}$ and expanded matrices $\mathbf{EMs}$ are fed into the convolutional layer in parallel. Kernel matrices of weights $\mathbf{k} \in R^{2\times n}$ with certain widths n are utilized to calculate convolution with the input matrices.

The number of kernel matrices and their width n are hyper-parameters of the network. As shown in Figure 3, to obtain the feature map $\mathbf{C}$, the convolutional operation is defined as taking the inner product of the kernel matrices $\mathbf{k}$ with pair-wise rows of each input matrix denoted by $\mathbf{X}$, as Equation (9).

$$\mathbf{C} = \begin{pmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \\ \vdots \\ \mathbf{c}_{d/2} \end{pmatrix} = \begin{pmatrix} \mathbf{k}_1 \\ \mathbf{k}_2 \\ \vdots \\ \mathbf{k}_{d/2} \end{pmatrix} \otimes \begin{pmatrix} \mathbf{X}_1^{win} \\ \mathbf{X}_2^{win} \\ \vdots \\ \mathbf{X}_{d/2}^{win} \end{pmatrix}^{\mathrm{T}} \tag{9}$$

where,

$$c_i^j = \mathbf{k}_i \cdot (\mathbf{X}_i^{win,j})^T \tag{10}$$

The c_i^j is real-valued and generated by inner-product. $\mathbf{X}_i^{win,j}$ is the submatrix windowed by $\mathbf{k}_i$ for jth times sliding on $\mathbf{X}$. The weights in $\mathbf{k}_i$ are part of parameters to be learned in training stage, which is corresponding to linguistic feature detectors and learns to recognize a specific class of n-grams.

To make the convolutional layer more robust, the weights of kernels for $\mathbf{PM}$ and $\mathbf{EMs}$ are learned respectively, since the word embeddings in $\mathbf{PM}$ contain order information, whereas $\mathbf{EMs}$ do not. In Equation (9), each kernel matrix is calculated convolution with pair-wise rows of input matrices. Thus the feature detectors are not independent to single dimension of word embeddings, and the folding operation is omitted, which appeared in [18].

4.1.3. K-Max Pooling

The feature map $\mathbf{C}$, in Equation (9), encoder local features and its size depends on the length of input short texts and the number of expanded matrices. With the aim of capturing most relevant global features with fixed-length, and enabling the output features to adapt for various classifiers, a K-max pooling operation is used to down-sample the feature maps $\mathbf{C}$, as follows:

$$\hat{C} = \max^{(k)}(C) \tag{11}$$

The max pooling operator is a non-linear subsampling function that returns the maximum of a sequence of values [30]. In our architecture, the K-max pooling operation is applied over each row of feature map C to return the sub-sequence of K maximum values, instead of the single maximum value, where K is a hyper-parameter optimized during training.

At last, the tangent function is chosen to perform non-linear and element-wise transformation over the down-sampled feature map $\hat{C}$ from K-max pooling layer,

$$\hat{f} = \tanh(\hat{C}) \tag{12}$$

thus, the feature representations of input short texts are obtained.

4.1.4. Output Layer

After short texts are put through the sequence of layers described above, semantic representations $\hat{f}$ with fixed-size are obtained. The last layer of our framework is fully connected with weights $\mathbf{W}_z$. For a short text $\mathbf{x}_i$, a linear transformation is first performed, as follows:

$$\phi(\mathbf{x}_i, \mathbf{W}_z) = \mathbf{W}_z \hat{f} \tag{13}$$

The output of Equation (13) is a vector with dimension of $|C|$, where C is the tags set. Each component of the output vector can be viewed as a possible score of the corresponding class.

Then, a softmax function is utilized to transform the score vector into a probability distribution,

$$p(c_j|\mathbf{x}_i, \mathbf{W}_z) = \frac{\exp(\phi_j(\mathbf{x}_i, \mathbf{W}_z))}{\sum_{j=1}^{|C|} \exp(\phi_j(\mathbf{x}_i, \mathbf{W}_z))} \tag{14}$$

At last, the class c_j with maximum $p(c_j|\mathbf{x}_i, \mathbf{W}_z)$ is chosen as the predicted label for $\mathbf{x}_i$.

4.2. Network Training

The network is trained with the objective that minimizes the cross-entropy of the predicted distributions and the actual distributions for all samples. The cross-entropy function is proven to be able to accelerate the back propagation algorithm and provide good overall network performance with relatively short stagnation periods [31], especially for classification task. During training the neural network, the set of parameters $\theta = \{\mathbf{k}, \mathbf{W}_z\}$ need to be optimized, where $\mathbf{k}$ is the kernel weights from convolutional layer, and $\mathbf{W}_z$ is the connective weights from output layer.

To construct the objective function, the cross-entropy loss function is considered and an L_2 regularization term [22] is introduced to prevent over-fitting problem over parameters set θ, as follows:

$$J(\theta) = -\frac{1}{t}\sum\nolimits_{i=1}^{t} \log p(c^{\dagger}|\mathbf{x}_i, \theta) + \alpha\|\theta\|^2 \tag{15}$$

where $c^{\dagger}$ is the correct class of input text $\mathbf{x}_i$, α is the factor of regularization term, and t is the number of training samples.

The network is learned with mini-batches of samples by back-propagation (BP). In order to deduce the BP updates for the parameters set θ, the gradient-based optimization is performed using the Adagrad update rule [32]. For each iteration , the differentiation chain rule is used, and the parameter θ is updated as follows:

$$\theta \leftarrow \theta + \lambda \frac{\partial J(\theta)}{\partial \theta} \tag{16}$$

where λ is the learning rate.

5. Experiments

To validate the effectiveness of the proposed method CCNN, we conduct experiments respectively on two benchmarks: Google Snippets [3] and TREC [15].

Table 1: Data distribution of Google Snippets

Labels	**Training**	**Test**
Business	1,200	300
Computers	1,200	300
Culture-arts-entertainment	1,880	330
Education-Science	2,360	300
Engineering	220	150
Health	880	300
Politics-Society	1,200	300
Sports	1,120	300
Total	10,060	2,280

Table 2: Data distribution of TREC

Labels	**Training**	**Test**
DESC.	1,162	138
ENTY.	1,250	94
ABBR.	86	9
HUM.	1,223	65
NUM.	896	113
LOC.	835	81
Total	5,452	500

5.1. Experimental Setup

5.1.1. Datasets

Google Snippets. This dataset consists of 10,060 training snippets and 2,280 test snippets from 8 categories, as shown in Table 1. On average, each snippet has 18.07 words.

TREC. As demonstrated in Table 2, TREC contains 6 different question types, including LOC., NUM., ENTY. and so on. The training dataset consists of 5,452 labeled questions, and the test dataset consists of 500 questions.

Table 3: Details of publicly available embeddings

Embeddings	Senna[5]	GloVe[6]	Word2Vec
Training Corpus	Wikipedia	Wikipedia/Gigaword	Google News
Dimensionality	50	50	300
Size of Vocab.	130,000	400,000	3,000,000

5.1.2. Pre-trained Word Embeddings

To validate the robustness of the proposed architecture, we respectively initialize the lookup table with three different pre-trained word embeddings and conducted experiments, which are publicly available. The summaries of these word embeddings are listed in Table 3, and some descriptions are provided as follows:

Senna. *Semantic/syntactic extraction using a neural network architecture* is abbreviated to Senna, which is a software distributed by Collobert et al. [11]. Its word embeddings have been trained over Wikipedia for about 2 months. Senna also can be used for part-of-speech (POS) tags, name entity recognition (NER), semantic role labeling (SRL) and syntactic parsing (PSG).

GloVe. Pennington et al. [33] proposed an unsupervised learning algorithm for obtaining word vector representations, called GloVe, for Global Vectors, since the global corpus statistics are captured directly by the model. GloVe is essentially a log-bilinear method with a weighted least-squares objective, which is trained over a 6 billion token corpus. The corpus is constructed using Wikipedia2014 and Gigaword5, with a vocabulary of the top 400,000 most frequent words and a context window size of 10.

Word2Vec. The Word2Vec tool provides an efficient implementation of the continuous Bag-of-Words and Skip-gram architectures for computing vector representations of words. The pre-trained word embeddings were learned on part of Google News dataset, which contains 300-dimensional vectors for 3 million words and phrases. The phrases were obtained using a simple data-driven approach described in [9].

[5] http://ml.nec-labs.com/senna/

[6] http://nlp.stanford.edu/projects/glove/

Table 4: The classification accuracy of proposed method against other models(%)

Methods		Google Snippets	TREC
CCNN	Senna	84.0	95.1
	GloVe	85.3	**96.8**
	Word2Vec	**85.5**	95.9
TF-IDF+SVMs		62.6	94.3
Paragraph Vector+SVMs		61.9	75.8
LSTM		63.0	95.6
DCNN [18]		–	93.0
SVMS [34]		–	95.0
CNN-multichannel [19]		–	93.6
LDA+MaxEnt [3]		82.7	–
Multi-Topics+MaxEnt [2]		84.17	–

5.2. Results and discussions

In our framework, the out-of-vocabulary words in short texts are simply discarded, since they are often low-frequency tokens. The experimental results and discussions are as follows.

5.2.1. Compared with the state-of-the-art methods

In order to make strong comparisons, 8 popular methods are introduced as baselines. In experiments, we evaluate three of them on our benchmarks, including TF-IDF, Paragraph Vector and LSTM. For the rest of baselines, we directly introduce the publicly published experimental results as shown in Table 4. All the results in Table 4 are obtained under the same distributions of experimental data, as shown in Table 1 and Table 2. In the following, some brief introductions of these methods are given:

TF-IDF+SVMs. The statistics Term Frequency (TF) and Inverse Document Frequency (IDF) were calculated as features, and SVMs classifier was adopted.

Paragraph Vector. A unsupervised algorithm that can be used to learn fixed-length feature representations for sentences, paragraphs, or documents [12]. The model considers a paragraph as a general word token, which acts as a memory that remembers what is missing from the current context.

LSTM. The method is a variation of the standard LSTM model [24], which is composed of a single LSTM layer followed by an average pooling and a logistic regression layer. In this variant, the activation of a cells output gate does not depend on the memory cells state, which allows us to perform part of the computation more efficiently.

DCNN. Kalchbrenner et al. [18] proposed the DCNN to model sentences. In their work, wide convolution is utilized to extract local features, followed by dynamic k-max pooling operation to capture global and fixed-size feature vector.

SVMs. Uni-bi-trigrams, wh word, head word, POS, parser, hypernyms, and 60 hand-coded rules were used as features to train SVMs [34].

Two-Channel CNN. Two input channels were used to allow the employment of task-specific and static word embeddings simultaneously, which was improved by Kim [19] based on CNN.

LDA+MaxEnt. Phan et al. [3] proposed the method to discover hidden topics, from external Wikipedia corpus, using LDA to expand short texts. After feature expansion, MaxEnt classifier is used to make prediction.

Multi-topics+MaxEnt. Based on the work of Phan et al. [3], Chen et al. [2] leverage topics at multiple granularity to model short texts precisely.

With the same setup of experimental data, the comparisons of our method against the 8 baselines are demonstrated in Table 4. As a whole, our proposed method CCNN achieves the best performance. For benchmark TREC, our framework initialized using the three different word embeddings all outperform the introduced baselines, so the semantic representations in the second layer can extract useful features. Furthermore, when GloVe word embedding is employed, the highest classification accuracy 96.8% is obtained, as a result of that GloVe is trained over Wikipedia2014 and Gigaword5 globally. Thus, the GloVe word embedding is more general compared with the others.

However, when the word embedding induced by Word2Vec is used, we obtain the best result of 85.5% on the benchmark Google Snippets. The most important reason is that the Word2Vec embedding is learned over Google News. So, the benchmark Google Snippets and Word2Vec embeddings have consistent semantics. Moreover, the Word2Vec embeddings have higher dimension and larger vocabulary.

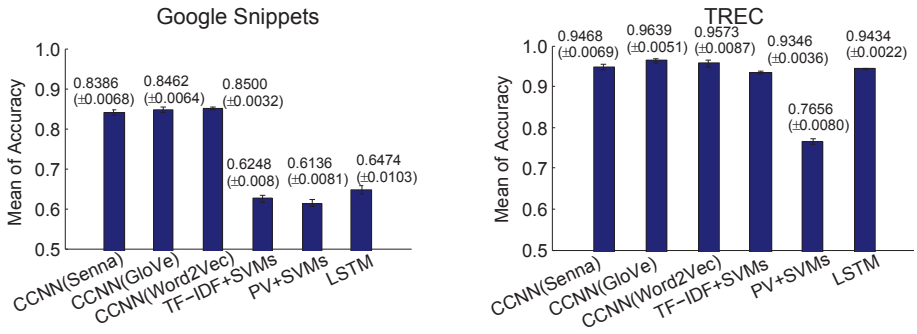

Figure 5: Experimental results (mean accuracy $\pm$ standard deviation) with 5-fold cross validation.

Table 5: The p-values of T-test over benchmark Google Snippets(*1.0e-3)

Methods	TF-IDF (+SVMs)	PV (+SVMs)	LSTM
CCNN (Senna)	0.000992	0.000383	0.001597
CCNN (GloVe)	0.001545	0.000395	0.004507
CCNN (Word2Vec)	0.000342	0.000393	0.002078

Table 6: The p-values of T-test over benchmark TREC

Methods	TF-IDF (+SVMs)	PV (+SVMs)	LSTM
CCNN (Senna)	0.0236	0.0174395*1.0e-5	0.2048
CCNN (GloVe)	0.0002	0.0441784*1.0e-5	0.0008
CCNN (Word2Vec)	0.0011	0.0210382*1.0e-5	0.0133

5.2.2. Statistical Significant Test

In order to demonstrate the significance of our method compared to baselines, we design 5-fold cross-validation experiments on two benchmarks. Different from the experimental setup in Section 5.2.1, we respectively mix up the original training samples and test samples of Google Snippets and TREC shown in Table 1 and Table 2. Then, we divide each mixed data set into 5 parts equally and conduct cross validation experiments. Similar to Table 4, we reproduce three of the baselines, which include TF-IDF+SVMs, Paragraph Vector+SVMs (abbr. to PV+SVMs) and LSTM. The com-

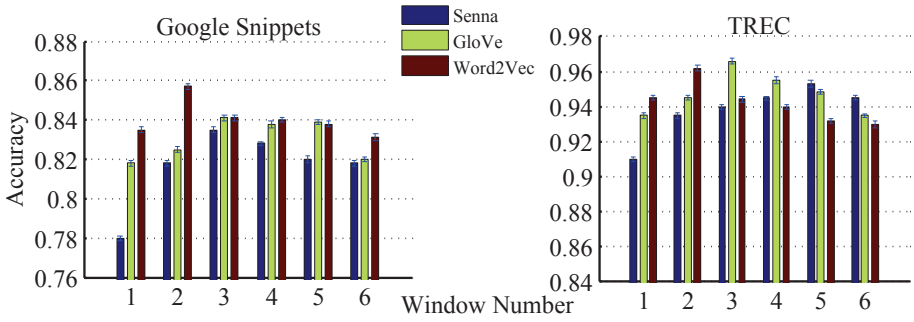

Figure 6: The number of window matrices for multi-scale semantic expansion.

parisons are demonstrated in Figure 5.

For the experimental results in Figure 5, we conduct T-test [35], and the p-values are shown in Table 5 and Table 6 respectively. From Table 5, we can observe that all p-values< 0.01, which indicate that our methods outperform baselines over benchmark Google Snippets obviously. However, the corresponding p-value> 0.05 of our method CCNN(Senna) against LSTM in Table 6, which implies that the two methods have obtained close performance over benchmark TREC.

Holistically, comparing the results in Figure 5, as well as p-values in Table 5 and Table 6, it is clear that PV+SVMs approach does poorly over two benchmarks, and although LSTM achieves almost the same results as our CCNN does for benchmark TREC, it does not do well in the dataset of Google snippets. While our CCNN approach performs consistently well across the two benchmarks. We guess that an explanation for the consistent good performance of our CCNN is that we use NWEs derived from semantic cliques for expanding raw text, as well as extra knowledge introduced by pre-trained word embeddings.

5.2.3. Effect of Hyper-parameters

As shown in Figure 4, in order to obtain the representations of semantic units with multi-scale, multiple window matrices with increasing width are used. For instance, if m window matrices are employed, their width are ranging from 2 to $m + 1$, which act as from 2-gram to $m + 1$-gram. Additionally, the projected matrix can be viewed as

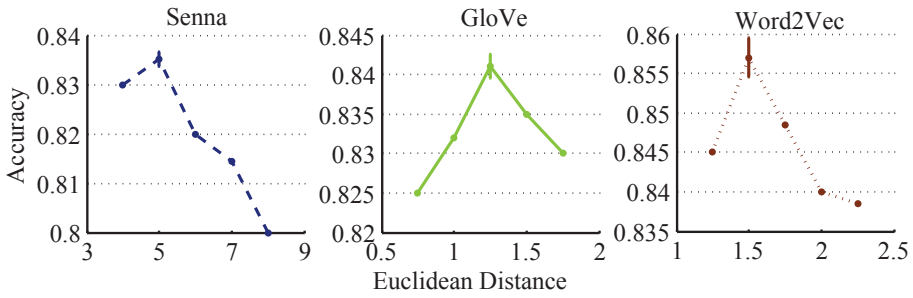

Figure 7: The influence of threshold preset by Euclidean distance.

Table 7: The vocabulary coverage rate of word embeddings on two benchmarks

DataSets	Google Snippets	TREC
$\|Vocab.\|$	30616	7065
Senna	62.5%	76.2%
GloVe	68.1%	81.3%
Word2Vec	73.3%	84.6%

unigram. Thus the proposed architecture can fully explore the contexture information of short texts to alleviate the negative effect of their short length in representations. The experimental results with respect to variable m are demonstrated in Figure 6. For benchmark Google snippets, the highest classification accuracy was achieved when the window width is 2. Meanwhile, we obtained the best result on TREC when the window width is 3. Then we can conclude that the small size of window may result in the loss of some critical information that induce the ambiguous phrase composition, whereas the window with large size may generate noise.

As described in Section 4.1.1, the representations of semantic units are induced by additive composition. However, compared with their precise results, the embedding offsets may produce, as shown in Equation (1) and Equation (2). The representations of semantic units and the word embeddings are vectors with equal dimension. Therefore, we discover the NWEs of semantic units in the embedding spaces to decrease the offsets. For each semantic unit, the chosen NWEs should satisfy the pre-set distance threshold d that limits the euclidean distance between them. The experimental results,

over benchmark Google snippets in terms of hyper-parameter d, are shown in Figure 7. We can find that when d is too small, only a few of NWEs can be available. However, when d is too large, many unrelated NWEs are enrolled. Furthermore, the optimized threshold d is variable when the initialization of lookup table is different.

From Figure 6 and Figure 7, we can also find that the performance of our method varies with the different initialization of lookup table utilizing the three pre-trained word embeddings. The parameters of the word embeddings described in Table 3, such as training techniques, corpus, dimension of word embedding, size of vocabulary, and the vocabulary coverage rate of three word embeddings on the two datasets demonstrated in Table 7 are the factors which affect classification accuracy. The impacts of other hyper-parameters like the number and size of the feature detectors in convolutional layer, and the variable K in K-max pooling layer are beyond the scope of this paper.

6. Conclusion

In this paper, we proposed a novel semantic hierarchy for short texts modeling and classification. The pre-trained words embeddings are used to initialize the lookup table, which introduce extra knowledge and enable us to measure words affinity by computing the Euclidean distance between two vector representations. The additive composition method is utilized to compute multi-scale semantic units for short texts expansion. In the embedding spaces, similar words are grouped together that help learning algorithms to achieve better performance. Experimental results on open benchmarks validated the effectiveness of the proposed method. Future improvements can be obtained by supervised feature down-sampling, task-specific embeddings learning, and embedding affinity measurement in vector spaces.

Acknowledgement

This work is supported by the National Natural Science Foundation of China (No. 61203281, No. 61303172, No. 61403385) and Hundred Talents Program of Chinese Academy of Sciences (No. Y3S4011D31).

References

[1] B. Sriram, D. Fuhry, E. Demir, H. Ferhatosmanoglu, M. Demirbas, Short text classification in twitter to improve information filtering, in: Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2010, pp. 841–842.

[2] M. Chen, X. Jin, D. Shen, Short text classification improved by learning multi-granularity topics, in: Proceedings of the 22nd International Joint Conference on Artificial Intelligence, 2011, pp. 1776–1781.

[3] X.-H. Phan, L.-M. Nguyen, S. Horiguchi, Learning to classify short and sparse text & web with hidden topics from large-scale data collections, in: Proceedings of the 17th International Conference on World Wide Web, ACM, 2008, pp. 91–100.

[4] D. M. Blei, A. Y. Ng, M. I. Jordan, Latent dirichlet allocation, Journal of Machine Learning Research 3 (2003) 993–1022.

[5] M. Sahami, T. D. Heilman, A web-based kernel function for measuring the similarity of short text snippets, in: Proceedings of the 15th International Conference on World Wide Web, ACM, 2006, pp. 377–386.

[6] X. Yan, J. Guo, Y. Lan, X. Cheng, A biterm topic model for short texts, in: Proceedings of the 22nd International Conference on World Wide Web, ACM, 2013, pp. 1445–1456.

[7] Y. Bengio, R. Ducharme, P. Vincent, C. Janvin, A neural probabilistic language model, Journal of Machine Learning Research 3 (2003) 1137–1155.

[8] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, arXiv preprint arXiv:1301.3781.

[9] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Advances in Neural Information Processing Systems, 2013, pp. 3111–3119.

[10] T. Mikolov, W.-t. Yih, G. Zweig, Linguistic regularities in continuous space word representations., in: HLT-NAACL, 2013, pp. 746–751.

[11] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa, Natural language processing (almost) from scratch, Journal of Machine Learning Research 12 (2011) 2493–2537.

[12] Q. V. Le, T. Mikolov, Distributed representations of sentences and documents, arXiv preprint arXiv:1405.4053.

[13] Z. Li, J. Liu, Y. Yang, X. Zhou, H. Lu, Clustering-guided sparse structural learning for unsupervised feature selection, Knowledge and Data Engineering, IEEE Transactions on 26 (9) (2014) 2138–2150.

[14] A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, Science 344 (6191) (2014) 1492–1496.

[15] X. Li, D. Roth, Learning question classifiers, in: Proceedings of the 19th International Conference on Computational Linguistics, Association for Computational Linguistics, 2002, pp. 1–7.

[16] G. Zhou, Y. Liu, F. Liu, D. Zeng, J. Zhao, Improving question retrieval in community question answering using world knowledge, in: Proceedings of the 23rd International Joint Conference on Artificial Intelligence, 2013, pp. 2239–2245.

[17] A. Mnih, Y. W. Teh, A fast and simple algorithm for training neural probabilistic language models, arXiv preprint arXiv:1206.6426.

[18] N. Kalchbrenner, E. Grefenstette, P. Blunsom, A convolutional neural network for modelling sentences, arXiv preprint arXiv:1404.2188.

[19] Y. Kim, Convolutional neural networks for sentence classification, arXiv preprint arXiv:1408.5882.

[20] D. Zeng, K. Liu, S. Lai, G. Zhou, J. Zhao, Relation classification via convolutional deep neural network, in: Proceedings of the 25th International Conference on Computational Linguistics, 2014, pp. 2335–2344.

[21] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, C. Potts, Recursive deep models for semantic compositionality over a sentiment treebank, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, Vol. 1631, 2013, p. 1642.

[22] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R. R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, arXiv preprint arXiv:1207.0580.

[23] M. Sundermeyer, R. Schlüter, H. Ney, Lstm neural networks for language modeling., in: INTERSPEECH, 2012.

[24] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Computation 9 (8) (1997) 1735–1780.

[25] I. Sutskever, O. Vinyals, Q. V. Le, Sequence to sequence learning with neural networks, in: Advances in Neural Information Processing Systems, 2014, pp. 3104–3112.

[26] A. Graves, A.-R. Mohamed, G. Hinton, Speech recognition with deep recurrent neural networks, in: Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, IEEE, 2013, pp. 6645–6649.

[27] L. Shang, Z. Lu, H. Li, Neural responding machine for short-text conversation, arXiv preprint arXiv:1503.02364.

[28] J. Mitchell, M. Lapata, Composition in distributional models of semantics, Cognitive Science 34 (8) (2010) 1388–1429.

[29] A. Yessenalina, C. Cardie, Compositional matrix-space models for sentiment analysis, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2011, pp. 172–182.

[30] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.

[31] J. Turian, L. Ratinov, Y. Bengio, Word representations: a simple and general method for semi-supervised learning, in: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, 2010, pp. 384–394.

[32] J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, Journal of Machine Learning Research 12 (2011) 2121–2159.

[33] J. Pennington, R. Socher, C. D. Manning, Glove: Global vectors for word representation, Proceedings of the Empiricial Methods in Natural Language Processing 12 (2014) 1532–1543.

[34] J. Silva, L. Coheur, A. C. Mendes, A. Wichert, From symbolic to sub-symbolic information in question classification, Artificial Intelligence Review 35 (2) (2011) 137–154.

[35] D. H. Johnson, The insignificance of statistical significance testing, Journal of Wildlife Management (1999) 763–772.