IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON SCALABLE DEEP LEARNING FOR BIG DATA

Received November 6, 2019, accepted November 22, 2019, date of publication November 26, 2019,
date of current version December 11, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2955924

Large-Scale Text Classification Using Scope-Based Convolutional Neural Network: A Deep Learning Approach

JIAYING WANG[1], (Member, IEEE), YAXIN LI[1], JING SHAN[1], JINLING BAO[2], CHUANYU ZONG[3], AND LIANG ZHAO[3], (Member, IEEE)

[1]Big Data Management and Analysis Laboratory of Urban Construction, Shenyang Jianzhu University, Shenyang 110168, China
[2]Computer Science Faculty, Baicheng Normal University, Baicheng 137000, China
[3]College of Computer Science, Shenyang Aerospace University, Shenyang 110136, China

Corresponding authors: Jing Shan (shanjing@sjzu.edu.cn) and Chuanyu Zong (zongcy@sau.edu.cn)

This work was supported in part by the National Natural Science Foundation for Young Scientists of China under Grant 61702346, Grant 61702345, Grant 61701322, and Grant 61802268, in part by the Key Projects of Liaoning Natural Science Foundation under Grant 20170540700, and in part by the Liaoning Provincial Department of Education Science Foundation under Grant JYT19052.

ABSTRACT Text classification is one of the most important and typical tasks in Natural Language Processing (NLP) which can be applied for many applications. Recently, deep learning approaches has shown their advantages in solving text classification problem, in which Convolutional Neural Network (CNN) is one of the most successful model in the field. In this paper, we propose a novel deep learning approach for categorizing text documents by using scope-based convolutional neural network. Different from window-based CNN, scope does not require the words that construct a local feature have to be contiguous. It can represent deeper local information of text data. We propose a large-scale scope-based convolutional neural network (LSS-CNN), which is based on scope convolution, aggregation optimization, and max pooling operation. Based on these techniques, we can gradually extract the most valuable local information of the text document. This paper also discusses how to effectively calculate the scope-based information and parallel training for large-scale datasets. Extensive experiments have been conducted on real datasets to compare our model with several state-of-the-art approaches. The experimental results show that LSS-CNN can achieve both effectiveness and good scalability on big text data.

INDEX TERMS Text classification, deep learning, convolutional neural network, scope-based convolution, local feature.

I. INTRODUCTION

The task of text classification (a.k.a. text tagging, text filtering or text categorization) is a process of categorizing a text document into one or multiple predefined categories based on the content. Concretely, the target is to build a classifier which takes a text document as an input, then automatically assigns relevant labels according its content. These text documents can be emails, comments, or movie reviews. Accordingly the labels can be spam/non-spam, positive/negative/neutral or review scores.

Text classification plays an important role in Natural Language Processing (NLP). It is widely adopted in many applications. For example, most of news services today needs to automatically organize a large volume of new articles every single day [1]. All the modern mail services provide a function to determine either a mail is a junk mail automatically [2]. Other applications include sentiment analysis [3], topic modelling [4], language translation [5], and intent detection [6], etc.

Text classification is a challenge problem. The sparse, high dimensional, and existence of irrelevant or noisy characteristics of text make it a non-trivial job to develop a good classifier for large-scale text data. Because of its importance and challenge, a lot of methods range from traditional feature engineering classification methods [7]–[10] with hand-crafted features to emerging deep learning methods [1], [11]–[15] have been proposed to solve the problem.

CNN is one of the most successful deep feed-forward network. Just like the other deep learning models, CNN is

The associate editor coordinating the review of this manuscript and approving it for publication was Liangxiu Han.

composed of multiple processing layers including one input layer, multiple hidden layers and one output layer. It utilize simple raw text data as the input and gradually transform it into higher and more abstract representation. Among all the hidden layers used in CNN, convolutional layer is the most special one. It is also the key feature to make the approach success. Except the convolution layer, there are also pooling layers (a.k.a subsampling layers) and several fully connected layers, which are widely used in a traditional feedforward neural network.

Traditional convolutional layer conduct convolution operation based on a sliding window. In this way, all the local informations within a semantic block of the certain length can be captured. Each neuron in a neural network computes an output value by applying a linear or non-linear function to the input values, which come from the the receptive field in the previous layer. The function is determined by a filter, which is a vector of weights and a bias. As different semantic blocks may have different lengths. Existing methods often utilize a group of sliding windows with different lengths to capture the different local informations [11], [12], [16], [17].

Although CNN is a good choice for text classification, it still has potential for improvement. First, to achieve a good result for a text classification task, a group of window sizes needs to be specified. The problem is that there are a lot of combinations to select. It is not an easy job for a user to discover a good setting for the optimal combination of window sizes. Suppose that the largest value for a length that we will use is n, any combination of values that is less than or equal to n is a valid combination. Thus there are 2^n combinations for users to select. A user needs a lot of experimental studies to achieve the full capacity of CNN for a real world text classification task. And different datasets may need different settings to achieve the optimal results.

Another important issue is that traditional CNN cannot capture the complicated deeper local features of text data. Since sliding window approach consider only the continuous language semantics. While in many languages (e.g. Slavic languages, Chinese and German languages), parts of phrases could be separated by several other words. Based on studies in [18], discontinuities is a common problem in NLP applications. Even in English, a rigid language, discontinuity can happen (e.g. ''She was, I believe, lying.''). A important thing is that the discontinuities destroy the direct connection between words or phrases in a text data. In which cases, choosing any window size cannot accurately capture the local features.

To this end, we propose a new convolutional neural network to solve all these problems. A user only need to set the largest length of a semantic block, which is much easier compared with selecting in a combination of window lengths. And it can effectively capture the complicated local features including discontinuities occur in the text context. The approach utilizes a flexible convolution operation, which is based on a concept called scope. The scope only impose restriction on the distance between elements (i.e. words or phrases) in a text document. The idea comes from a intuition that if an element x has a strong connection to another element y, the distance between x and y should not be too far away. We use term scope to represent the maximum distance that we allow.

We also propose several optimized techniques to boost the process of discovering local feature and increase the scalability of the methods. Together we propose a large-scale scope-based convolutional neural network (LSS-CNN) for text classification problem. The approach can not only effectively capture complicated local features, but also reduce human effect to select a good setting. It can discover the deeper local information that hides within an abstract representation of a variable length text or a discontinuous text structure. We also propose an aggregation optimization, and pooling mechanism to merge the local features and select the most signification one to fit in the next layers. The new approach can be taken as a generalization of traditional CNN, since if a good feature is discovered by sliding windows, it can also be captured by LSS-CNN. And LSS-CNN can discover better and deeper local features.

In order to show the effectiveness and efficiency of this method, we conduct extensive experiments to study our proposed approach. We also compare it with several state-of-the-art methods on the eight real datasets. The experimental result show that LSS-CNN can achieve high effectiveness and good scalability on big text corpora.

In summary, the contributions of this paper include:

- We propose LSS-CNN, a new large-scale scope-based convolutional neural network model. Based on scope convolution operation, pooling mechanism, it can effectively capture complicated local features of text data and reduce human effort to select a good setting.
- We propose several optimizations to boost the classification process, which make it possible to handle large scale text data, and achieve more efficient text classification on big text corpora.
- We conduct extensive experimental study on several real datasets, and we conduct a comparison with existing state-of-the-art approaches to demonstrate the effectiveness of the proposed method.

II. RELATED WORK

A. TEXT CLASSIFICATION

The study of text classification has a long history, which can be mainly separated into three stages.

In the first stage, text classification is achieved by creating a set of hand-crafted linguistic rules [19], [20]. The good point of these approach is that since rules are human comprehensible, they can be improved over time. However, creating rules requires extensive domain knowledge [21] and it is still time-consuming to create rules for a complicated text classification tasks. And different rules may be conflicted with each other. Thus these method cannot scale well.

In the second stages, machine learning methods are adopted in text classification. Instead of using hand-crafted

rules, machine learning based methods [7]–[10] try to learn classification rules from pre-labelled examples. In this way, a machine learning algorithm can learns the different associations between pieces of text and some particular tags automatically [22]. Thus it can discover the rules without human efforts, which can achieve more accurate than hand-crafted rules methods, especially on complicated tasks. Bayesian classifier [23], decision tree [24] and support vector machine [25] are widely used machine learning methods.

Although machine learning approach can achieve relative good result, it still needs to carefully extract features for text data. The extracted features play a key role in the effectiveness of a system [26]. Bag-of-words (i.e. BOW) and n-grams representation are mostly used text representation techniques [21]. The frequency of terms (or words) or the variant such as term frequency-inverse document frequency (TF-IDF), information gain or entropy, or mutual information x^2 statistic, generalized singular value decomposition of values can be used as features [21]. Based on those techniques, a text document is mapped into a high dimensional feature vector. It is obviously, these approaches loses all the word order information and only retaining the frequency of the terms in the document.

To solve the problem of text representation and take advantage of word order information, deep learning methods [1], [11]–[15] are proposed in the third stage. These methods are inspired by how the human brain works. It can achieve high accuracy with less need of engineered features. Thus it help reduce human effort to extract features for the classifiers. Recently, models based on deep learning networks have become increasingly popular [27]–[29].

B. DEEP LEARNING NEURAL NETWORK

The most popular deep learning architecture used in text classification is Convolutional Neural Networks (CNN) because its capability to capture local information [11]. CNN was first invented for solving computer vision problem. It is now the dominant approach for feature extraction from audio, video or textual data. The idea of using sliding windows to conduct convolution for text data is first proposed in [30]. It was further improved in [11] by adopting filters with multiple window sizes. Reference [12] use Dynamic k-Max Pooling to conduct semantic modelling of sentences. [16] use n-gram features to exploit interactions between words. Reference [17] improved the performance on short text using a taxonomy knowledge base. Reference [31] deepen the network without dramatically increasing the computational cost. Reference [32] proposed a simple linear model to conduct fast text classification. Reference [1] proposed a recurrent convolutional neural network, which utilizes a recurrent structure to capture contextual information. Reference [15] use dense connections between different convolution layers to extract variable-size features.

Except CNN, RNN is also an important deep learning model. Reference [33] proposed a RNN based multi task learning method, which jointly learns across multiple related tasks to solve insufficient training data issue. Reference [34] utilized a long short-term memory recurrent neural network (LSTM) combined with CNN phrase representation for text classification. Reference [35] utilize Tree-structure LSTM, a variant of RNN to improve semantic representation. One important problem in deep learning is how to represent text data. Word embedding is current the best solution.

C. WORD EMBEDDING

The target of word embedding is to map a word from high dimensional text data to a real number vector within a continuous vector space of much lower dimensions. It is an important method to solve the sparsity problem of the text data (a.k.a curse of dimensionality). Currently, the most successful word embedding toolkits are word2vec [36] and GloVe [37]. There are also several other variant word embedding approaches. In [38], they use a sparse Shifted Positive PMI word-context matrix to represent words. In [39] sentiment-specific word embedding was learnt from the positive and negative emoticons collected in massive distant-supervised tweets.

Recently, the development of deep learning neural networks and word embedding have brought new inspiration to various NLP tasks. In [40], researchers found that most of parameters in a model can be pre-trained, such that only one additional output layer is needed for a wide range of tasks, such as text classification, next sentence prediction, question answering and language inference. Our work is also based on pre-trained word embedding.

III. PROBLEM DEFINITION AND PRELIMINARIES

A. PROBLEM DEFINITION

The problem of text classification is defined as follows. Given a set of text documents $D = \{d_1, d_2, \ldots d_N\}$, such that each document is tagged with a set of labels. Each tag is drawn from a set of x different labels $\{l_1, l_2, \ldots l_x\}$. The target is to construct a classification model, which relates the text documents to their labels. Thus for a given test document, the trained model can predict the set of labels for that instance. By performing the prediction, we actually predict a probability value for each label, such that if the result is larger than 0.5 will be taken as accept label. Notice that in this work, we consider the assignment of multiple labels. It can be easily modified to solve single label classification problem (i.e. There is only one label for each text document), in which only one label with the maximum probability value will be assigned to the test instance.

B. PRELIMINARIES

1) DOCUMENT REPRESENTATION

We represent a text document as a sequence of words, that $d = w_1\, w_2 \ldots w_{|d|}$, in which w_i represents a word located at position i in the document d. A subsequence of d is a new sequence of words, which are derived from d by deleting some words without modifying the order of the remaining

words. We use $sub(d)$ to represent the subsequence. Consider a document $d = w_1\ w_2\ w_3\ w_4$, $sub(d) = w_1\ w_2\ w_3$ is a subsequence. We use $C_{sub}(d)$ to represent all the subsequences of d. In the example that $d = w_1\ w_2\ w_3\ w_4$, $C_{sub}(d) = \{\emptyset, w_1, w_2, w_3, w_1\ w_2, w_2\ w_3, w_1\ w_3, w_1\ w_2\ w_3\}$. There are $2^{|d|}$ subsequences for a text document d. We utilize $dis(w_i, w_j)$ to represent the distance from word w_i to word w_j, such that $dis(w_i, w_j) = j - i$.

2) DOCUMENT PADDING

Notice that text documents do not always have the same length, which brings difficulties to fit variable-length text documents to a deep learning model. Thus we first do an operation to transform the variable-length text documents to a fixed-length representation. We achieve that by padding documents to the maximum document length. Concretely, we represent each document with n words, in which n is the maximum length of document in D, i.e. $n = max(|d|), d \in D$. We pad those text documents with length less than n by adding word NULL to the end. Thus that after the padding all documents have n words.

3) WORD EMBEDDING

The target is to map each word into a pre-trained vector. By using an existing embedding toolkit (e.g. word2vec or GloVe), a word w_i is transformed to a d-dimensional vector $v_i \in \mathbb{R}^d$. In this way, we map the representation of a document from a sequence of words to a sequence of fixed length vector (i.e. a matrix with fixed shape $n \times d$, in which n is the number of pre-trained word vectors, and d is the dimension of each word vector). Notice that we use 0^d, a d-dimensional zero vector, to represent the special word NULL. After the transform, the whole document set D can be represented as a matrix of $N \times n \times d$. All the subsequence and k-scope sequence based on word representation are also applied on the vector representation of document. i.e. $d = v_1\ v_2 \ldots v_n$. For the sake of simplicity, if we do not emphasize it is based word representation, we mean the vector representation hereinafter.

4) VECTOR CONCATENATION

We use $v_i \oplus v_j$ or $\oplus(v_i, v_j)$ to represent the new vector by concatenating the word vectors v_i and v_j, such that $v_i \oplus v_j$ is a $2 \times d$ dimensional vector, which is called the concatenation of v_i and v_j. Similarly, we can get a $3 \times m$ dimensional vector by concatenating three word vectors $v_i \oplus v_j \oplus v_k$. For the sake of consistency, we let $\oplus(v_i) = v_i$. We use $\oplus(d)$ to represent the concatenation of all the word vectors in document d. And similarly $\oplus(sub(d))$ represents the concatenation of all the word vectors in the subsequence $sub(d)$.

IV. SCOPE-BASED CONVOLUTIONAL NEURAL NETWORK

In this section, we present scope-based convolutional neural network. We first explain the concept of scope.

Scope: The concept of scope is actually represent a constraint of distance. We call word v_i and word v_j satisfy k-scope constraint iff $dis(v_i, v_j) \leq k$. For each position i, we use a definition called k-scope sequence to represent an ordered sequence of word vectors that satisfy the k constraints with v_i. The formal definition is given in Definition 1. The idea of k-scope is based on intuition that two words with strong connects should not be too far away.

Definition 1 (k-Scope Sequence): Consider a position i in the text document d, the k-scope sequence of position i is a word vector sequence S_i^k, such that each word $v_j \in S_i^k$ satisfies $0 \leq dis(v_i, v_j) \leq k$. And for each $v_j \in S_i^k$ and $v_{j'} \in S_i^k$, v_i precedes v_j iff $i < j$.

Consider an example $d = v_1\ v_2\ v_3\ v_4\ v_5$, $k = 1$ and $i = 3$, we have $S_3^1 = v_2\ v_3\ v_4$.

For CNN, convolution operation is the key operation. Next we present how to utilize the concept of k-scope to conduct convolution operation.

A. K-SCOPE CONVOLUTION

1) k-SCOPE CONCATENATION SET

Our convolution operation is based on k-scope sequence. We call it k-scope convolution, which is used to capture the local feature within a k-scope sequence. Specifically, a k-scope convolution operation is conducted on each position in a text document d. And the k-scope convolution operation at position i takes responsibility to compute the local feature within a region that is at most k distance from the position i. To compute the deeper local feature, we consider all the concatenation that within k-scope sequence of v_i (i.e. all the subsequence of S_i^k). The formal definition of k-scope concatenation set is given in Definition 2.

Definition 2 (k-Scope Concatenation Set): Consider a position i in document d, the k-scope concatenation set of position i is a vector set $\uplus_i^k$, which contains concatenations of all the subsequences of k-scope sequence of v_i, i.e. $\uplus_i^k = \{\oplus(x) \mid x \in C_{sub}(S_i^k)\}$. The whole k-scope concatenation set is the set union of all the k-scope concatenation sets, i.e. $\uplus^k = \uplus_1^k \cup \uplus_2^k, \ldots \cup \uplus_n^k$.

Consider the example $d = v_1\ v_2\ v_3\ v_4\ v_5$, let $k = 1$, the k-scope concatenation set for position 2 is $\uplus_2^1 = \{v_1, v_2, v_3, v_1 \oplus v_2, v_1 \oplus v_3, v_2 \oplus v_3, v_1 \oplus v_2 \oplus v_3\}$. In this way, for a convolutional layer of the network, we needs $n \times m \times (2^{2k+1} - 1)$ neural cells.

Notice that this representation of local features is not good enough, as there is redundancy for the convolution operation. Consider the k-scope concatenation set for position 3 is $\uplus_3^1 = \{v_2, v_3, v_4, v_2 \oplus v_3, v_2 \oplus v_4, v_3 \oplus v_4, v_2 \oplus v_3 \oplus v_4\}$. There are three redundancy computation v_2, v_3 and $v_2 \oplus v_3$ for $\uplus_3^1$ and $\uplus_2^1$.

We can easily eliminate the redundancy by fixing the start position. We call it k-scope concatenation set with fixed start position, the formal definition is given in Definition 3.

Definition 3 (k-Scope Concatenation Set With Fixed Start Position): Given a position i in document d, the k-scope concatenation set with fixed start position of i is a vector set $\widehat{\uplus}_i^k$, which contains concatenations of all the subsequences of k-scope sequence of v_i and the subsequences start with v_i, i.e.

$\widehat{\uplus}_i^k = \{\oplus(x) \mid x \in C_{sub}(S_i^k) \wedge x_1 = v_i\}$. *The whole k-scope concatenation set with fixed start position is the set union of all the k-scope concatenation sets with fixed start position, i.e.* $\widehat{\uplus}^k = \widehat{\uplus}_1^k \cup \widehat{\uplus}_2^k, \ldots \cup \widehat{\uplus}_n^k$.

For example, let $k = 2$, the k-scope concatenation set with fixed start position for position 2 is $\widehat{\uplus}_2^2 = \{w_2, w_2 \oplus w_3, w_2 \oplus w_4, w_2 \oplus w_3 \oplus w_4\}$.

Now, we show that k-scope concatenation set with fixed start position is good enough to replace k-scope concatenation set. We prove the k-scope concatenation set with fixed start position satisfies Theorem 1.

Theorem 1: For a text document, the k-scope concatenation set with fixed start position $\widehat{\uplus}^k$ *covers the* $(\lfloor k/2 \rfloor)$*-scope concatenation set* $\uplus^{\lfloor k/2 \rfloor}$*, but do not have redundant elements.*

Proof: Consider k is an even number, $(\lfloor k/2 \rfloor)$-scope concatenation set $\uplus^{\lfloor k/2 \rfloor}$ contains all the subsequences of $\lfloor k/2 \rfloor$-scope sequence with the max length of $\lfloor k/2 \rfloor \times 2 + 1$. While the k-scope concatenation set with fixed start position $\widehat{\uplus}^k$ contains all the subsequences of k-scope sequence with the max length of $k+1$. Since $k+1 = \lfloor k/2 \rfloor \times 2 + 1$, the two set are equivalent. Notice that the elements in $\uplus_i^{\lfloor k/2 \rfloor}$ are separated in many $\widehat{\uplus}_j^k$ sets, in which $i - \lfloor k/2 \rfloor \leq j \leq i + \lfloor k/2 \rfloor$. Next consider k is an odd number, now we have $k + 1 > \lfloor k/2 \rfloor \times 2 + 1$. In this case, the k-scope concatenation set with fixed start position covers the $(\lfloor k/2 \rfloor)$-scope concatenation set. Since for any element in $\widehat{\uplus}_i^k$, we do not have redundant elements, and for any two different element $x \in \widehat{\uplus}_i^k$ and $y \in \widehat{\uplus}_j^k$, x starts with v_i and y starts v_j. They can not be the same. Thus the k-scope concatenation set with fixed start position do not have redundant elements. Thus the theorem holds. □

For the sake of simplicity, hereinafter we use k-scope concatenation set to represent the k-scope concatenation set with fixed start position.

2) COMPUTE k-SCOPE CONVOLUTION SET

The computation of k-scope concatenation set can be efficiently achieved using a dynamic programming algorithm as depicted in Alg. 1. For each position i, we concatenate the word vector v_j that satisfies k-scope constraint with v_i into the current temporary concatenation set. Initially we set each temporary concatenation set $\widehat{\uplus}_i^k$ to be empty.

Consider $k = 2$, $d = v_1\ v_2\ v_3\ v_4\ v_5$. To compute the k-scope concatenation set $\widehat{\uplus}_1^2$, we first consider the first word vector v_1. Since the current concatenation set is empty, we directly add it into the set. Then the current set is $\widehat{\uplus}_1^2 = \{v_1\}$. Next we consider the second element in S_1^2, v_2. Since currently there is an element v_1 in the concatenation set, we can concatenate it with v_2, which generate $v_1 \oplus v_2$, and the concatenation set is updated as $\widehat{\uplus}_1^2 = \{v_1, v_1 \oplus v_2\}$. Next we consider the third element v_3. By concatenating with existing concatenation set $\{v_1, v_1 \oplus v_2\}$, we generate two new elements $v_1 \oplus v_3$ and $v_1 \oplus v_2 \oplus v_3$. Similarly we add them to the current concatenation set. As v_3 is the last element, we just finish the process. Finally the method generates the result $\widehat{\uplus}_1^2 = \{v_1, v_1 \oplus v_2, v_1 \oplus v_3, v_1 \oplus v_2 \oplus v_3\}$. In this way, for the first convolutional layer of the network, we needs $n \times m \times 2^k$ nodes.

Algorithm 1 COMPUTEKSCOPECONVSET(d, k)

Input: d: a text document
k: scope
Output: $\widehat{\uplus}^k$: k-scope set

```
⊎̂^k ← ∅ ;
for i ← 1 to |d| do
    ⊎̂_i^k ← ∅ ;
    for j ← i to min(i + k, |d|) do
        t ← |⊎̂_i^k| ;
        if t = 0 then
            ⊎̂_i^k .add(d_j) ;
        else
            /* Concatenate word vector v_j
               with elements in ⊎̂_i^k      */
            for x ← 1 to t do
                ⊎̂_i^k .add(⊎̂_i^k [x] ⊕ v_j) ;
    ⊎̂^k .add(⊎̂_i^k) ;
return ⊎̂^k ;
```

We use $f(\cdot)$ to represent an active function utilized to generate a local feature for each element (i.e. concatenated vector) in the k-scope concatenation set. The output of a element $x \in \widehat{\uplus}_i^k$ is given in Equation 1. The activation function is used to adjust the output of convolutional layer. A lot of non-linear activation function can be used, such as sigmoid, tanh, and ReLU. Based on that, we get an active value.

$$f(x) = f(x, \omega) = activation(\sum_i x_i \times \omega_i + b) \qquad (1)$$

In the equation, we use ω to represent the weights on the element x. Take $v_1 \oplus v_2 \oplus v_3$ for example, the ω is a vector with $3 \times m$ dimensions, which is the length of concatenated vector x. We add another weight b for a bias weight. For the sake of simplicity, we use $f(x)$ instead of $f(x, \omega)$ which omits the ω hereinafter.

The example of 2-scope convolution is illustrated in Figure 1. For the the convolutional layer, the input is the document, which is a sequence of word vectors $v_1\ v_2 \ldots v_n$. For each position i, there is a convolution cell (i.e. nerve cell) C_i which conduct a convolution operation base on word vector $v_i v_{i+1} \ldots v_{i+k}$ and generate a group of active values (e.g. $f(v_2), f(v_2 \oplus v_3), f(v_2 \oplus v_4), f(v_2 \oplus v_3 \oplus v_4)$) as the output, which is the local features generated from the input document.

B. AGGREGATION OPTIMIZATION

Based on k-scope convolution technique, the method will firstly generate a group of features for each position i.

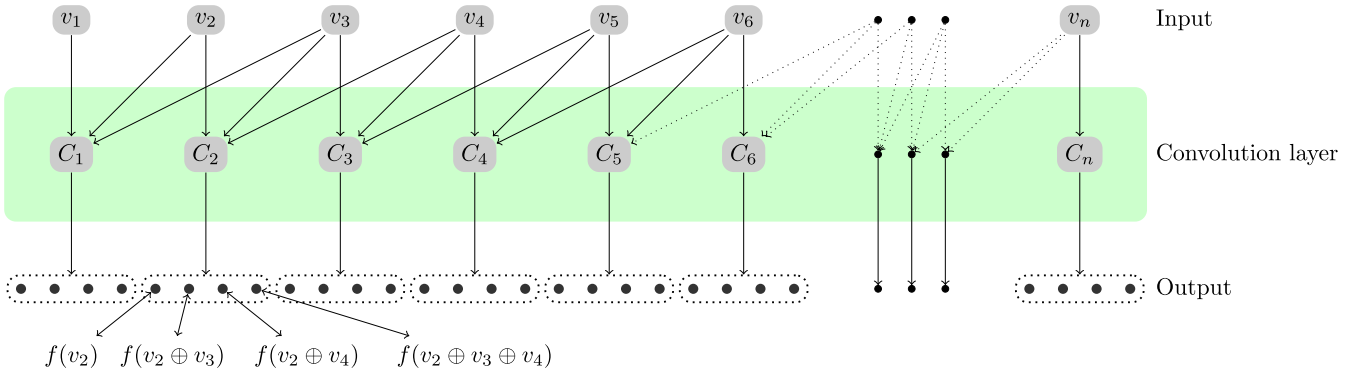


FIGURE 1. (Better viewed in color) Intuitive illustration of a k-scope convolution layer. In this example, we show the input and output for the 2-convolutional layer. The input part of the layer is a document d, which is a sequence of word vectors v_1 $v_2 \ldots v_n$. After the convolution operation, 2-scope convolutional features are generated as the output, which contains a group of activate values, which is computed based on activate function (e.g. $f(v_2), f(v_2 \oplus v_3), f(v_2 \oplus v_4), f(v_2 \oplus v_3 \oplus v_4)$).

For k-scope convolutional method, we will generate 2^k features for a position i. Thus, totally we will generate $n \times 2^k$ features. Consider the example in Fig. 1, that $k = 2$, after the convolutional operation, we will generate $n \times 2^k = 4 \times n$ features.

By increasing k in the k-scope convolutional operation, we can discover deeper local information in the text document. However, one problem of increasing k is that we have to take larger space to hold the output features. In the given example, 4 times larger feature space. What's more, if we connect it to another convolutional layer, the number of features will dramatically increase to 16 times. Thus it will exponentially slow down the whole learning process and lead to poor scalability as we need to adjust much more parameters to train the subsequent layers in the network.

To solve the problem, we propose a method called aggregation optimization. It conduct a aggregation operation on each position in the convolution layer and serve as a middleware to aggregate the active values generated by a convolution cell C_i. It works like a pooling operation except that a pooling is utilized to conduct down sampling by summarizing the features in the whole layer. While aggregation optimization conduct down sampling by summarizing the features for each convolution cell C_i.

Aggregation operation can help to select the most significant feature from each k-scope concatenation set of position i. The formal operation conducted in aggregation optimization on convolution cell C_i is given in Equation 2. The result is taken as a new feature for position i.

$$a_i = \max_{x \in \biguplus_i^k} f(x) = \max \begin{cases} f(w_i), \\ f(w_i \oplus w_{i+1}), \\ f(w_i \oplus w_{i+2}), \\ \ldots, \\ f(w_i \oplus w_{i+1} \oplus \ldots \oplus w_k) \end{cases} \quad (2)$$

We call the neural cell, which conducts the aggregation optimization, aggression cell. Based on the aggregation optimization, the output of the whole convolution layer will be an n-dimensional vector, which is actually the same length with the input document d. It can be easily fed to another k-scope convolutional layer without increasing the number of convolution cells. Thus that the next convolution layer can take it as a new input document, which is actually a high-level abstract representation of the best local features in the previous layer.

Except reducing the number of parameters in the network and boost the learning process, another advantage of the aggregation optimization is that since we present each position i of document into one local feature. And for a n-length document, it can at most hold n detailed informations. Using this representation, we still hold all the details without losing any information.

Fig. 2 show the illustration of feeding the input document through multi convolution layers. For each convolutional layer, there are two kinds of cells. We use $C_{i,j}$ to denote j-th convolution cell on the layer i. $A_{i,j}$ is the corresponding aggregation cell on layer i. The input document is first fed to the Layer 1, in which the k-scope convolution set of position j is firstly computed by convolution cell $C_{1,j}$, then the most significant activate value is selected by aggregation cell $A_{1,j}$. After that, we generate an n dimensional vector as the output. Then the output result will be taken as input to feed to Layer 2, and so on.

Pooling Mechanism: We also bring in a filter and pooling mechanism to further reduce the dimension of the representation and control over-fitting. For each layer, we create a group of different filters for a convolutional layer, and combine the result in a pooling layer, whose function is to select the most significant feature among all the filters. It also help progressively reduce the size of the representation to reduce the amount of parameters and computation for the subsequent layers in the network.

C. MODEL ARCHITECTURE

Based on proposed scope-based convolution, aggregation and pooling method, we propose our model architecture, which is

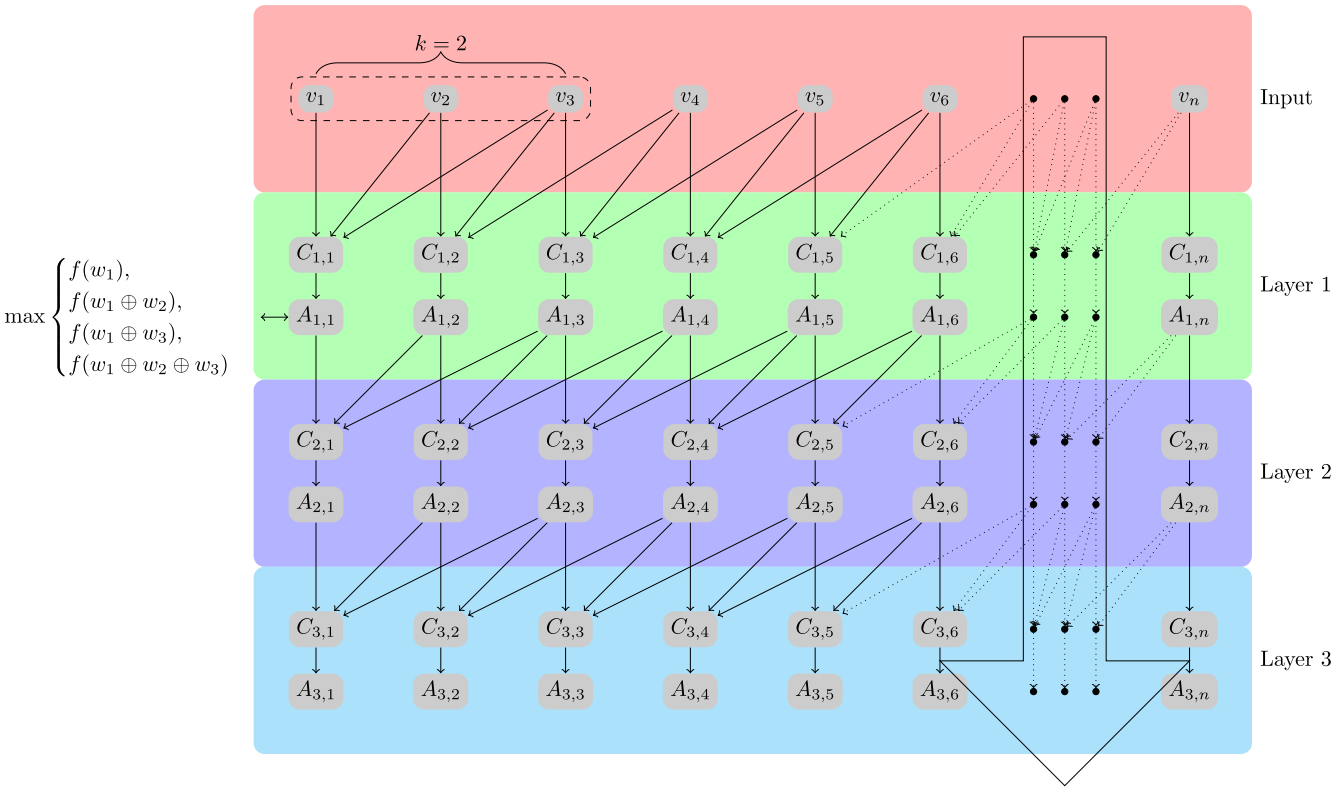


FIGURE 2. (Better viewed in color) Intuitive illustration of k-scope convolution and aggregation optimization. In this example, $k = 2$. In the Layer 1, it first conduct 2-scope convolution operation in each convolution neural cell $C_{1,i}$, in which $i \in [1, n]$. Then it conduct aggregation optimization in aggregation cell, to select the most significant activate value. Those activate values is taken as the new input for Layer 2. Similarly process is conducted on Layer 2 and generate new features fed forward to Layer 3, etc.

illustrated in Fig. 3. The whole framework includes one input layer, several scope-based convolutional layers and pooling layers, one fully connected network, and one output layer.

The input layer takes a text document as an $n \times d$ matrix. The matrix is fed to the k-scope convolutional layer, which conduct k-scope convolution and aggregation operations and generate n features. By utilizing multiple filters, we generate $n \times m$ features, in which m is the number of filters.

Then we connect it to a max pooling layer to further merge the neighbour units thus it can reduce the dimension of the representation and generate a high-level features for the next layer. Then output is fed forward to another k-scope convolutional layer. We repeat the process for several rounds to progressively transform the text to high-level representation and extract the local informations until the representation is good enough to be fed in a fully connected network.

For the fully connected network, we utilize ReLU as the active function for the hidden layers and Softmax function, a generalization of logistic regression to handle multiple classes, as the output layer to compute the probability distribution over all the pre-defined categories.

We utilize stochastic gradient descent (SGD) as the learning algorithm for back-propagate to update the parameters of the objective in the network and minimize the loss function in systemic manner. We also adopt a dropout operation [41] as a regularization technique to reduce overfitting in the neural network.

To further accelerate the performance and improve the scalability, we conduct parallel training using multiple-CPU machine. Notice that our approach conduct k-scope convolution operation and aggregation operation on each position i. The computation on different positions are fully vectorized and separated, such that we can easily transform the whole computation into a parallel manner. Consider using thread number p, we separate a text document into p parts. Each thread handles the computation of positions $\lfloor \frac{n}{p} \rfloor$ or $\lceil \frac{n}{p} \rceil$. In this way, we can achieve the near fully parallel computation.

V. EXPERIMENTS

A. DATASETS

The experiments were conducted on eight real large-scale datasets, which are released in [13] including AG's news corpus(AG), Sogou news corpus(Sogou), DBPedia ontology dataset, Yelp review polarity dataset(Yelp P.), Yelp review full dataset (Yelp F.), Amazon review polarity dataset(Amazon P.), Amazon review full dataset(Amazon F.), and Yahoo answers dataset(Yahoo). The statistics of these datasets can be found in Table 1.

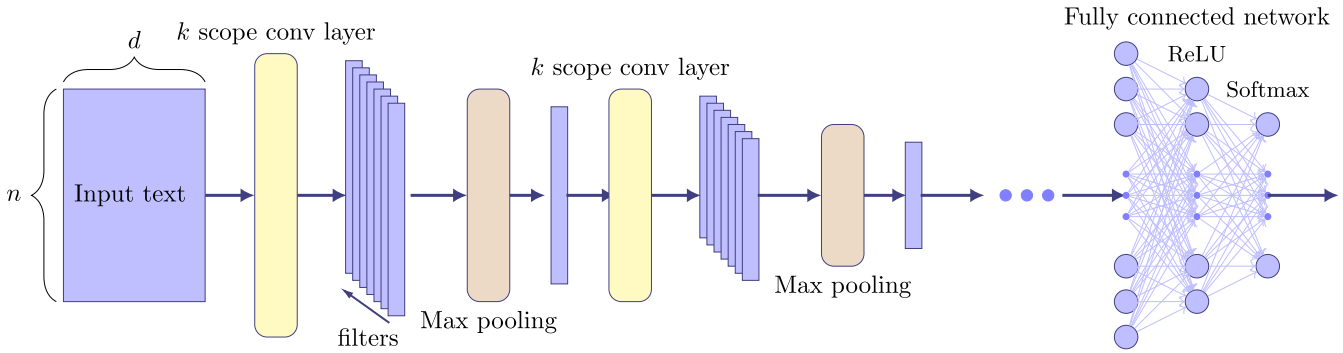


FIGURE 3. (Better viewed in color) Architecture of SCNN model.

TABLE 1. Dataset statistics.

Dataset	classes	Train samples	Test samples
AG	4	120K	7.6K
Sogou	5	450K	60K
DBPedia	14	560K	70K
Yelp P.	2	560K	38K
Yelp F.	5	650K	50K
Amazon P.	2	3.6M	650K
Amazon F.	5	3M	650K
Yahoo	10	1.4M	60K

B. IMPLEMENTATION DETAILS

Our word embeddings is from GloVe [37]. The dimension of pre-trained word vector $d = 300$. The input text sequence was padded to a fixed length n, which is the maximum text document length. For AG's News, Sogou News and DBPedia, $n = 100$; For other dataset $n = 300$ respectively. For all the datasets, we set scope parameter $k = 5$, which conducts 5-scope convolution process. We utilize 3 filters in the model. We modify number of k-scope convolutional layer and max pooling layer to study the influence of the scope parameter k.

Training Settings: We used SGD with a mini-batch of 256. The learning rate is initially set to be 0.01 and gradually decreased to $1e^{-8}$. The training process lasts 20 epoches on all the datasets. We applied "L2" regularization and dropout rate is 0.5 for training data.

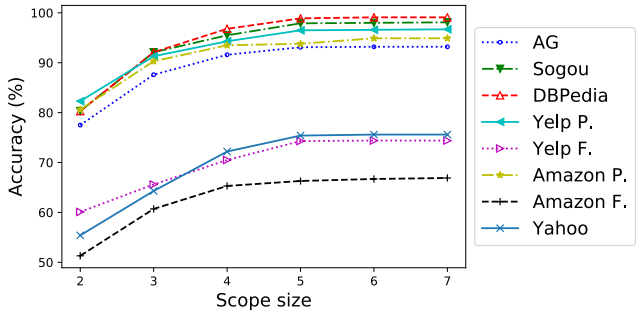


FIGURE 4. Accuracy with different scope sizes.

C. TUNING OF HYPERPARAMTERS

1) SCOPE SIZE

We modify the scope parameter k range from 2 to 7 to study how performance is influenced by the scope size k. As shown in Figure 4, we can find with the increase of scope size k, the trend of accuracy can be separated in two stages. From $k = 2$ to $k = 5$, the accuracy is significantly influenced by scope size k, that is when k increase the accuracy also increase; while from $k = 5$ to $k = 7$, the accuracy is not significantly influenced by k. The result show that 5 is a recommended scope to capture the local features for common text data. The result is in line with the intuition that the words that have a strong connection to each other should be too far away.

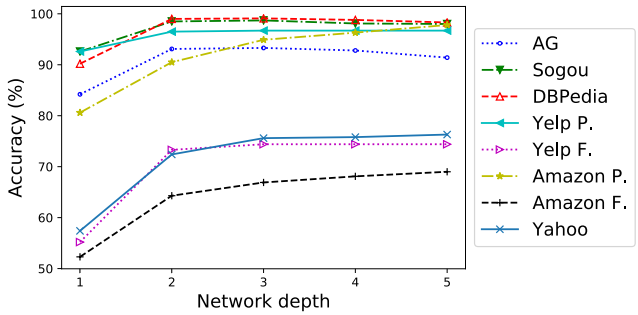


FIGURE 5. Accuracy with different network depths.

2) NETWORK DEPTH

We evaluated how performance is influenced by the network depth l in Fig. 5. We set the scope of $k = 5$ and modify the number of k-scope convolutional layer and pooling layers. We find that with the increase of network depth, the performance is further improved for large datasets like Amazon P. and Amazon F.; While for a small dataset such like AG, it first increased from $l = 1$ to 2, then stopped increasing from 2 to 4, and slightly decreased by further increasing network depth from 4 to 5. The result show that our model can achieve a quite good result with relative few network depth, and deeper network can achieve better performance for larger datasets as we have more learnable parameters. But for small dataset, deeper network also may lead to overfitting. 2 or 3 is a recommend network depth for common text data.

D. SCALABILITY

We also study the scalability of the proposed model. We first study how multiple-CPU influenced the performance. Then we compared the time cost by training different dataset.

TABLE 2. Discontinuous local features discovered by LSS-CNN.

class	context
positive	A **bad mood** just **evaporates** like sugar in the rain
	plus it's **worth** wading through the **tears** to get to the fun parts
	Hawthorne has a **huge flare** for the **dramatic** which is great for the reader
	This card has to be the **best purchase** I've **ever** made!
	the whole city watch series is just about my **favourite** bunch of **books** I have ever read.
	Since you open the **book** you **cannot close** it
negative	it was **very boring** through out the **book**
	but **nothing** ever **gets resolved**
	I finally **returned** the **product** for a different bran
	This is perhaps the **worst** of all **videos** out today of its type
	Worst test prep **book** I have ever used.
	It's a **great** book. **But** the Mindstorms microproessor is now NXT

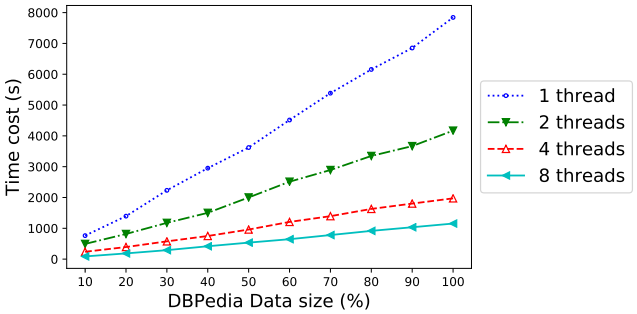


FIGURE 6. Time cost with parallel training.

Fig. 6 show the time cost of parallel training with different thread number. We conduct the experiment on machine with 8 cores. Thus at most we can test on the parallel training cost with 8 threads. We test the training time cost using 1, 2, 4 and 8 threads. We conduct the experiment on DBPedia dataset. We scaled the percent of training dataset from 10% to 100% and test the elapsed time cost. We found that our model could achieve nearly fully parallel acceleration.

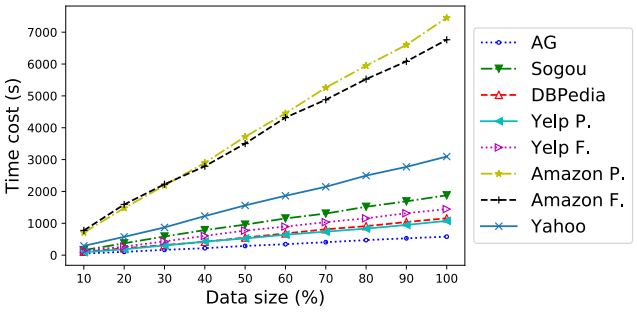


FIGURE 7. Time cost with different data sizes.

Then we fixed using 8 threads, and conduct experiments on all the 8 datasets. We still scaled the percent of training dataset from 10% to 100% and test the elapsed time cost. Fig. 7 show the elapsed time cost by training different datasets with different data sizes. We find the time cost of our proposed method increases almost linearly with the data size. The result show that our proposed scope-based convolution operation and aggregation optimization have a good scalability and can be effectively conducted on big data corpus under multi-CPU environment.

E. CASE STUDY

We also conduct case study of the proposed method by tracking the local feature discovered in our model. Notice that since for efficient consideration, we didn't record detailed information about where does the max pooling value actually come from. It is not an easy job to track the activate words come from.

To execute case study, we conduct experiment on a small portion (1 %) of Amazon P. dataset. We record the features kept in the max pooling layer, then we backtrack the process to restore the word vectors that contribute the discovered local features. To show it more intuitive, we transform the word vector back to their original word representation. To make the representation more clear, we also attach the neighbour context.

Table 2 shows several examples that a discontinuous local features were discovered by LSS-CNN model. For example, text sequence "it was very boring all through out the book", one filter of our model capture **very boring book** as the local feature, which fairly accurately captured the local information. Another interesting discovery is that since we do not take punctuation mark into account, the words constructing a local feature can across two sentence. For example, In the sentence "It's a great book. But the Mindstorms microproessor is now NXT", a filter capture the word **great but** as the maximum activate value, which is just good to reflect the inversion of the semantics.

F. COMPARISON WITH EXISTING METHODS

We compared our model with several existing state-of-art models including linear model [32], TextCNN [11], LSTM models [31], dynamic pooling CNN model [12], character-level CNN model [13] and densely connected CNN attention model [15].

The results are listed in Table 3. The proposed model outperforms most of the existing models on most datasets. Our proposed approach LSS-CNN show significant advantage on large datasets (eg. Yahoo, Amazon F.) because the proposed model can effectively capture complicated deeper local features in the text data. Take TextCNN model for a comparison. Our method outperform TextCNN in all the datasets. We believe the reason is that scope-based convolution in

TABLE 3. Accuracy of all the models on the eight datasets.

Model	AG	Sogou	DBPedia	Yelp P.	Yelp F.	Amazon F.	Amazon P.	Yahoo
Linear Model	91.9	86.2	95.2	95.3	63.7	60.6	93.2	60.7
LSTM	92.2	96.1	98.5	92.8	59.3	66.3	91.2	63.6
Dynamic-Pool CNN	91.2	93.5	98.4	94.7	63.0	61.3	93.1	69.4
Char-level CNN	87.1	92.4	98.3	94.6	61.2	59.3	93.4	72.5
TextCNN	91.5	96.3	98.6	93.5	67.2	57.5	93.6	71.8
Densly Connected CNN	**93.5**	97.7	**99.1**	96.4	66.1	62.9	94.7	73.3
LSS-CNN	93.3	**98.1**	**99.1**	**96.7**	**74.4**	**66.9**	**94.9**	**75.6**

LSS-CNN capture better local features than window-based structure in TextCNN. And our method is quite easy to find a good setting. To find a good setting, it needs to try different combinations of window sizes. And in the experiment, in AG, Sogou, and DBPedia the best performance for TextCNN was found with combination of window sizes as [3, 4, 5, 7, 11, 15, 20, 25]; while for Yelp P. dataset and Yelp F. dataset, the best performance is found with window sizes [2, 3, 4, 5, 9, 11, 17, 22]; and for Amazon P., Amazon F. and Yahoo datasets, the best performance is found with [3, 4, 5, 6, 8, 15, 25]. While our approach just need to try a few different scope size. Actually in all the experiments, we just fixed $k = 5$ for all the datasets. This is in line with our claim that our proposed SCNN model is more flexible and more easy setting to capture complicated local features.

Compared with densely connected CNN attention model, our approach just require 3 convolutional layers to achieve a good result, while we can achieve higher accuracy compared with the densely connected CNN attention model with 6 convolutional layers.

VI. CONCLUSION

In this paper, we present LSS-CNN, a new large-scale scope-based convolutional neural network for solving text classification problem. Different from CNN, which uses a sliding window to capture local feature, LSS-CNN is based on a concept called scope. We propose scope convolution operation, which is more flexible and efficient to capture complicated local features such as variable length phrases or discontinuous phrases in the text data. We also propose aggregate optimization to accelerate the convolution operation and still able to capture the deeper local information. We also utilize a pooling mechanism to reduce the dimension of the representation for fully connect network, and parallel training to boost the model. We verify the effectiveness of the method by conducting experimental study on eight real text datasets, and compare our proposed model with the existing state-of-the-art approaches. The result demonstrates that our proposed model can achieve the competitive performance.

REFERENCES

[1] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 2267–2273.

[2] T. Wu, S. Liu, J. Zhang, and Y. Xiang, "Twitter spam detection based on deep learning," in *Proc. ACSW*, 2017, Art. no. 3.

[3] M. V. Mäntylä, D. Graziotin, and M. Kuutila, "The evolution of sentiment analysis—A review of research topics, venues, and top cited papers," *Comput. Sci. Rev.*, vol. 27, pp. 16–32, Feb. 2018.

[4] J. He, Z. Hu, T. Berg-Kirkpatrick, Y. Huang, and E. P. Xing, "Efficient correlated topic modeling with topic embedding," in *Proc. KDD*, 2017, pp. 225–233.

[5] Y. Wu *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *CoRR*, Sep. 2016. [Online]. Available: http://arxiv.org/abs/1609.08144

[6] J.-K. Kim, G. Tür, A. Çelikyilmaz, B. Cao, and Y.-Y. Wang, "Intent detection using semantically enriched word embeddings," in *Proc. IEEE Spoken Lang. Technol. Workshop (SLT)*, Dec. 2016, pp. 414–419.

[7] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell, "Text classification from labeled and unlabeled documents using EM," *Mach. Learn.*, vol. 39, nos. 2–3, pp. 103–134, 2000.

[8] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins, "Text classification using string kernels," *J. Mach. Learn. Res.*, vol. 2, pp. 419–444, Feb. 2002.

[9] G. Forman, "An extensive empirical study of feature selection metrics for text classification," *J. Mach. Learn. Res.*, vol. 3, pp. 1289–1305, Mar. 2003.

[10] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *J. Mach. Learn. Res.*, vol. 2, pp. 45–66, Nov. 2001.

[11] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. EMNLP*, 2014, pp. 1–6.

[12] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," in *Proc. ACL*, 2014, pp. 1–11.

[13] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Proc. NIPS*, 2015, pp. 649–657.

[14] A. Conneau, H. Schwenk, L. Barrault, and Y. LeCun, "Very deep convolutional networks for text classification," *CoRR*, Jun. 2016. [Online]. Available: http://arxiv.org/abs/1606.01781

[15] S. Wang, M. Huang, and Z. Deng, "Densely connected CNN with multi-scale feature attention for text classification," in *Proc. IJCAI*, 2018, pp. 1–7.

[16] T. Lei, R. Barzilay, and T. Jaakkola, "Molding CNNs for text: Non-linear, non-consecutive convolutions," in *Proc. EMNLP*, 2015, pp. 1–11.

[17] Z. Wang, Z. Wang, D. Zhang, and J. Yan, "Combining knowledge with deep convolutional neural networks for short text classification," in *Proc. IJCAI*, 2017, pp. 2915–2921.

[18] M. Coavoux and B. Crabbé, "Incremental discontinuous phrase structure parsing with the GAP transition," in *Proc. EACL*, 2017, pp. 1–13.

[19] H. Borko and M. Bernick, "Automatic document classification," *J. ACM*, vol. 10, no. 2, pp. 151–162, 1963.

[20] M. Sasaki and K. Kita, "Rule-based text categorization using hierarchical categories," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, vol. 3, Oct. 1998, pp. 2827–2830.

[21] C. C. Aggarwal and C. Zhai, "A survey of text classification algorithms," in *Mining Text Data*. Springer, 2012, pp. 163–222. [Online]. Available: https://dblp.uni-trier.de/rec/bibtex/books/sp/mining2012/AggarwalZ12b

[22] F. Sebastiani, "Machine learning in automated text categorization," 2001, *arXiv:cs/0110053*. [Online]. Available: https://arxiv.org/abs/cs/0110053

[23] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, "A Bayesian approach to filtering junk e-mail," in *Proc. Learn. Text Categorization, Papers Workshop*, 1998, pp. 98–105.

[24] D. E. Johnson, F. J. Oles, T. Zhang, and T. Götz, "A decision-tree-based symbolic rule induction system for text categorization," *IBM Syst. J.*, vol. 41, no. 3, pp. 428–437, 2002.

[25] T. Joachims, "A statistical learning learning model of text classification for support vector machines," in *Proc. SIGIR*, 2001, pp. 128–136.

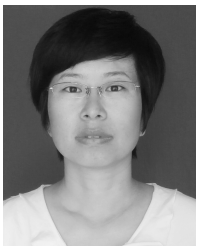
[26] S. Scott and S. Matwin, "Feature engineering for text classification," in *Proc. ICML*, 1999.

[27] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.
[28] Y. Li, R. Jin, and Y. Luo, "Classifying relations in clinical narratives using segment graph convolutional and recurrent neural networks (Seg-GCRNs)," *J. Amer. Med. Inform. Assoc.*, vol. 26, no. 3, pp. 262–268, 2018.
[29] J. Wu, L. Zou, L. Zhao, A. Ai-Dubai, L. Mackenzie, and G. Min, "A multi-UAV clustering strategy for reducing insecure communication range," *Comput. Netw.*, vol. 158, pp. 132–142, Jul. 2019.
[30] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *J. Mach. Learn. Res.*, vol. 12, pp. 2493–2537, Aug. 2011.
[31] R. Johnson and T. Zhang, "Deep pyramid convolutional neural networks for text categorization," in *Proc. ACL*, 2017, pp. 562–570.
[32] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," in *Proc. EACL*, 2017, pp. 1–5.
[33] P. Liu, X. Qiu, and X. Huang, "Recurrent neural network for text classification with multi-task learning," in *Proc. IJCAI*, 2016, pp. 1–7.
[34] C. Zhou, C. Sun, Z. Liu, and F. C. M. Lau, "A C-LSTM neural network for text classification," *CoRR*, Nov. 2015. [Online]. Available: http://arxiv.org/abs/1511.08630
[35] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," in *Proc. ACL*, 2015, pp. 1–11.
[36] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, Jan. 2013. [Online]. Available: http://arxiv.org/abs/1301.3781
[37] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proc. EMNLP*, 2014, pp. 1532–1543.
[38] O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization," in *Proc. NIPS*, 2014, pp. 2177–2185.
[39] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin, "Learning sentiment-specific word embedding for Twitter sentiment classification," in *Proc. ACL*, 2014, pp. 1555–1565.
[40] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *CoRR*, Oct. 2018. [Online]. Available: http://arxiv.org/abs/1810.04805
[41] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

JIAYING WANG received the Ph.D. degree in computer science from Northeastern University, China, in 2016. He is currently an Associate Professor with the Information and Control Engineering Faculty, Shenyang Jianzhu University, China. His research interests include big data processing, data mining, and approximate data query processing and optimization. He is a member of CCF.

YAXIN LI is currently pursuing the M.S. degree in computer science with Shenyang Jianzhu University, China. Her research interests include big data processing and deep learning.

JING SHAN received the Ph.D. degree in computer science from Northeastern University, China, in 2016. She is currently an Associate Professor with the Information and Control Engineering Faculty, Shenyang Jianzhu University, China. Her research interests include social network analysis, community mining, and graph mining. She is a member of CCF.



JINLING BAO received the Ph.D. degree in computer software and theory from Northeastern University, China, in 2018. She is currently an Associate Professor with the Computer Science Faculty, Baicheng Normal University, China. Her research interests include spatial database and approximate data query processing, and optimization.

CHUANYU ZONG received the Ph.D. degree from the School of Computer Science and Engineering, Northeastern University, in 2017, China. He is currently a Lecturer with Shenyang Aerospace University, China. His research interests mainly include data cleaning, data provenance, query processing, and optimization.

LIANG ZHAO received the Ph.D. degree from the School of Computing, Edinburgh Napier University, in 2011. He worked as an Associate Senior Researcher with Research and Development Corporation, Hitachi, China, from 2012 to 2014. He is currently an Associate Professor with Shenyang Aerospace University, China. His research interests include VANETs, SDVN, FANETs, and WMNs.