2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing

Sentiment Analysis Using Convolutional Neural Network

Xi Ouyang[1], Pan Zhou[2]
School of Electronic Information and Communications
Huazhong University of Science & Technology
Wuhan, China
Email: ouyang@hust.edu.cn[1], panzhou@hust.edu.cn[2]

Cheng Hua Li
DNN (Deep Neural Network) Lab
Beijing JingDong century trade .ltd (JD.COM)
Beijing, China
Email: lchbidatech@163.com

Lijun Liu
Wuhan TipDM Intelligent Technology
Wuhan, China
Email: liulijun9@hotmail.com

***Abstract*—Sentiment analysis of text content is important for many natural language processing tasks. Especially, as the development of the social media, there is a big need in dig meaningful information from the big data on Internet through the sentiment analysis. Inspired by the successes of deep learning, we are interested in handling the sentiment analysis task using deep learning models. In this paper, we propose a framework called Word2vec + Convolutional Neural Network (CNN). Firstly, we use the word2vec proposed by Google to compute vector representations of words, which will be the input for the CNN. The purpose of using word2vec is to gain the vector representation of word and reflect the distance of words. That will lead to initialize the parameters at a good point of CNN, which can efficiently improve the performance of the nets in this problem. Secondly, we design a suitable CNN architecture for the sentiment analysis task. We use 3 pairs of convolutional layers and pooling layers in this architecture. To the best of our knowledge, this is the first time that a 7-layers architecture model is applied using word2vec and CNN to analyze sentences' sentiment. We also use the Parametric Rectified Linear Unit (PReLU), Normalization and Dropout technology to improve the accuracy and generalizability of our model. We test our framework in a public dataset which is the corpus of movie review excerpts that includes fives labels: negative, somewhat negative, neural, somewhat positive and positive. Our network achieves test accuracy of 45.4% in this dataset, which is a better performance than some other neural network model like Recursive Neural Network (RNN) and Matrix-Vector Recursive Neural Network (MV-RNN).**

***keywords*—sentiment analysis; deep learning; word2vec;**

I. Introduction

The task of natural language processing includes sentiment analysis, paraphrase detection, entailment recognition, summarisation, discourse analysis, machine translation, grounded language learning and image retrieval. Sentiment analysis is one of the important parts of understanding how human being deal with text information and has the wide application in the industry. For instance, a famous Chinese internet company, Baidu provides a new server to help people select better products. This service will show people the sentiment analysis of the product by using the data of people' reviews in some online shopping sites. If we can reduce the error and complexity of sentiment analysis, this service will be more helpful. In the past decades, the approach to sentiment analysis is using traditional classification models such as Naive Bayes (NB), Naive Bayes with bag of bigram features (BiNB) and support vector machines (SVMs).

Recently, deep learning has achieved impressive results in vision, speech recognition, and elsewhere. In the field of sentiment analysis, there has already been researchers using deep learning model to gain a better performance. As the most famous example, Socher [10] used a deep learning model called Recursive Neural Network (RNN), who used a fully labeled parse trees to represent the movie reviews from the *rottentomatoes.com* website. The nodes of the tree are words or the groups of words, which means that we not only need the labels of the whole sentences, but also get the labels of each word and the group of words. If the dataset is too big, the work of labeling data will be huge. To reslove this issue, we decide to use other deep learning models. As we konw, there are several popular and effective models in deep learning. Compared to other deep models, Convolutional Neural Network (CNN) has much fewer connections and parameters and they are easier to train. Moreover, CNN is a powerful deep model in the field of understanding image content. And it's becoming increasingly popular for many classification problems, with the success of ImageNet classification with ConvNets [1]. Because of the great performance of CNN, we have reason to believe that CNN can be applied to the field of sentiment analysis. Another advantage of using CNN is that we just need to label the whole sentences artificially, which avoids the huge work in RNN. The key enabling factors behind this success is how to adjust CNN to deal with the non-image data well. We will discuss more details in our experiment part.

Another important thing is that just using a single deep learning method often does not bring the best results, which indicates a combination of deep learning and other pre-training methods should lead to a higher accuracy rate. Word2vec [2][3][4] is a neural net that processes text before that text is handled by deep-learning algorithms. The word2vec tool takes a text corpus as input and produces the word vectors as output.

978-1-5090-0154-5/15 $31.00 © 2015 IEEE
DOI 10.1109/CIT/IUCC/DASC/PICOM.2015.349

It first constructs a vocabulary from the training text data and then learns vector representation of words. The resulting word vector file can be used as features in the CNN. So we use the word vector file produced by word2vec tool as the input data for the CNN instead of initializing all word vectors by randomly sampling each value from a uniform distribution. When two words' meaning is similar, the vectors' value of them is closed too. Then if the word's sentiment is similar, we will gain the similar vector input for the CNN. This will improve the possibility of getting the same label when two sentences have similar words.

Thus, in this paper, we use a framework of Word2vec + CNN. Firstly, we use the word2vec to transform the words to the vectors, so that we can build up the sentences' vectors. Secondly, in order to classify the sentences to different sentiment labels, we input the sentences' vectors to CNN. We build our convolutional neural network in the open source framework called Caffe. Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by the Berkeley Vision and Learning Center (BVLC) and by community contributors. Yangqing Jia created the project during his PhD at UC Berkeley [5]. We can build a CNN conveniently and quickly by using Caffe. On the other hand, there is a HDF5 layer in Caffe, which will be convenient to deal with the text data.

The remainder of this paper is organized as follows. Section II reviews the related work on sentiment analysis and the development of Convolutional Neural Network. The proposed architecture is described in Section III. In the next, various experimental results and analysis are presented in Section IV. In Section V, we make some conclusion.

In the end, we train our network in a NVIDIA Tesla K20Xm GPU. This network won't take too much time to train, while we use Caffe in GPU.

II. RELATED WORK

Sentiment analysis of online user generated content is important for many social media analytics tasks. In the context of social media, there are several challenges. First, there are huge amounts of data available. Second, messages on social networks are by nature informal and short. Therefore, sentiment analysis is a very challenging task [6][7]. Many different approaches to solve sentiment analysis have been developed by researchers from natural language processing and information retrieval, most of which in this field use bag of words representations [8].As the development of Internet, we can gain huge amounts of data in different web sites such as social media or online-shopping sites. Therefore, many researchers begin to use this available data to analyse sentiment in the social media data. Snyder and Barzilay [9] analysed the sentiment of multiple aspects of restaurants like food or atmosphere to deal with larger reviews in more detail.

The deep learning framework is very powerful in both supervised and unsupervised learning fields. Because of the recent achievement of deep learning, more and more researchers are trying to solve the challenging sentiment analysis task using deep learning algorithms.

A famous deep learning framework provided by Socher [10] is the Recursive Neural Network (RNN). Socher used a fully labeled parse trees to represent the movie reviews from the *rottentomatoes.com* website. Recursive neural models compute parent node vectors use the two children's node vectors to compute the parent vectors. Finally, the root node of the parse tree can represent the sentence. This is the simplest member of this family of the Recursive Neural Network proposed by Scocher. In 2013 [11], Socher proposed a improved Recursive Neural Network called Recursive Neural Tensor Network (RNTN). The main idea is to use the same, tensor-based composition function for all nodes. In other words, they take into account the distance of the word in a sentence by adding a parameter to the models. However, there is a disadvantage in both RNN and RNTN. Both of them need to a fully labeled parse trees to train the neural network, so it will be a lot of heavy work to gain the each word and group of words. On the contrary, Convolutional Neural Network only need the labels of the whole sentences instead of the fully labeled parse tree. At the same time, CNN have much fewer connections and parameters and they are easier to train.

Convolutional Neural Network [12] have been proved very effective in solving the tasks related to computer vision. CNN contains the convolution layers and pooling layers. In a general situation, the convolutional is followed by the pooling layers which combine the outputs of neuron clusters [13][14]. However, when given more complex problems, the breadth and depth of CNN will continue to increase which would become limited by computing resources. Recently, thanks to the increasing efficient GPU computing, training a large deep convolutional neural network become possible. In 2012, Dan Ciresan et al. significantly improved upon the best performance in the literature for multiple image databases, including the MNIST database, the NORB database, the HWDB1.0 dataset (Chinese characters), the CIFAR10 dataset (dataset of 60000 32x32 labeled RGB images) [15], and the ImageNet dataset [16]. CNN models have subsequently been shown to be useful for the sentiment analysis problem. N. Kalchbrenner described a Dynamic Convolutional Neural Network (DCNN) [17] for the semantic modelling of sentences. The network used Dynamic k-Max Pooling, a global pooling operation over linear sequences.

Our work is inspired by [11] and [17], where we pre-train a 7-layers CNN model using the word2vec output vectors in order to represent the distance of each word. To the best of our knowledge, this is the first time that a 7-layers architecture model is applied using word2vec and CNN to analyze the sentences' sentiment. The reason why we choose a 7-lays model is that we balance the computational complexity and output accuracy from many experiments.

III. WORD2VEC + CNN

In our framework, we use the word2vec to gain the vectors for the words as the input. Word2vec is a neural net that

processes text before that text is handled by deep-learning algorithms. In this problem, what we want to do is to classify the sentences with CNN, but CNN can't understand the sentences directly as a human. To deal with it, word2vec translates text into the vector that CNN can understand. At the same time, the vector produced by the word2vec can represent the distance of words. When two words' meaning is similar, the vectors' value of them is closed too. A string of sentences is expected by word2vec as its input. Each sentence, which is each array of words, is transformed to vectors and compared to other vectors' lists of words in an n-dimensional vector space. Related words or groups of words appear next to each other in that space. When we gain the vectors of words, it allows us to measure their similarities with some exactitude and cluster them. CNN will have a better performance because of it.

A. Word2vec

Word2vec is a tool proposed by Google in 2013. It's open source and follows the Apache License 2.0 open source license. Word2vec creates features without human intervention, which can also work very well even the input is the text of individual words. Given enough data, usage and context, word2vec can make a highly accurate guess about a word's meaning. Another important advantage of word2vec is that word2vec runs fast even in a big dataset. For the purpose of deep learning, a word's meaning is a simple signal that helps to classify larger entities.

In this paper, We use the pre-trained vectors trained on part of Google News dataset(about 100 billion words) [2]. The vectors can be download freely in *https://code.google.com/p/word2vec/*. The model contains 300-dimensional vectors for 3 million words and phrases. We gain the more precise relation of words from such a big corpus. For examples, the top most related terms are associated with "bad", "good", etc as shown in TABLE I. Those examples will average vectors for words and will return the closest neighbors to the resulting vector. In Fig. 2, we visualize a diffusion network to represent the relationship of the word in the Google News dataset. The node represents the word in the Google News dataset. The edge width represents the strength of the word similarity. From the graph, we can clearly see that the similar word will be clustered in the same group. For example, the positive words, "good", "wonderful", and "fantastic" disappear in the same side of the graph, while the negative words like "bad" disappear in the other side of the graph. From this graph, we can see the word2vec tool can accurately describe the similarity of the words, so the same sentiment label words' vectors will be similar. That will definitely train the CNN in a good initial staring point.

Moreover, we have to consider a question is some word we will use is not in this vector file. Though we use the Google News dataset which is a big corpus, there will be some word that didn't disappear in this corpus. In this case, we still need to get the vectors of those words. Thus, we will initialize the words randomly which are not present in the set of pre-trained words. Those words vectors will be 300-dimensional.

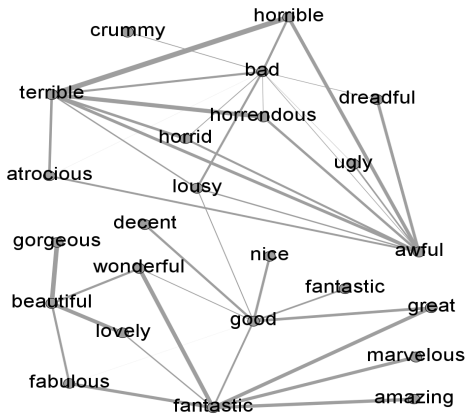

Fig. 2: Diffusion network to represent the relationship of the word in the Google News dataset. The node represent the word in the Google News dataset. The edge width represents the strength of the word similarity.

Another important thing is that the input data of the CNN we will use in next is expected of the same size, which means we must input the sentences that have the same number of words. The max number of words in all the sentences is 53, so we set the size of each sentence to be 53. We will add the zero vectors in the end of the sentences that are less than 53 words. This sentence matrix s, where each column corresponds to the feature vector $w_i \in R^d$ of a word in the sentence:

$$s = \begin{pmatrix} \vdots & \dots & \vdots \\ w_1 & \vdots & w_{53} \\ \vdots & \dots & \vdots \end{pmatrix}. \quad (1)$$

B. CNN

After we get the pre-trained word vectors from word2vec, we will train a convolutional neural network. In this paper, we use 3 convolutional layers and 3 pooling layers. The layer configuration of our CNN is schematically shown in Fig. 1. Our experiment is derived from the publicly available Caffe toolbox [5] based on one NVIDIA Tesla K20Xm GPU. Here, we use convolutional layers, pooling layers, Parametric Rectified Linear Unit (PReLU) layers and dropout layers in CNN. In the architecture of CNN, the most time of training the neural network is spent in the convolution. Meanwhile, the full-connected layer takes up most of the parameters of the

TABLE I: The example output of ./distance GoogleNews-vectors-negative300.bin.

bad	terrible	horrible	lousy	crummy	horrid
	0.682861	0.670260	0.664764	0.567782	0.565169
good	great	decent	nice	excellent	fantastic
	0.729151	0.683735	0.683609	0.644293	0.640778
awful	horrible	terrible	dreadful	horrendous	horrid
	0.759767	0.747892	0.721818	0.697003	0.672018
beautiful	gorgeous	lovely	wonderful	fabulous	loveliest
	0.835300	0.810064	0.685409	0.670007	0.661258
terrible	horrible	horrendous	dreadful	awful	horrid
	0.924392	0.846727	0.802277	0.747891	0.717903
fantastic	wonderful	great	amazing	marvelous	fabulous
	0.804792	0.793521	0.778987	0.768760	0.760597

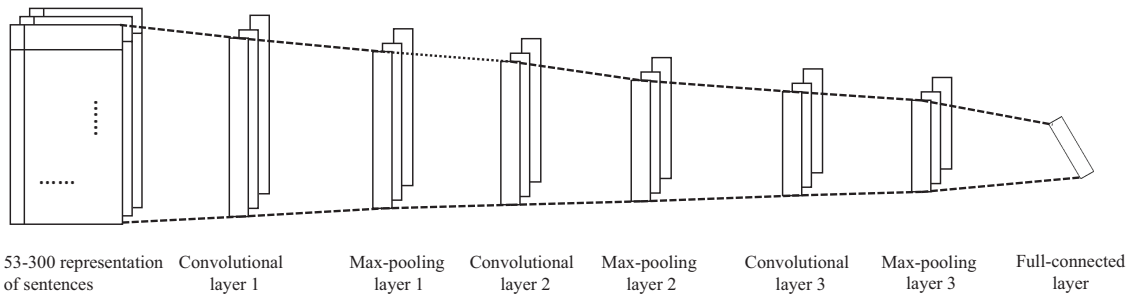

Fig. 1: The architecture of our CNN. The size of network's input is 53×300.

network. The main aim of convolution is to extract the input feature, and pooling is to sample the convolution matrix.

When training a neural network model, Dropout will be an important trick which is perhaps a biggest invention in the field of neural networks in recent years proposed by Hinton [21]. It solves the main problem in machine learning, that is over-fitting. Dropout is a recently introduced algorithm for training neural networks by randomly dropping units during training to prevent their co-adaptation. Dropout consists of setting to zero the output of each hidden neuron with probability of 0.5. The algorithm will drop out the neurons which don't contribute to the forward pass and don't participate in backpropagation. For each training example, a different set of units to drop is randomly chosen. In our framework, we use all the neurons but multiply their outputs by 0.5 after the convolutional layers.

1) Initializing CNN: We use the mini-batch stochastic gradient desent with momentum set to 0.9 and weight decay set to 0.0005 to initialize the network and train our model using stochastic gradient descent with a batch size of 10 examples. Then the update rule for weigh w was:

$$v_{i+1} := 0.9 \cdot v_i - 0.0005 \cdot \varepsilon \cdot w_i - \varepsilon \cdot \langle \frac{dL}{dw}|_{w_i} \rangle_{D_i}, \quad (2)$$

$$w_{i+1} := w_i + v_{i+1}, \quad (3)$$

where i is the iteration index, v is the momentum variable, ε is the learning rate, and $\langle \frac{dL}{dw}|_{w_i} \rangle_{D_i}$ is the average over the ith batch D_i of the derivative of the objective with respect to w, evaluated at w_i.

In each layer, we initialized the weights from a zero-mean Gaussian distribution with standard deviation 0.01. We used an equal learning rate for all layers, where Caffe adjusted automatically throughout training. We initialize the learning rate at 0.001 and divide the learning rate by 10 every 800 iterations. We show the details in TABLE II.

TABLE II: Details of the initialization parameters. Base lr: the initialization learning rate. Gamma and step size: the original learning rate will be divided by 10 every 800 iterations. Momentum: the weight of the previous update. Solver mode: computer will train the CNN in CPU/GPU mode.

Base lr	Gamma	Step size	Max iter	Momentum	Solver mode
0.001	0.1	800	14500	0.9	GPU

IV. EXPERIMENTS

We evaluate our algorithm and train our convolutional neural network on the public dataset, which is availabled on *http://nlp.stanford.edu/sentiment/*. We then relate the results of the experiments and we compare them with some previous machine learning algorithms.

A. Dataset

This public dataset is the corpus of movie review excerpts form the rottentomatoes.com website originally collected and published by Pang and Lee [18]. There are 10662 sentences of movie reviews, half of which was positive and the other half negative in the original dataset. The overall sentiment of the long movie review can be reflected by the label. Meanwhile, they have deleted the HTML tags and the parts that are not in English. In 2013, R. Socher did an extension of the movie review dataset and gain the 5-class classification: negative, somewhat negative, neutral, positive or somewhat positive [11]. He used the Amazon Mechanical Turk to relabel the sentences' sentiment, which is one of the sites of Amazon Web Services to enables individuals to coordinate the use of humman intelligence to perform tasks that computers are currently unable to do. We use 9645 sentences to train the CNN and test it by 2210 sentences. TABLE III are the details of this public dataset. Just as we mentioned above, we will translate the context of the dataset to the vectors which is the language computer can understand. The details of the process are shown in Fig. 3. This graph will show that how we get the sentence vectors from the original dataset.

TABLE III: Details of the dataset. c: Number of classification. $l_{\max}$: Maximum sentence length. N: Number of sentences in this dataset. $|V|$: Vocabulary size. $|V_{pre}|$: Number of words present in the set of Google-News word vectors. Test: Number of sentences in test set.

Data	c	$l_{\max}$	N	$\|V\|$	$\|V_{pre}\|$	Test
Review	5	53	11855	17833	16262	2210

B. Result and Discussion

The public framework Caffe has a wide application in dealing with the image problems. Many famous CNN framework have been built in Caffe, such as AlexNet, LeNet and

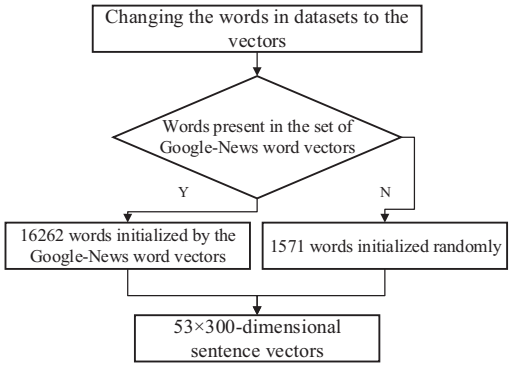

Fig. 3: Details of the process to the datasets.

GoogleNet. Meanwhile we can use our own pictures as the input of Caffe and add the layers like convolutional layers or pooling layers conveniently and quickly. In order to build a CNN in Caffe, we just need to some configuration profiles instead of rewrite large codes. In this experiment, we apply Caffe in the natural language processing, which is different from the the previous problem. To solve this problem, we save our dataset to HDF5 to feed vectors to Caffe, which is a unique technology suite that makes possible the management of extremely large and complex data collections.

TABLE IV: Neural network size and parameters.

Data sets	Input size	Hidden layer numbers	Output label numbers	Iterations	Momentum
Movie reviews	53×300	6	5	14500	0.9

In this paper, the architecture of the CNN we employ for sentiment analysis is shown in Fig. 2. Each sentences is resized to 53 $\times$ 300. The input is processed by three convolutional layers. The first two convolutional layer is also followed by Parametric Rectified Linear Unit (PReLU) layers [20], dropout layers, max-pooling layers and normalization layers. The first convolutional layer has 50 kernels of size 4 $\times$ 5 with a stride of 5. Next, there is a PReLU layer in order to improve model fitting with nearly zero extra computational cost and little overfitting risk. In the dropout layer, we set the dropout regularisation radio to 0.5 to reduce complex co-adaptations of neurons in nets. Then, there is the max-pooling layer whose kernel size is 2 $\times$ 2 and stride is 2. The second convolutional layer has 100 kernels of size 5 $\times$ 6. Similarly, there are the PReLU layer, dropout layer, max-pooling layer and normalization layer after the second convolutional layer. After these layers, we have the last pair of the convolutional layer and max-pooling layer. Furthermore, we have only one fully connected layer in the net. The neural networks' size and parameters are given in TABLE IV.

The last layer is designed to learn the parameter w by minimizing the loss function ($J(w)$)'s value using a softmax classifier ($x^{(i)}$ and $y^{(i)}$ are the feature vector and label for the i-th instance respectively):

$$J(w) = -\frac{1}{m}[\sum_{i=1}^{m}\sum_{j=0}^{1} 1\{y^{(i)} = j\} \log p(y^{(i)} = j|x^{(i)}; w)], \tag{4}$$

where

$$p(y^{(i)} = j|x^{(i)}; w) = \frac{e^{w_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{w_j^T x^{(i)}}}. \tag{5}$$

We test our network on the public dataset (TABLE III). We compare to some famous methods like Naive Bayes (NB), Support Vector Machine (SVM), Naive Bayes with bag of bigram features (BiNB) and a model that averages neural word vectors and ignores word order (VecAvg). We also compare it to some neural network model like Recursive Neural Network (RNN) [10] and Matrix-Vector Recursive Neural Network (MV-RNN) [19]. The RNN computes each node in the parse tree after words are first mapped into a semantic space and then they are merged into phrases in a syntactically and semantically meaningful order. The main idea of the MV-RNN is to represent every word and longer phrase in a parse tree as both a vector and a matrix. When two constituents are combined the matrix of one is multiplied with the vector of the other and vice versa. Hence, the compositional function is parameterized by the words that participate in it.

TABLE V: Accuracy for fine grained (5-class) to the sentences' sentiment. NB: Naive Bayes. SVM: Support Vector Machine. BiNB: Naive Bayes with bag of bigram features. VecAvg: a model that averages neural word vectors and ignores word order. RNN: Recursive Neural Network from Socher et al., 2011a [10]. MV-RNN: Matrix-Vector Recursive Neural Network from Socher et al., 2012 [19].

Model	Fine-gain(%)
NB	41.0
SVM	40.7
BiNB	41.9
VecAvg	32.7
RNN	43.2
MV-RNN	44.4
Our Framework	45.4

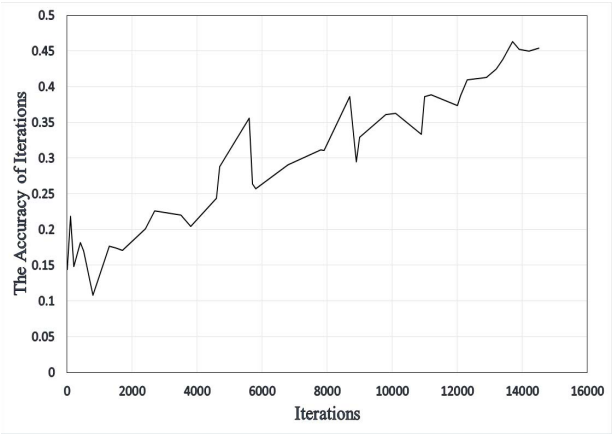

Fig. 4: The Convolutional Neural Network accuracy on the data set.

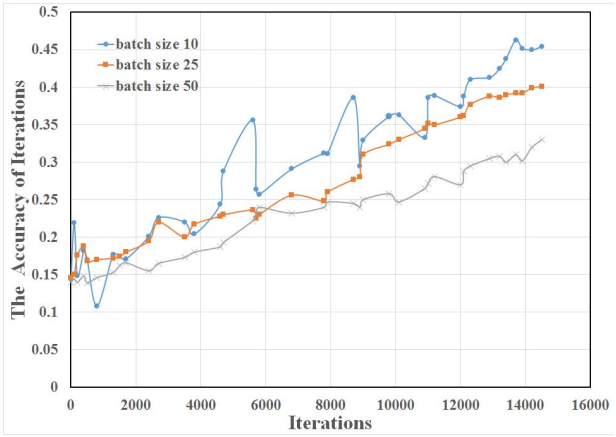

Fig. 5: Influence of different batch size.

In the case of the CNN, we can see in Fig. 4 that, at the beginning of the training phase,the accuracy is increased very rapidly. But accuracy increment slows down after a certain number of iterations, and after 14500 iterations we achieve test accuracy of 45.5%. In Fig. 5, we compare influence of different batch size. It is clear that 10 batch size will be the most suitable for this neural network.

TABLE V shows the result of our experiment. The output is a fine grained classification into 5 classes: negative, somewhat negative, neutral, positive or somewhat positive. TABLE VI shows that our framework gets the highest performance, followed by the MV-RNN and RNN. Our network achieves test accuracy of 45.4%.

V. CONCLUSION

Sentiment analysis is a challenging and attractive job. In this paper, we apply a convolutional neural network to solve this problem. We have described an interesting framework called Word2vec + CNN on the available public dataset of movie reviews. The experimental results suggest that convolutional neural networks that are properly trained can outperform the shallow classification algorithms. Meanwhile, the way of pre-trained + fine-tuning is adopted to train the convolutional neural network on the natural language processing task, which is better than some other deep learning models like Recursive Neural Network and Matrix-Vector Recursive Neural Network.

In our model, we don't need to label every word in the sentences. What we need are the labels of the whole sentences. The larger datasets, along with the proposed deep CNN and its training strategies, give rise to better generalizability of the trained model and higher confidence of such generalizability. We believe that sentiment analysis is more and more important since we have more and more large scale online user generated content. There is a big need for using social media information for prediction and forecasting. We can dig a lot of valuable information by the data sentiment analysis, which will have a wide application like public opinion analysis, product reviews analysis and so on. We believe that our sentiment analysis results can give some further inspiration to other researchers.

ACKNOWLEDGMENT

This research is supported by NSFC No. 61401169.

REFERENCES

[1] Krizhevsky A, Sutskever I, Hinton G E, "Imagenet classification with deep convolutional neural networks," In Neural Information Processing Systems, pp. 1097-1105, 2012.
[2] Mikolov T, Chen K, Corrado G, et al, "Efficient estimation of word representations in vector space," arXiv preprint arXiv:1301.3781, 2013.
[3] Mikolov T, Sutskever I, Chen K, et al, "Distributed representations of words and phrases and their compositionality," In Neural Information Processing Systems, pp. 3111-3119, 2013.
[4] Mikolov T, Yih W, Zweig G, "Linguistic Regularities in Continuous Space Word Representations," In HLT-NAACL, pp. 746-751, 2013.
[5] Jia Y, Shelhamer E, Donahue J, et al, "Caffe: Convolutional architecture for fast feature embedding," In Proceedings of the ACM International Conference on Multimedia. ACM, pp. 675-678, 2014.
[6] Liu B, Dai Y, Li X, et al, "Building text classifiers using positive and unlabeled examples," In Data Mining (ICDM), 2003. IEEE 3th International Conference on. IEEE, pp. 179-186, 2003.
[7] Li G, Hoi S C H, Chang K, et al, "Micro-blogging sentiment detection by collaborative online learning," In Data Mining (ICDM), 2010. IEEE 10th International Conference on. IEEE, pp. 893-898, 2010.
[8] Pang B, Lee L, "Opinion mining and sentiment analysis," Foundations and Trends in Information Retrieval, 2.1-2: pp. 1-135, 2008.
[9] Snyder B, Barzilay R, "Multiple Aspect Ranking Using the Good Grief Algorithm," In HLT-NAACL, pp. 300-307, 2007.
[10] Socher R, Lin C C, Manning C, et al, "Parsing natural scenes and natural language with recursive neural networks," In Proceedings of the 28th International Conference on Machine Learning (ICML), pp. 129-136, 2011.
[11] Socher R, Perelygin A, Wu J Y, et al, "Recursive deep models for semantic compositionality over a sentiment treebank," In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1631-1642, 2013.
[12] LeCun Y, Bottou L, Bengio Y, et al, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, 1998.
[13] Ciresan D C, Meier U, Masci J, et al, "Flexible, high performance convolutional neural networks for image classification," In IJCAI Proceedings-International Joint Conference on Artificial Intelligence, vol. 22, no. 1, pp. 1237-1242, 2011.
[14] Krizhevsky A, Sutskever I, Hinton G E, "Imagenet classification with deep convolutional neural networks," In Neural Information Processing Systems, pp. 1097-1105, 2012.
[15] Ciresan D, Meier U, Schmidhuber J, "Multi-column deep neural networks for image classification," In Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. IEEE, pp. 3642-3649, 2012.
[16] Deng J, Dong W, Socher R, et al, "Imagenet: A large-scale hierarchical image database," In Computer Vision and Pattern Recognition (CVPR), 2009 IEEE Conference on. IEEE, pp. 248-255, 2009.
[17] Kalchbrenner N, Grefenstette E, Blunsom P, "A convolutional neural network for modelling sentences," arXiv preprint arXiv:1404.2188, 2014.
[18] Pang B, Lee L, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," In Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, pp. 115-124, 2005.
[19] Socher R, Huval B, Manning C D, et al, "Semantic compositionality through recursive matrix-vector spaces," In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. Association for Computational Linguistics, pp. 1201-1211, 2012.
[20] Chatfield K, Simonyan K, Vedaldi A, et al, "Return of the devil in the details: Delving deep into convolutional nets," arXiv preprint arXiv:1405.3531, 2014.
[21] Hinton G E, Srivastava N, Krizhevsky A, et al, "Improving neural networks by preventing co-adaptation of feature detectors," arXiv preprint arXiv:1207.0580, 2012.