A Word Distributed Representation Based Framework for Large-scale Short Text Classification

Di Yao, Jingping Bi, Jianhui Huang
Institute of Computing Technology
Chinese Academy of Sciences
Beijing, 100190 China
Email: {yaodi, bjp, huangjianhui}@ict.ac.cn

Jin Zhu
Yunnan University of Nationalities
Kunming, 650031 China
Email: zhujin833@163.com

***Abstract*—With the development of internet, there are billions of short texts generated each day. However, the accuracy of large scale short text classification is poor due to the data sparseness. Traditional methods used to use external dataset to enrich the representation of document and solve the data sparsity problem. But external dataset which matches the specific short texts is hard to find. In this paper, we propose a framework to solve the data sparsity problem without using external dataset. Our framework deal with large scale short text by making the most of semantic similarity of words which learned from the training short texts. First, we learn word distributed representation and measure the word semantic similarity from the training short texts. Then, we propose a method which enrich the document representation by using the word semantic similarity information. At last, we build classifiers based on the enriched representation. We evaluate our framework on both the benchmark dataset(Standford Sentiment Treebank) and the large scale Chinese news title dataset which collected by ourselves. For the benchmark dataset, using our framework can improve 3% classification accuracy. The result we tested on the large scale Chinese news title dataset shows that our framework achieve better result with the increase of the training set size.**

I. Introduction

During the past decades, text classification (text categorization) is one of the most important processes used in mining knowledge from large data set. Lots of machine learning methods have proved to be effective tools for text classification such as K-NN, SVM, Naive Bayes and Maximum Entroy. These advanced techniques have achieved excellent results on some benchmark collections (20Newsgroups, Reuters-21578).

With the rapid developing of Internet, many applications such as Twitter, Microblogs, Q&A System ,Web search and e-commerce produce large amount of short texts every day. The volume of these data is very large while the length of each is short. Therefore, it plays a key role to process text data in many fields. For instance, the result of short text classification can be used to improve the accuracy rate of information retrieval system. However, due to the specificity of the short text, the processing of it encounter new challenges. Compared with normal long text, short text has less words and much noises. The less words make the representation of short text become more sparse and it has no efficient information on topic and word co-occurrence. So normal machine learning methods are not suitable for the tasks in short text.

There have been several works that attempted to solve the sparsity and noisy of short text. These works can be categorized into two groups. The first group try to use external data sources to enrich the short text representation. One way is to extract topic from the external data sources using topic model such as pLSA [6] and LDA [5]. Another way is to measure the word similarity from WordNet or Wikipedia. A disadvantage of this group is that the selection of external data sources is very hard and there is no proper external data to some special problem. The second group proposed new models which take the specificity of short text into consideration. Biterm-based topic model [4] is used to extract topic information from short text. However, this group of works represents the short text in One-hot representation such as Vector Space Model (VSM) which loses the semantic information to some extent.

Inspired by the works mentioned above, we present a framework for building a classifier which aimed to deal with large short text corpus using their own semantics information. The main idea of the framework is that for each word in the corpus, we learn a distributed representation for words using neural probabilistic language model [2] and then we calculate the representation for each document which combines the semantic information of word and the structure information of document. Finally, we use machine learning method to build the classification model. The framework is mainly based on word distributed representation and powerful classifier such as maximum entropy and SVMs.

We test our framework on the benchmark collection and the real Chinese news title dataset, the results of both data sets show that our work is outperform the state-of-art methods for short text classification.

The main contributions of our work is summarized as follows:

- To the best of our knowledge, this is the first attempt to apply neural probabilistic language model and the word distributed representation for short text classification. Our experimental study has validated its effectiveness.
- We proposed a method which solves data sparsity problem of short text without any external data source. The document representation is enriched by adding the word semantic similarity information.
- We tested our framework in different training sets. The result shows that our framework has better accuracy rate on large training sets.

978-1-4799-1959-8/15/$31.00

The rest of this paper is organized as follows: in Section 2 we present the related work on short text classification. Then we give an general overview of our frame work in Section 3. In Section 4 we analyse the main steps in our framework. Section 5 describes the experiments of our framework on the benchmark collection and the real data on the Internet. Finally, we conclude our work in Section 6.

II. Related Work

"Distributed Representations of Sentences and Documents" by Quoc Le [1] is probably the study most related to our work. Quoc Le proposed a method for sentence and document representation based on the word distributed representation that proposed by Mikolov [3]. This approach solve the problem of data sparsity and learn a distributed representation of document. However, using Distributed Memory Model of Paragraph Vector(PV-DM) which only take the nearby words into consideration to get the document representation loses the globe information of the whole training set. So in our framework, we combine the word distributed representation with the globe information together to represent the document.

The works related to the short text representation can be divided in two groups. One group of works focus on the language model which used to refine the representation of short text. Dou Shen [11] extends the n-gram based naive Bayes classifier by using multigram language model. The integration of the multigram language model can improve the accuracy of naive Bayes classifier. Vidit Jain [8] advanced a short text representation using diffusion wavelets. Neural probabilistic language model is promoted by Bengio [2]. It is the first attempt that represent a word to a continuous vector. In our framework, the distributed representation of word is based on the neural probabilistic language model.

Another group of studies focus on using the topic information of external data sources to enrich the short text representation. Sarah Zelikovitz [7] using LSI to extract topics for short text classification. Xuan-Hieu Phan [9]'s work learns topics from external data collections and build the classifier using maximum entropy and SVMs. Mengen Chen [10] improve the short text classifier by learning multi-granularity topics. The main difference between these works and our work is that we use the word similarity instead of topic to enrich the representation and we use no external data sources.

With respect to the classifier, SVMs have a natural advantage for classifying text [12]. Xinruo Sun [16] proposed a multi-stage classification approach to classify short text. SVM also can be integrated with other classifier such as MaxEnt to make a comprehensive classification result [17]. And many short text classification works we mentioned above are using SVM to build the classifier [9] [10].

III. The General Framework

In this section we present the proposed framework that aimed to build a classifier for large-scale short text. The framework is depicted in Fig. 1.

Fig. 1 explains four steps for building the large-scale short text classifier. The general idea of each step is described as follows:

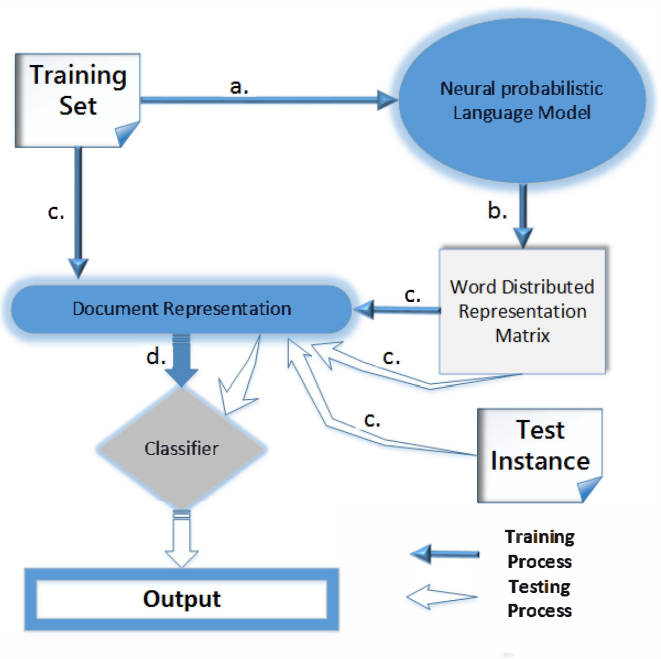

Fig. 1: The General Framework for Our Work

- Step a: Preprocessing. In this step, short text in training set is preprocessed as the input of neural probabilistic language model. In general, the preprocessing includes removing stop words(usually omit for short text), named entity recognition, text segmentation(may be used in some scenarios) and etc.
- Step b: Learning the word distributed representation. Using neural probabilistic language model, we can learn a distributed representation for words from the preprocessed training set. This representation can represent word semantic information.
- Step c: Learning a new document representation. Combine the word semantic information with the traditional document representation methods, we propose a new method to learn a new document representation. Documents in both the training and test set can be represented as a vector using our method.
- Step d: Building the classifier. In training process, we build a classifier based on the training instances representation and label. In test process, test instances is input to the classifier and the classifier outputs the label.

In the next section, we first introduce the techniques we used for learning the word representation. Then, our document representation method is specified. At last, we will explain the technique we used to build the classifier.

IV. Methodology

A. Distributed Representation of Word

The main idea of statistical language model is that the probability of a document can be represented as the joint probability of each word denoted as $P(w_1, w_2, \cdots, w_T)$. There are many

TABLE I: Variables Used in Word Distributed Representation

Variable	Details
w_i	Word i
V	Vocabulary of the corpus
C	Word distributed representation matrix for all words
$C(w_i)$	Distributed representation for word w_i
θ	Overall parameters set $\theta = (b, d, W, U, H, C)$
$R(\theta)$	Regularization term
b	Bias of Input layer to Output layer
W	Weight of Input layer to Output layer
d	Bias of Input layer to Hidden layer
H	Weight of Input layer to Hidden layer
U	Weight of Hidden layer to Output layer
L	Penalized Log-likelihood function for neural language model

ways to model the joint probability. In general, The probability of each word can be represent by the conditional probability of the next word given all the pervious words.

$$P(w_1, w_2, \cdots, w_T) = \prod_{t=1}^{T} P(w_t | w_1, w_2, \cdots, w_{t-1}) \quad (1)$$

where w_t is the t-th word. Considering the fact that farther words in the word sequence are statically more dependent, so the conditional probability of the next word given all the pervious words is approximately equal to it given the nearest n words.

$$P(w_t | w_1, w_2, \cdots, w_{t-1}) \approx P(w_t | w_{t-n+1}, \cdots, w_{t-1}) \quad (2)$$

This is the main idea of n-gram model. This model is widely used in many technological applications involved in natural language processing. However, this model does not consider the semantic of the document. Two documents have similar semantic meaning could have extremely different probability. To solve this problem, Bengio [2] extended the n-gram model using neural network and proposed a neural language model(NLM).

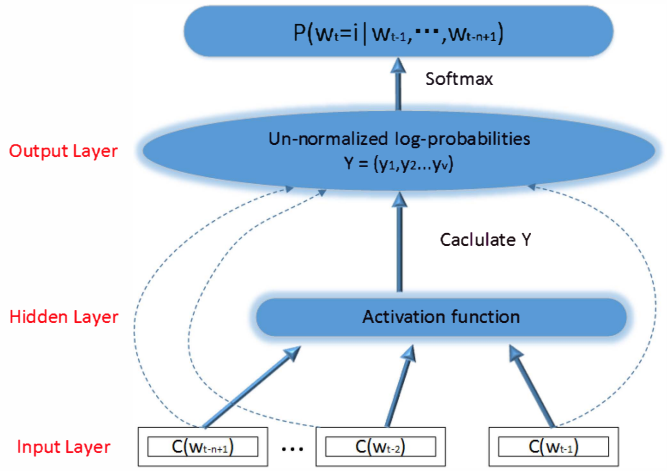

Fig. 2: Neural Architecture for NLM

The architecture of neural language model is shown in Fig. 2. Using this model, we can get the distributed representation of words in the vocabulary. Given a sequence of training words $w_1, w_2, \cdots, w_T$, the penalized log-likelihood function of the training corpus L.

$$L = \frac{1}{T} \sum_{t=1}^{T} log(P(w_t | w_1, w_2, \cdots, w_{t-1}, \theta)) + R(\theta) \quad (3)$$

where θ is the overall parameters of the model and $R(\theta)$ is a regularization term. We need to maximize the penalized log-likelihood function L and learn the parameters θ. Next, we will illustrate this model layer by layer.

For the input layer, we stitch each word feature vector together and denote it as x.

$$x = (C(w_{t-1}), C(w_{t-2}), \cdots, C(w_{t-n+1})) \quad (4)$$

where $C(w_t)$ is the word vector of w_t. For the hidden layer, the activation function is $g(x)$. So the output of hidden layer is $g(d + H \cdot x)$, where the H is the hidden layer weight, d is the hidden layer bias. For the output layer, the input of this layer can be divided into two parts. Part one is from the input layer. Part two is from the hidden layer. So the output of output layer y:

$$y = b + W \cdot x + U \cdot g(d + H \cdot x) \quad (5)$$

where b is the output biases, W is the output weight, U is the hidden-to-output weight. So the overall parameters $\theta = (b, d, W, U, H, C)$. Finally, the softmax function of this model is the logistic function.

$$P(w_t | w_{t-n+1}, \cdots, w_{t-1}) = \frac{e^{y_{w_t}}}{\sum_{i=1}^{v} e^{y_{w_i}}} \quad (6)$$

In the training process, stochastic gradient ascent is used to infer the parameters. For each step iteration, θ is calculated as follows:

$$\theta \leftarrow \theta + \varepsilon \cdot \frac{\partial \log P(w_t | w_{t-n+1}, \cdots, w_{t-1})}{\partial \theta} \quad (7)$$

where ε is the learning rate. After iteration process, we can get the parameters θ.

Neural probabilistic language model encounter a performance bottle-neck when applying on large-scale data set. The calculation in hidden layer is the most time consuming process. In order to extend the NLM architecture to large scale collections. Mikolov [3] improves the neural probabilistic language model and get observe large improvements in accuracy at much lower computational cost.

The architecture of Mikolov's work is shown in Figure3. The implementation of Mikolov's algorithm is available at word2vec. There are three different points between neural probabilistic language model and word2vec.

1) : The input layer of word2vec is the sum not the stitch of word vectors.

2) : There is no hidden layer in word2vec architecture.

3) : The output layer of word2vec is a Huffman tree not a linear structure.

Word2vec has been tested to be useful in many applications such as the word similarity. So we choose word2vec to train the word distributed representation.

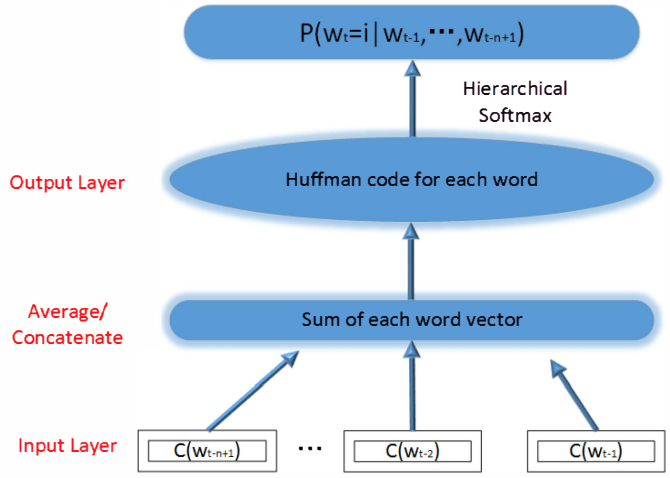

Fig. 3: Neural Architecture for Word2Vec

B. Our Document Representation Method

Nowadays, perhaps the most popular vector representation method for document is the bag-of-word. Due to its simplicity, efficiency and often surprising accuracy, the bag-of-words model also used to represent the input document for many classifier. However, the bag-of-words has many disadvantages. Using bag-of-word model, the word order information of document is lost. Thus, different document would have the same representation if the same word are used. The bag-of-words model consider little about the semantics of the words. It also leads to the data sparsity and curse of dimensionality.

Quoc Le [1] proposed paragraph vector to replace the bag-of-words model. This paragraph vector is learned based on the word2vec algorithm. However, word2vec learning the word vector only use the local information of word because word2vec is based on the n-gram model.

In order to overcome the above-mentioned disadvantages, we propose a new method to represent the document which combine the bag-of-words model and the word distributed representation.

Assuming that the representation of a document we get using n-gram bag-of-word is $d_{BOW} = (f_1, f_2, \cdots, f_v)$ where f_i is the value of word w_i, V is the length of word dictionary for training set. For the word w_i in document d, the method we used to compute the value is tf-idf.

$$\begin{aligned} f_i &= tf(w_i, d) \cdot idf(w_i, D) \\ &= \frac{|w_i \in d|}{N_d} \cdot \log \frac{N_D}{|d \in D \,\&\, w_i \in d|} \end{aligned} \tag{8}$$

where N_d is the word count of document d, N_D is the document count of training set D. Due to the sparsity of the short text, most of values in d_{BOW} is zero. So we need a method to enrich the values. Considering the word distributed representation method which is illustrated above, we measure the word similarity. Words which have high similarity should have similar value in d_{BOW}. We calculate the word similarity as follows:

$$similar(w_i, w_j) = \frac{C(w_i) \cdot C(w_j)}{|C(w_i)| \cdot |C(w_j)|} \tag{9}$$

where the $C(w_i)$ is the distributed representation of word w_i. And then we enrich the representation of the document d which contains w_i and doesn't contain w_j. For each word w_i in the document, we enrich the value of similar word w_j as follows:

$$f_j = f_i \cdot similar(w_i, w_j) \tag{10}$$

In order to bring little noise to new document representation, we set a threshold to the similarity between words. If the word similarity is bigger than the threshold we add it to the document representation. For the word similar with two or more words in the document, we choose the biggest value of it. Our new document representation is calculated by using Algorithm 1.

Algorithm 1 Document Representation Algorithm

Input:
The word dictionary of corpus W
The word dictionary of word2vec training set W_{w2v}
The word similarity threshold t
Output:
The document representation d_{new}

$d_{new} = \{\}$
for each word w in document d **do**
  for each word w_{new} in d_{new} **do**
    if $w_{new} == w$ **then**
      $d_{new}.remove(w_{new})$
    end if
  end for
  $d_{new}.add(w)$
  if w in W_{w2v} **then**
    for w_{sim} in $SimilarityList(w)$ **do**
      if $w_{sim}.simility \geq t$ **and** w_{sim} not in d **then**
        $w_{sim}.weight = w.weight * w_{sim}.simility$
        for each word w_t in d_{new} **do**
          if $w_t == w_{sim}$ **then**
            $d_{new}.remove(w_t)$
            if $w_t.weight \geq w_{sim}.weight$ **then**
              $w_{sim} = w_t$
            end if
          end if
        end for
        $d_{new}.add(w_{sim})$
      end if
    end for
  end if
end for

As is shown in Algorithm 1, a document is represented as a list of words. For each word, we find the words which is similar to it and add the similar word into the document representation. In this algorithm, The most time consuming process is the calculation of $SimilarityList$ of w. However, the calculation result of word similarity is reusable. We can calculate and freeze it in the training process. Because the test set is usually not very large, the increase of computational complexity can be tolerated.

C. Technique to Build the Classifier

With respect to the classifier, Support Vector Machine(SVM) is an efficient method used for text classification

[12]. According to the above-mentioned document representation method, document d_i in the training set can be represent as (X_i, Y_i), where X_i is the representation of document i and Y is the index of class which document d_i belongs. In SVM the decision boundary which predicts the class of test instance is chosen to be the one for which the margin is maximized. The margin is defined as the distance between the decision boundary and the closest point that represent a vector in high dimensional space [18]. Short text have many properties such as high dimensional input space, document vectors are sparse and most of documents are linearly separable, which fit the character of SVM.

Assuming there are l instances in the training set, the purpose of SVM for binary-class linearly separable is to solve the optimization problems as follows:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2} w^T w \\ s.t. \quad & Y_i(w \cdot X_i + b) \geq 1, i = 1, 2, \cdots, l \end{aligned} \tag{11}$$

For the multi-classes problem, there are various methods which were proposed for combining multiple binary-class SVMs in order to build a multi-classes classifier. We use the one-versus-the-rest approach which trains N SVM classifiers for N classes to solve this problem.

For the linearly inseparable dataset, there are two method used to solve this problem: kernel method and soft margin. The soft margin algorithms in SVMs can change each linearly inseparable problem into a separable one. SVM use the kernel method which transforms the original data to a new Hilbert space H. Vectors in the new space is linearly separable.

However, for large-scale short text classification, kernel method is much more sophisticated. So considering the fact that most of documents are linearly separable, we choose soft margin multi-classes SVM in our framework. The loss function of it is shown as follows:

$$\min_{w} \quad \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi(w; X_i, Y_i) \tag{12}$$

where $C > 0$ is a penalty parameter. $\xi(w; X_i, Y_i)$ is the loss function.

The effectiveness of it has been tested by C.-J. Lin. The code and data of Lin's work is available on github project libshorttext [13] and libliner [14]. Using their work, we can choose the loss function and regularization for SVM by changing the parameters. The loss function can choose L1-loss or L2-loss.

V. Experiment

In this section, we perform experiments to evaluate our framework. We test our framework on two short text classification tasks: sentiment analysis and news title classification. The reason why sentiment analysis experiment is chosen is because the result of this experiment shows that our framework can improve the accuracy of classification on benchmark dataset. And, the news title classification task is chosen for evaluating the effectiveness of our framework on large scale dataset.

A. Data Set

For sentiment analysis, we use Standford sentiment treebank dataset which proposed by Socher at 2013 [15] to benchmark our work. This dataset has 11855 documents which were taken from the movie review site Rotten Tomatoes. Socher divided the dataset into three sets: training set, test set and validation set. There are 8544 documents for training, 2210 documents for test and 1101 documents for validation. Each document in Socher's dataset is a single short sentence.

The dataset not only labeled the documents but also came with detailed labels for subphrases in each document. With similar experimental to Quoc Le [1], we only consider the label of the document. The author of this dataset proposed two ways to use the label. For each document, we could consider a 5-classes classification task where the labels are {Very Negative, Negative, Neutral, Positive, Very Positive} or a 2-classes classification task where the labels are {Negative, Positive}.

For news title classification, we use all news in 2014 which is crawled on ChinaNews. There are 917626 news titles in the dataset. Each news belongs to a session such as sport news, finance news and so on. For the whole dataset, there are 31 sessions corresponding to 31 classes. The biggest class is social news which has 113414 news. Due to different classes may have similar meaning, in our experiment, we evaluate our framework with 559031 news titles from 5 distinguishing classes. We equally and randomly separate all data into 10 subset. The first subset which inclued 55899 documents is treated as the test set. The premier training set which is one of the 9 subsets has 55907 news. The second training sets are add a subset to the premier training set, and so on. Finally, we construct 9 training sets which have different volumes.

B. Compared Methods & Experimental Protocols

The main idea of our work is to enrich the document representation using the word semantic similarity which is learned by neural language model. Then, The enriched representation of documents are used to learn the classifier. So the baseline we compared is the classification result which use the original BoW document representation.

Socher compared serval methods to their dataset and finded that their works much better than bag-of-word method. There are two reasons why we don't choose their work as the baseline. One is that our work aims to solve the data sparsity by enriching the representation while they don't. Another reason is that our framework is used to classify large-scale short text. However, Socher's work is too complicated to used in big data.

Followed the experimental protocols described in III, we divide our experiment into training process and test process. There are two phases in the training process. To make full use of the training data, in the first phases, we treat each sub phrase as an independent document and we learn the distributed representation for all words using word2vec [3]. In the second phase, we use our method to represent the document and train the classifier. Considering our work is used for large-scale short text classification, we choose linear SVM to build our classifier [14].

In the test process, we freeze the vector representation for each word, and represent the test instance using our method. Once the test document is represented as a vector, we use the classifier which is trained above to classify it. However, in our experiment, the dictionaries we get from the bag-of-word and that we get from word distributed representation are not same. In this case, we choose the bag-of-word dictionary as the standard one. If the similar word is not in the standard dictionary, we will ignore it.

C. Result & Analysis

We learned the word distributed representation from the training instances in Standford sentiment treebank dataset and evaluated the similarity of words. The partial result of this work is given in Table II. It is clear that the semantics similarity of word can be represent using the word distributed representation. For instance, the word $movie$ is similar with $film$ on semantic.

TABLE II: Word Similarity Result

word	similar words(word similarity)
movie	film 0.541, it 0.397, . 0.316, is 0.303, movies 0.295
a	the 0.425, A 0.394, . 0.391, that 0.356, an 0.353
I	'm 0.672, liked 0.554, myself 0.538, ribcage 0.537, Memories 0.537
he	his 0.542, disloyal 0.462, satyr 0.459, evaluate 0.445, swaggers 0.432
Good	Pretty 0.458, Girl 0.445, Next 0.437, Society 0.400, Poets 0.390,
But	While 0.335, Interminably 0.333, it 0.312, But 0.304, conceal 0.302,
painful	elegy 0.431, underplays 0.357, utterly 0.352, lie 0.348, fearless 0.347
happiness	felicity 0.647, fine 0.544, Raphael 0.535, express 0.531, plung 0.512

In our experiment, we use word2vec to train the word distributed representation. There are several parameters we can tune in word2vec. For each document in our training set is short, we set the learning rate = 0.025 window = 5 sample = 0.001 and achieve significant semantic similarity of words. So in the next step, we can enrich the document representation by using these information and build the classifier.

Then, we built classifiers using soft margin multi-classes SVM. The result of our framework is shown in Table III.

TABLE III: Sentiment Analysis Result

Method	Binary-Classes error rate	Multi-Class error rate
Soft Margin SVM	22.1	61.0
Kernel SVM	20.6	59.3
Our Method	18.6	58.1

The result shows that our method has an absolute improvement of 3% in terms of error rate. This result means that words similarity information can make the classifier better.

For the news title classification task, we freeze the test set and expand the training set step by step according to the separation of training set we mentioned above. The result of our experiment is shown in Fig. 4. Compared with the BoW representation+SVM framework, our framework achieve better accuracy rate and the gap between them is trend to increase as the training set is enlarged. The reason about this result is that the word similarity becomes more accurate. One could easily

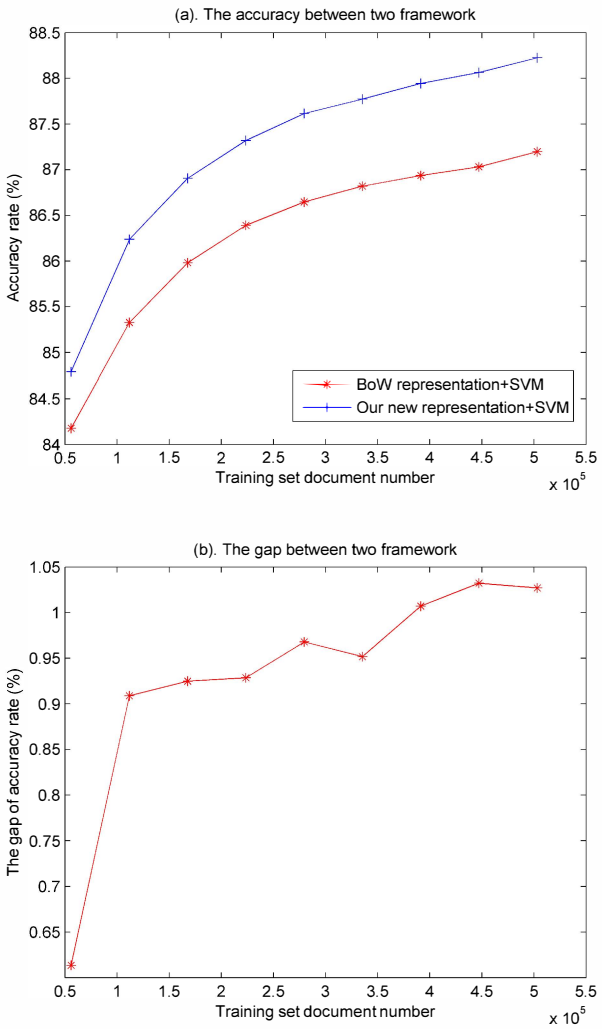

Fig. 4: News Titles Classification Result

argue that if the training set only contains one training subset, the word similarity would bring noise in the classifier. The result illustrate that the our framework can be easily used on Chinese classification task. Using our framework, the accuracy rate of classification has significantly increased.

VI. Conclusion & Future Work

In this paper, we proposed a word distributed representation based framework for short text classification. In our framework, neural probabilistic language model is used to generate the word distributed representation. Using the word similarity information mining from the word distributed representation, we enriched the document representation and solved the data sparsity problem. Then, we adopt the one-vs-all strategy to combine many soft margin binary-class SVM classifiers together for multi-class classification.

To the best of our knowledge, it is the first attempt that the word distributed representation based word similarity information is used for large scale short text classification. We propose a efficient document representation method which enrich the short text representation without using external data

source and integrate it in our framework. Then, we demonstrate the efficiency of our framework on both the Chinese news title dataset and Standford sentiment treebank dataset. Experimental results show that our approaches significantly improve the prediction accuracy while maintaining the time efficiency in the prediction phase.

As to future work, there are two aspects which probably can improve our framework. First, in our work, the document representation only consider the word similarity information. We are interested in taking the word order information into consideration to enrich the document representation. Second, due to the training set is usually not large, the similarity information may be incorrect. We are also interested in using external datasets to refine the word similarity information. We believe that the proposed framework has great potential to achieve much better result.

ACKNOWLEDGMENT

We would like to thank Mikolov for sharing his implementation of word2vec. This work is supported by research grant No.61472403 and No. 61303243 from National Natural Science Foundation of China (NSFC).

REFERENCES

[1] Le Q V, Mikolov T. Distributed Representations of Sentences and Documents[C].//Proceedings of the 31 st International Conference on Machine Learning, Beijing, China, 2014.

[2] Bengio Y, Schwenk H, Sencal J S, et al. Neural probabilistic language models[M]//Innovations in Machine Learning. Springer Berlin Heidelberg, 2006: 137-186.

[3] Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality[C]//Advances in Neural Information Processing Systems. 2013: 3111-3119.

[4] Yan X, Guo J, Lan Y, et al. A biterm topic model for short texts[C]//Proceedings of the 22nd international conference on World Wide Web. International World Wide Web Conferences Steering Committee, 2013: 1445-1456.

[5] Blei D M, Ng A Y, Jordan M I. Latent dirichlet allocation[J]. the Journal of machine Learning research, 2003, 3: 993-1022.

[6] Hofmann T. Probabilistic latent semantic analysis[C]//Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc., 1999: 289-296.

[7] Zelikovitz S, Hirsh H. Transductive LSI for Short Text Classification Problems[C]//FLAIRS conference. 2004: 556-561.

[8] Jain V, Mahadeokar J. Short-text representation using diffusion wavelets[C]//Proceedings of the companion publication of the 23rd international conference on World wide web companion. International World Wide Web Conferences Steering Committee, 2014: 301-302.

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

[10] Chen M, Jin X, Shen D. Short text classification improved by learning multi-granularity topics[C]//IJCAI. 2011: 1776-1781.

[11] Shen, Dou, et al. Text classification improved through multigram models. Proceedings of the 15th ACM international conference on Information and knowledge management. ACM, 2006.

[12] Joachims T. Text categorization with support vector machines: Learning with many relevant features[M]. Springer Berlin Heidelberg, 1998.

[13] Yu H, Ho C, Juan Y, et al. Libshorttext: a library for short-text classification and analysis[R]. Technical Report. http://www. csie. ntu. edu. tw/ cjlin/ papers/libshorttext. pdf, 2013.

[14] Fan R E, Chang K W, Hsieh C J, et al. LIBLINEAR: A library for large linear classification[J]. The Journal of Machine Learning Research, 2008, 9: 1871-1874.

[15] Socher R, Perelygin A, Wu J Y, et al. Recursive deep models for semantic compositionality over a sentiment treebank[C]//Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP). 2013: 1631-1642.

[16] Sun X, Wang H, Yu Y. Towards effective short text deep classification[C]//Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval. ACM, 2011: 1143-1144.

[17] Wang P, Zhang H, Wu Y F, et al. A robust framework for short text categorization based on topic model and integrated classifier[C]//Neural Networks (IJCNN), 2014 International Joint Conference on. IEEE, 2014: 3534-3539.

[18] Bishop C M. Pattern recognition and machine learning[M]. New York: springer, 2006.