An Ensemble Method with Sentiment Features and Clustering Support

Nguyen Huy Tien
Japan Advanced Institute of
Science and Technology (JAIST)
ntienhuy@jaist.ac.jp

Nguyen Minh Le
Japan Advanced Institute of
Science and Technology (JAIST)
nguyenml@jaist.ac.jp

Abstract

Deep learning models have recently been applied successfully in natural language processing, especially sentiment analysis. Each deep learning model has a particular advantage, but it is difficult to combine these advantages into one model, especially in the area of sentiment analysis. In our approach, Convolutional Neural Network (CNN) and Long Short Term Memory (LSTM) were utilized to learn sentiment-specific features in a freezing scheme. This scenario provides a novel and efficient way for integrating advantages of deep learning models. In addition, we also grouped documents into clusters by their similarity and applied the prediction score of Naive Bayes SVM (NB-SVM) method to boost the classification accuracy of each group. The experiments show that our method achieves the state-of-the-art performance on two well-known datasets: IMDB large movie reviews for document level and Pang & Lee movie reviews for sentence level.

1 Introduction

The emergence of web 2.0, which allows users to generate content, is causing the rapid increase in the amount of data. This platform, which enables millions of users to share information and comments, has a high demand for extracting knowledge from user-generated content. An important information to be analyzed from those comments is opinions/sentiments, which express subjective opinions of particular users. Sentiment analysis is a fundamental task and has attracted a huge amount of research in recent years (Pang and Lee, 2008; Liu, 2012). The task calls for identifying the sentiment polarity (positive, negative) of a comment or review.

Wang (2012) used a Support Vector Machine variant with Naive Bayes feature (NBSVM). Presenting a document or a sentence with Bag of bi-gram features, NBSVM consistently performs well across datasets of long and short reviews. Recently, the success of deep learning in natural language processing has led to many efficient methods for sentiment analysis such as Paragraph Vector (Le and Mikolov, 2014), CNN (Kalchbrenner et al., 2014; Kim, 2014; Zhang and Wallace, 2015), LSTM (Wang et al., 2015; Liu et al., 2015). In Paragraph Vector, Le and Mikolov employed the technique of Word embedding representation using neural networks (Bengio et al., 2003; Collobert and Weston, 2008; Mnih and Hinton, 2009; Turian et al., 2010; Mikolov et al., 2013) to represent a document or paragraph as a vector. This document modeling outperformed the Bag of Words model in sentiment analysis and information retrieval. Li (2015) has enhanced the architecture of Paragraph Vector by allowing the model to predict not only words but also n-gram features (DVngram). CNN is capable of capturing local relationships between neighbor words in a sentence but fails for long-distance dependencies. LSTM can handle CNN's limitation because it is able to memorize information for a long period of time. Our motivation is to build a combination approach taking the advantages of these methods.

In this paper, we separately designed CNN and LSTM models to encode sentiment information into feature vectors. To apply for sentiment classification, these sentiment-specific vectors and the semantic-specific DVngram vector were passed into the 3-layer neural network. In sentiment analysis, two sentences with a slight difference could provide opposite sentiments. Generative models, however, have a tendency to encode

Proceedings of the The 8th International Joint Conference on Natural Language Processing, pages 644–653,
Taipei, Taiwan, November 27 – December 1, 2017

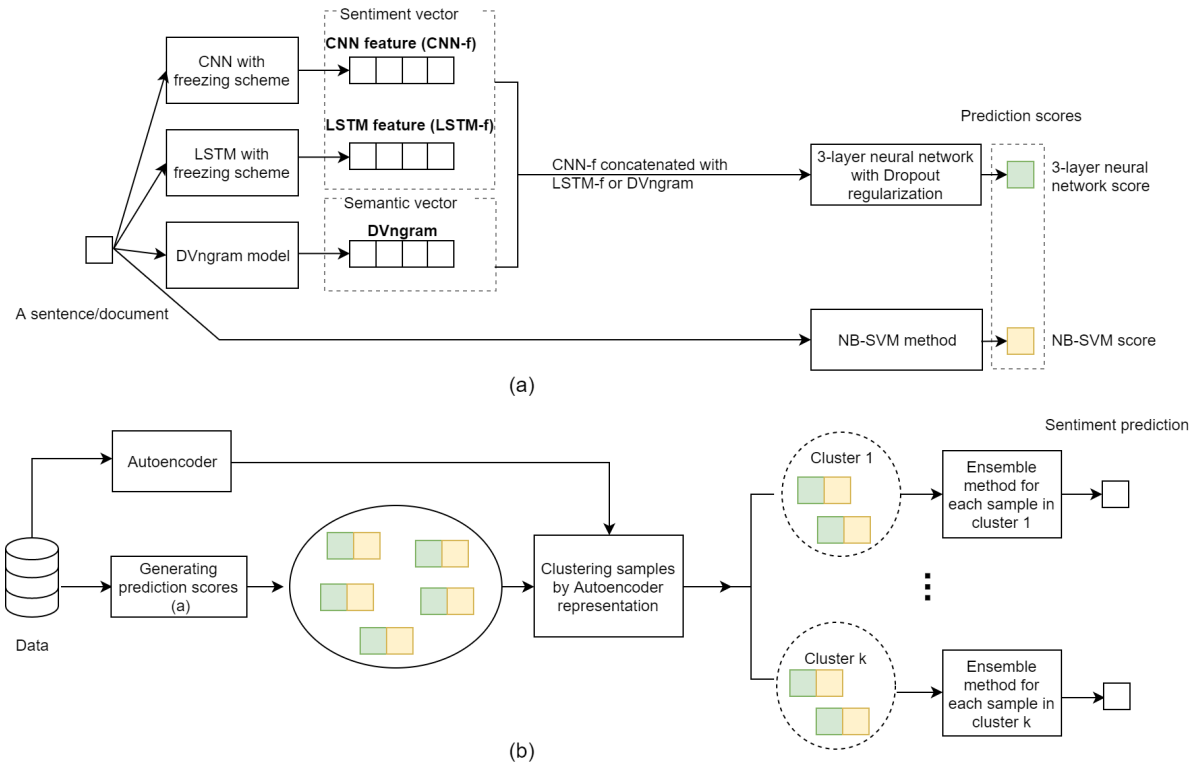


Figure 1: The proposed framework for sentiment analysis

similar sentences/documents into similar vectors. For that reason, we designed an autoencoder model to learn representation vectors for sentences/documents and used these vectors for clustering. The prediction score of NBSVM method is provided to enhance the sentiment prediction of each cluster. Figure 1 shows the architecture of our framework.

We compared our method with NBSVM, CNN, LSTM, Paragraph Vector, LinearEnsemble (Mesnil et al., 2014), DSCNN (Zhang et al., 2016) on three well-known datasets: IMDB large movie reviews (Maas et al., 2011) for document level, Pang & Lee (2005) movie reviews and Stanford Sentiment Treebank (Socher et al., 2013) for sentence level. The experimental results show that our method consistently performs well on both document and sentence level data. The main contributions of this work are as follows:

- We applied a freezing scheme to CNN and LSTM models for encoding sentiment information into vectors. These vectors provide a simple and efficient way to integrate the strong abilities of deep learning models.

- We proposed a scenario to divide data into groups of similar sentences/documents. Then, each sentence/document in each group is represented by the prediction score of NB-SVM method and the prediction score of the proposed 3-layer neural network. We proposed an ensemble method to employ these scores.

2 Sentiment representation learning

In this section, we describe the freezing scheme to generate sentiment vectors from two models: (i) CNN, (ii) LSTM; and a method to employ these vectors. To feed into LSTM/CNN model, each word of a sentence/document is transformed into a word embedding vector using *Word2Vec*[1].

Le and Mikolov (2014) extended the word embedding learning model by incorporating paragraph information. Given a paragraph, Le's method captures and encodes semantics into a representation vector or a semantic feature.

This work inspired us to develop a document representation learning model to capture sentiment information. In our work, we proposed an approach to generating sentiment representation from CNN and LSTM models. Our idea is to train CNN and LSTM models under the sentiment classification task. In a deep learning network, we

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

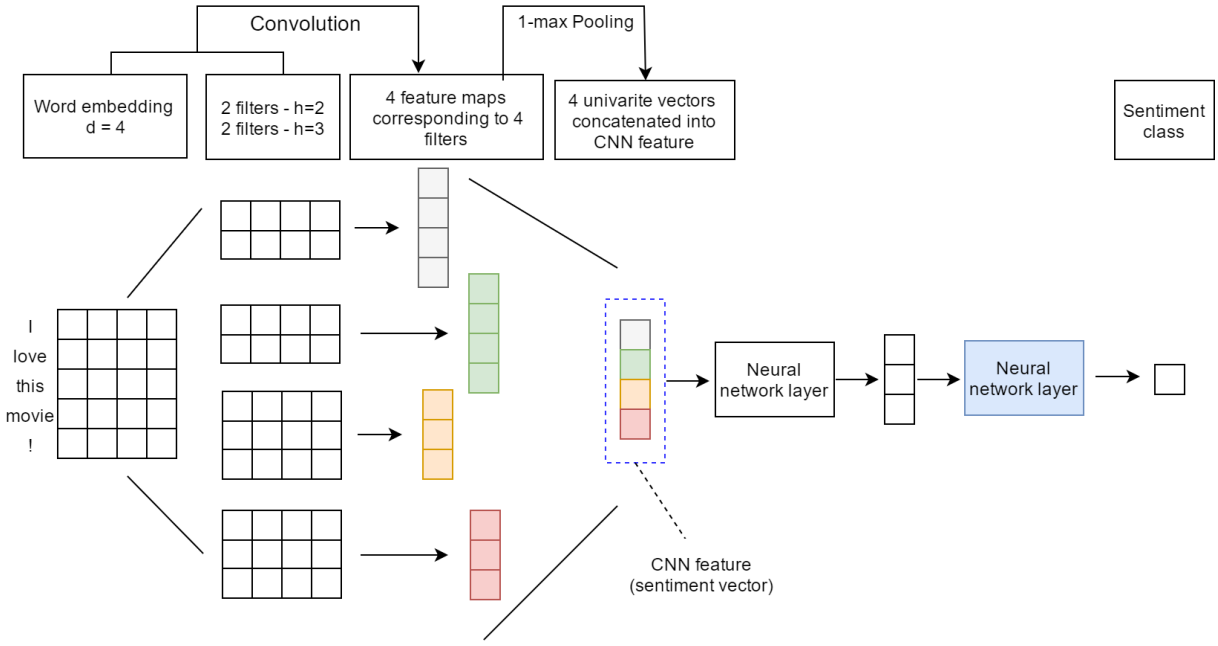


Figure 2: Illustration of our CNN framework to generate sentiment features. Given a sequence of d-dimension word embeddings ($d = 4$), the model applies 4 filters: 2 filters for region size $h = 2$ and 2 filters for region size $h = 3$ to generate 4 feature maps. During the training process, the parameters of the last neural network layer (blue one) are frozen (untrained)

could separate the model into two parts: (i) **Building target feature** - from input samples, the first part encodes target information into vectors, (ii) **Classifying layer** - the second part tries to learn a layer (or a boundary) for classifying these vectors into target labels. Sentiment vectors generated by a model, however, are much fitting to the classifying layer of this model. It is not efficient to combine two sentiment vectors generated from two models because each sentiment vector is fitting to its particular classifying layer. To address this problem, we proposed a freezing scheme. According to this scheme, the parameters of the classifying layer are initialized from the uniform distribution and in the training phase, these parameters are kept unchanged. This technique makes sentiment vectors not too fit to a particular classifying layer.

2.1 LSTM for sentiment feature engineering - LSTM feature

The LSTM architecture was introduced by Hochreiter (1997). By designing a memory cell preserving its state over a long period of time and non-linear gating units regulating information flow into and out of the cell, Hochreiter made LSTM able to capture efficiently long distance dependencies of sequential data without suffering the exploding or vanishing gradient problem of Recurrent neural network (Goller and Kuchler, 1996).

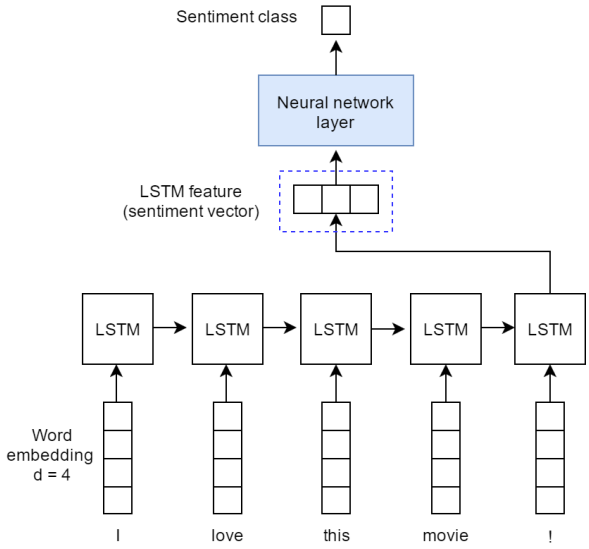


Figure 3: Illustration of our LSTM model to generate sentiment vectors. During the training process, the parameters of the neural network layer (blue one) are frozen (untrained)

Figure 3 explains how to employ the LSTM architecture for memorizing sentiment information

over sequential data. The model contains two parts: (i) **Building sentiment feature** - the LSTM layer encodes sentiment information of input into a fixed-length vector; (ii) **Classifying layer** - this sentiment-specific representation vector will be classified by the last neural network (NN) layer (the blue layer in Figure 3). As applying the freezing scheme, this NN layer's parameters are unchanged during the training process.

2.2 CNN for sentiment feature engineering - CNN feature

We present a sentence of length s as a matrix $d \times s$, where each row is a d-dimension word embedding vector of each word. Given a sentence matrix **S**, CNN performs convolution on this input via linear filters. A filter is denoted as a weight matrix W of length d and region size h. W will have $d \times h$ parameters to be estimated. For an input matrix $S \in \mathbb{R}^{d \times s}$, a feature map vector $O = [o_o, o_1, ..., o_{s-h}] \in \mathbb{R}^{s-h+1}$ of the convolution operator with a filter W is obtained by applying repeatedly W to sub-matrices of S:

$$o_i = W \cdot S_{i:i+h-1} \quad (1)$$

where $i = 0, 1, 2, ..., s-h$, $(\cdot)$ is dot product and $S_{i:j}$ is the sub-matrix of S from row i to j.

Each feature map O is fed to a pooling layer to generate potential features. The common strategy is 1-max pooling (Boureau et al., 2010). The idea of 1-max pooling is to capture the most important feature v corresponding to the particular feature map by selecting the highest value of that feature map:

$$v = \max_{0 \le i \le s-h} \{o_i\} \quad (2)$$

We have described in detail the process of one filter. Figure 2 shows an illustration of applying multiple filters with variant region sizes to obtain multiple 1-max pooling values. After pooling, these 1-max pooling values from feature maps are concatenated into a CNN feature carrying sentiment information. Intuitively, the CNN feature is a collection of maximum values from the feature maps. To make a connection to these values, we provide an NN layer to synthesize a high-level feature from the CNN feature. Finally, this high-level feature is passed to an NN layer with sigmoid activation to generate the probability distribution over sentiment labels.

In the training phase, similar to the strategy in our LSTM model, the last NN layer's parameters are kept untrained to make the sentiment vectors not too fit to a particular classifying layer.

2.3 Classifying with sentiment vectors

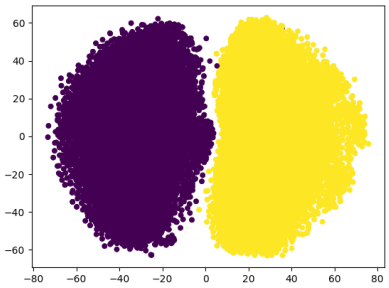
Figure 4 visualizes the results of encoding sentiments into vectors using our CNN model. As we can see in the development set, there are some unambiguous cases. Therefore, we add more information to CNN sentiment vectors by concatenating them with LSTM sentiment vectors or DVngram semantic vectors.

As CNN and LSTM sentiment vectors are, however, generated from models of sentiment classification, these vectors are easily separated in terms of sentimental categories by machine learning methods. In other words, a multi-layer NN sentiment classifier using both of these vectors as input reaches the state of perfect classification on the training set after a few epochs. In this case, the classifier's parameters are not efficiently optimized and the classifier's performance has no improvement on the testing set, compared with using LSTM or CNN for classification (or we call the model overfitting).

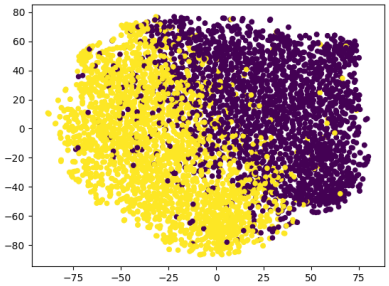
To address this problem, we employ a **3-layer NN** with Dropout regularization (Hinton et al., 2012) on the first and second layers. By randomly dropping out each hidden unit with a probability p on each presentation of each training case, Dropout prevents overfitting and provides a way to combine many variant NN architectures efficiently. By applying Dropout, our model has an ability to examine efficiently variant combination ways from feature vectors.

3 Ensemble with clustering support

As we discussed in Section 1, a slight difference between two sentences could lead to two opposite sentiments. However, similar sentences/documents have a tendency to be encoded into similar vectors by generative models. Therefore, it causes some difficulties in sentiment classification. To address this problem, we divided data into groups of similar sentences/documents and then provided an additional feature to boost the performance of classification in each group. For dividing data, we applied autoencoder models to encode word embedding representations of sentences/documents into fix-length vectors. These vectors then were used for clustering sentences/documents. For each sentence/document in



(a) Sentiment vectors in the train set



(b) Sentiment vectors in the development set

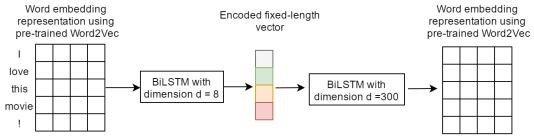
Figure 4: The t-SNE projection for IMDB dataset's sentiment vectors (positive and negative) generated from our CNN model.

each cluster, the prediction score of the method in section 2 are combined with the prediction score of NBSVM. The reason for choosing NBSVM is that NBSVM is an efficient method not based on neutral network architectures, and using Bag of Word model to represent sentences/documents, which is different from the word embedding representation. We consider NBSVM's score as an additional channel and expect it to support well for each group of similarity sentences/documents in terms of word embedding representation.

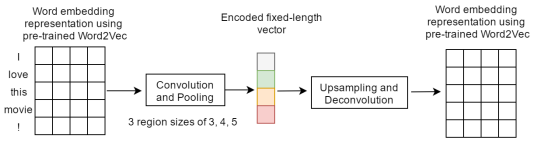
Given a sentence/document, we will have two prediction scores: one from the proposed method in section 2 and one from NBSVM. To employ these scores, we used a voting method. This scheme allows each classifier f_i to give a vote with a confident ratio r_i to the final probability score over classes distribution as follows:

$$p(c_i|x) = \frac{1}{N}\sum_{k=1}^{N} p_k(c_i|x)r_k \qquad (3)$$

where c_i is the ith sentiment class, N is the number of classifiers, $p_k(c_i|x)$ is the prediction score of the classifier k on the ith class for a sentence/document x.



(a) BiLSTM model



(b) CNN model. In MR-L dataset, each region size has 300 filters. In MR-S and SST dataset, each region size has 100 filters

Figure 5: Autoencoder models

The objective of this ensemble method is to select a suitable confident ratio for each classifier to optimize the performance of classification. In our approach, a 2-layer NN is employed to define a voting architecture. We consider a feedforward process in NN as a scheme of voting and the NN's weights as confident ratios. Adamax algorithm (Kingma and Ba, 2014) is applied to optimize the weights of the model.

Dataset	l	$train$	$test$	$\|V\|$
MR-L	300	25000	25000	169940
MR-S	20	10662	cv	18765
SST	19	9613	1821	16185

Table 1: Summary statistic of datasets. l denotes the average length of reviews, $train$ and $test$ are sizes of the training set and the test set respectively, cv is 10-fold cross validation, and $|V|$ is vocabulary size.

4 Dataset and Experiment setup

4.1 Dataset

We evaluated our models on three well-known datasets. Table 1 shows the statistic summary of datasets.

- For document level, IMDB large movie review dataset **MR-L** is used. Each review contains numerous sentences (Maas et al., 2011).
- For sentence level, Pang & Lee (2005) provided **MR-S** dataset having one sentence per

movie review. In addition, we also did experiment on Stanford Sentiment Treebank **SST** (Socher et al., 2013) - an extension of MR-S with two labels (positive and negative). In SST, all sentences and phrases of the training set are used for training.

4.2 Experimental setup

To tune hyper-parameters of our models, we do a grid search on 30% of each dataset.

- For MR-L:
  - LSTM model has dimension $d = 32$.
  - CNN model: using 3 region sizes of $3, 4, 5$; the number of each region size is 300 and the dimension of penultimate NN layer is 100.
  - 3-layer NN model for classification with sentiment vectors: the first NN layer has the same dimension as the input feature, and the dropout ratio $p = 0.9$; the second NN layer has the dimension of 64 and the dropout ratio $p = 0.5$.
  - Autoencoder models: we examined two autoencoder models - CNN and BiLSTM. The details are in Figure 5.
  - Clustering: k-mean is applied. The number of clusters is $k = 2$.
  - Ensemble model: the first NN layer has the dimension of $3 \times$ the input's dimension or the number of classifiers.
- For MR-S and SST: the same configuration as MR-L, except the number of each region size is 100.

For word vectors, we obtained pre-trained word vectors *Word2Vec*. Its vectors have the dimension of 300. In our LSTM and CNN models, these pre-trained word vectors are optimized during the training process.

5 Results and Discussion

We compared our models against the other methods showed in table 2 on the binary sentiment classification task. In SST dataset, we could not reproduce the result 88.1% of CNN (Kim, 2014). According to the empirical results, our method of combining feature vectors *3-layer NN* outperforms the individual methods: LSTM, CNN, and DVngram. That proves the efficiency of the feature combination strategy. In addition, our ensemble method with clustering support outperforms the current state of the art method on MR-L and MR-S datasets. As we mentioned in Section 3, NBSVM uses a different way to present sentences/documents and a different approach to learning (a discriminative model), so it gives a significant support in our ensemble method. On document level, LSTM method produced a much lower performance than DVngram method. As a result, the feature vectors generated from LSTM model does not support as well as DVngram's vectors when combining with CNN feature vectors.

5.1 Freezing vs Unfreezing in the last NN layer of feature engineering phase

In the engineering phase, we freeze (untrain) the last NN layer's parameters to create efficient sentiment vectors. To evaluate the efficiency of this technique, we compared our vector's performance against the sentiment-specific vector from the unfreezing scheme. We passed these vectors to our 3-layer NN model to achieve the results (details in table 3). One interesting observation is that the performance of a feature vector in freezing mode is better than one in unfreezing mode for most of the cases. In addition, we combined a sentiment-specific vector with the semantic-specific vector - DVngram for evaluating the performance. In general, our freezing scheme provided a higher performance than the unfreezing scheme. The experimental results show that our freezing scheme works more efficiently on CNN model than LSTM model, especially in a case of combining a sentiment-specific vector and a semantic-specific vector.

5.2 Evaluation on combining features

In this section, we compared in performance our approach to combining features from variant models against **Merging scheme** which horizontally merges variant models (details in figure 6).

From the result showed in table 4, we found that our approach for feature vectors combination is applied more efficiently to CNN model than LSTM model. In the scheme of combining feature vectors, CNN feature vector provides a robust performance, while LSTM feature vector provides inconsistent results: better when combining with CNN feature vector, worse when combining with DVngram vector (compared with Merg-

Method		MR-S	MR-L	SST
LSTM		80.17	86.23	87.81
CNN (Kim, 2014)		81.31	91.18	86.33
DVngram (2015)		73.51	92.14	74.2
NBSVM (2012)		79.26	91.87	80.39
DV-Ensemble (2015)		-	93.05	-
DAN (2015)		80.3	89.4	86.3
SA-LSTM (2015)		80.7	92.8	-
DSCNN-Pretrain (2016)		82.2	90.7	**88.7**
Proposed methods				
3-layer NN (CNN-f+LSTM-f) (1)		81.59	91.16	88.41
3-layer NN (CNN-f+DVngram) (2)		81.11	92.98	86.66
Without clustering	Ensemble ((1) + NBSVM)	82.18	92.50	88.36
	Ensemble ((2) + NBSVM)	81.1	93.25	87.31
CNN autoencoder	Ensemble ((1) + NBSVM)	82.2	92.55	88.46
	Ensemble ((2) + NBSVM)	81.74	93.29	86.87
BiLSTM autoencoder	Ensemble ((1) + NBSVM)	**82.22**	92.54	88.58
	Ensemble ((2) + NBSVM)	81.8	**93.32**	87.09

Table 2: Accuracy results on the binary sentiment classification task. ***3-layer NN***$(F_1 + F_2)$ denotes using feature vector F_1 and F_2 as input; **CNN-f**, **LSTM-f** denote sentiment-specific feature vectors generated from the proposed CNN, LSTM respectively; ***Ensemble***$(p_1 + p_2)$ denotes applying the proposed Ensemble for the prediction scores of p_1 and p_2.

Feature	MR-S	MR-L	SST
CNNorg	80.61	91.22	86.05
CNN-f	**80.89**	**91.38**	**86.27**
LSTMorg	78.97	**85.5**	86.99
LSTM-f	**79.11**	85.14	**87.64**
CNNorg + LSTMorg	80.95	90.34	87.31
CNN-f + LSTM-f	**81.59**	**91.16**	**88.41**
CNNorg + DVngram	80.6	92.66	85.34
CNN-f + DVngram	**81.11**	**92.98**	**86.66**
LSTMorg + DVngram	79.38	**90.41**	87.2
LSTM-f + DVngram	**79.59**	88.04	**88.14**

Table 3: Accuracy results of 3-layer NN method on different features. **CNNorg**, **LSTMorg** denote sentiment-specific features engineering from the proposed CNN, LSTM without freezing the last NN layer respectively

Method	MR-S	MR-L	SST
3-layer NN (CNN-f + LSTM-f)	**81.59**	**91.16**	**88.41**
CNN-LSTM	81.07	91.07	86.49
3-layer NN (CNN-f + DVngram)	**81.11**	**92.98**	**86.66**
CNN-DVngram	80.79	92.12	85.39
3-layer NN (LSTM-f + DVngram)	79.59	88.04	**88.14**
LSTM-DVngram	**80.61**	**92.08**	86.49

Table 4: Accuracy results of features combining scheme and Merging scheme.

ing scheme). In most of the cases in Merging scheme, a composition model (i.e. CNN-LSTM) try to reproduce the result of its child models (e.g. CNN, LSTM) and does not provide a significant improvement.

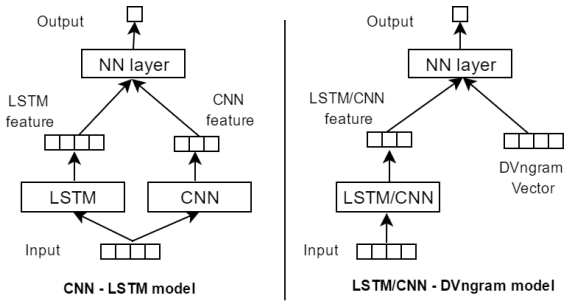


Figure 6: The architecture of merging models.

5.3 Error analysis

To get a better sense of the limitation of the proposed model, we manually inspect some cases of the wrong prediction, which are showed in table 5. These sentences are good examples of the proposed model's weakness.

The first source of false hits is the lack of syntactic information. The model tried to identify sentiment words in a sentence (i.e. not, bad, at all) but it failed to interpret the whole sentence.

The second reason of the wrong prediction comes from missing context information. A word (i.g. foul, freaky) carries a positive or negative sentiment depend on context or domains. We believe that the promising direction in future work will be to improve the model for capturing syntactic and context information.

id	**Sentence**	*L*
1	*Not a bad journey at all.*	1
2	*The best way to hope for any chance of enjoying this film is by lowering your expectations.*	0
3	*You've seen them a million times.*	0
4	*A whole lot foul, freaky and funny.*	1

Table 5: Examples of the wrong prediction. L denotes the true label with 0,1 for negative, positive sentiment labels respectively)

6 Related work

Sentiment analysis is a study of determining people's opinions, emotions toward to entities. Taboada (2011) assigned sentiment labels to text by extracting sentiment words. Liu (2012) formulated the sentiment analysis as a classification task and applied machine learning techniques for this problem. In this approach, dominant research concentrated on designing effective features such as word ngram (Wang and Manning, 2012), emoticon (Zhao et al., 2012), sentiment words (Kiritchenko et al., 2014). However, designing handcraft features requires an intensive effort.

Recently, the emergence of deep learning models has provided an efficient way to learn continuous representation vectors for sentiment classification. Bengio (2003) and Mikolov (2013) introduced learning techniques for semantic word representation. By using a neural network in the context of a word prediction task, the authors generated word embedding vectors carrying semantic meanings. Embedding vectors of words which share similar meanings are close to each other. Semantic information maybe provides opposite opinions in different contexts. Therefore, some research (Socher et al., 2011; Tang et al., 2014) worked on learning sentiment-specific word representation by employing sentiment text. For sentence and document level, composition approach attracted many studies. Yessenalina and Cardie (2011) modeled each word as a matrix and used iterated matrix multiplication to present a phrase. Deep recursive neural networks (DRNN) over tree structures were employed to learn sentence representation for sentiment classification such as DRNN with binary parse trees (Irsoy and Cardie, 2014), Recursive tensor neural network with sentiment treebank (Socher et al., 2013). CNN has recently been applied efficiently for semantic composition (Kim, 2014; Zhang and Wallace, 2015). This technique uses convolutional filters to capture local dependencies in term of context windows and applies a pooling layer to extract global features. Le and Mikolov (2014) applied paragraph information into the word embedding technique to learn semantic representation. Tang et al. (2015) used CNN or LSTM to learn sentence representation and encoded these semantic vectors in document representation by Gated recurrent neural network. Zhang (2016) proposed Dependency Sensitive CNN to build hierarchically textual representations by processing pretrained word embeddings. Wang (2016) used a regional CNN-LSTM to predict the valence arousal ratings of texts.

In our work, we designed a freezing approach for learning efficiently sentiment document representation from two variant deep-learning models: CNN and LSTM. Afterward, these sentiment-specific vectors and the semantic DVngram vector were employed for sentiment classification. This strategy captures the advantages of variant models by using vectors, which each model generated. We also used NBSVM in clustering mode to boost the performance of classification.

7 Conclusion

In this work, we introduced a novel approach to synthesize feature vectors for sentiment analysis from CNN, LSTM. These vectors provide a simple and efficient way to integrate the strong abilities of these models. For sentiment classification with CNN, LSTM vectors, we proposed a 3-layer neural network which efficiently takes advantages of these vectors. In addition, we proposed a strategy to cluster documents/sentences by their similarity. In each cluster, we applied an ensemble method of the 3-layer neural network and NBSVM. It achieves the state of the art results in the datasets: MR-S and MR-L. In the current work, we just focused on individual models. Research on applying combination models for feature engineering maybe provides interesting results.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *journal of machine learning research*, 3(Feb):1137–1155.

Y-Lan Boureau, Jean Ponce, and Yann LeCun. 2010. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 111–118.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, pages 3079–3087.

Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Neural Networks, 1996., IEEE International Conference on*, volume 1, pages 347–352. IEEE.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ozan Irsoy and Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. In *Advances in Neural Information Processing Systems*, pages 2096–2104.

Mohit Iyyer, Varun Manjunatha, Jordan L Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *ACL (1)*, pages 1681–1691.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Svetlana Kiritchenko, Xiaodan Zhu, and Saif M Mohammad. 2014. Sentiment analysis of short informal texts. *Journal of Artificial Intelligence Research*, 50:723–762.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196.

Bofang Li, Tao Liu, Xiaoyong Du, Deyuan Zhang, and Zhe Zhao. 2015. Learning document embeddings by predicting n-grams for sentiment classification of long movie reviews. *arXiv preprint arXiv:1512.08183*.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Pengfei Liu, Shafiq Joty, and Helen Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Grégoire Mesnil, Tomas Mikolov, Marc'Aurelio Ranzato, and Yoshua Bengio. 2014. Ensemble of generative and discriminative techniques for sentiment analysis of movie reviews. *arXiv preprint arXiv:1412.5335*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Andriy Mnih and Geoffrey E Hinton. 2009. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pages 1081–1088.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models

for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational linguistics*, 37(2):267–307.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1432.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *ACL (1)*, pages 1555–1565.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

Jin Wang, Liang-Chih Yu, K Robert Lai, and Xuejie Zhang. 2016. Dimensional sentiment analysis using a regional cnn-lstm model. In *The 54th Annual Meeting of the Association for Computational Linguistics*, volume 225.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics.

Xin Wang, Yuanchao Liu, Chengjie Sun, Baoxun Wang, and Xiaolong Wang. 2015. Predicting polarities of tweets by composing word embeddings with long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 1, pages 1343–1353.

Ainur Yessenalina and Claire Cardie. 2011. Compositional matrix-space models for sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 172–182. Association for Computational Linguistics.

Rui Zhang, Honglak Lee, and Dragomir Radev. 2016. Dependency sensitive convolutional neural networks for modeling sentences and documents. *arXiv preprint arXiv:1611.02361*.

Ye Zhang and Byron Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.

Jichang Zhao, Li Dong, Junjie Wu, and Ke Xu. 2012. Moodlens: an emoticon-based sentiment analysis system for chinese tweets. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1528–1531. ACM.