2016 IEEE 28th International Conference on Tools with Artificial Intelligence

Combining Statistics-based and CNN-based Information for Sentence Classification

Lang Zhining*‡, Gu Xiaozhuo*, Zhou Quan*, Xu Taizhong†
*Institute of Information Engineering, Chinese Academy of Sciences
†National Computer Network Emergency Response Technical Team/Coordination Center
‡University of Chinese Academy of Sciences
Beijing, China
Email: {langzhining, guxiaozhuo, zhouquan}@iie.ac.cn, tzh_xu@sohu.com

***Abstract*—Sentence classification, serving as the foundation of the subsequent text-based processing, continues attracting researchers attentions. Recently, with the great success of deep learning, convolutional neural network (CNN), a kind of common architecture of deep learning, has been widely used to this filed and achieved excellent performance. However, most CNN-based studies focus on using complex architectures to extract more effective category information, requiring more time in training models. With the aim to get better performance with less time cost on classification, this paper proposes two simple and effective methods by fully combining information both extracted from statistics and CNN. The first method is S-SFCNN, which combines statistical features and CNN-based probabilistic features of classification to build feature vectors, and then the vectors are used to train the logistic regression classifiers. And the second method is C-SFCNN, which combines CNN-based features and statistics-based probabilistic features of classification to build feature vectors. In the two methods, the Naive Bayes log-count ratios are selected as the text statistical features and the single-layer and single-channel CNN is used as our CNN architecture. The testing results executed on 7 tasks show that our methods can achieve better performance than many other complex CNN models with less time cost. In addition, we summarized the main factors influencing the performance of our methods though experiment.**

***Keywords*-Sentence Classification; Convolutional Neural Network; Navie Bayes**

I. Introduction

As the foundation of the subsequent text-based processing, Sentence classification plays an important role in natural language procession (NLP). Since sentences usually tend to be short and have colorful expressions for one certain meaning, it's difficult for a machine to classify sentence based on semantic characteristics.

For a long time, some classical machine learning methods, such as Naive Bayes, Support Vector Machine, conditional random fields, are the main streams to processing sentence classification. Wang and Manning proposed two simple methods with uni-bigrams based on Naive Bayes [1]. One is the multinomial Naive Bayes (MNB), the other is the scheme combining Naive Bayes and Support Vector Machine (NB-SVM). Nakagawa proposed a dependency tree-based method using conditional random fields with hidden variables in [2], and Yang *et al.* put forward a context-aware method for classification with posterior regularization in [3].

Recently, with the great success of deep and neural learning in computer vision [4] and speech processing tasks [5], deep learning has been applied to many tasks of NLP. By learning word embedding representations through neural language model [6]–[8], researchers can perform classification tasks over the learned word embeddings [9]. In this manner, each sentence can be treated as a matrix. This mapping method inspires researchers to apply convolutional neural networks (CNN), one of the common architectures of deep learning, to sentence classification. Thus, many CNN-based methods were proposed and they can achieve satisfactory performance as expected [10]–[13]. Kalchbrenner [11] proposed a multi-layer CNN model with dynamic k-Max pooling, positing random low-dimensional word vectors as inputs. Kim [10] defined a CNN model with only one convolution layer, initializing vectors with pre-trained word embeddings. On the basis of Kim's work, Zhang and Wallace [14] summarized the influences of various free parameters in the single-layer model, such as word vectors, filter size, activation function, etc. Yin and Schütze [13] proposed a multichannel variable-size CNN architecture that combines diverse versions of pre-trained word embeddings and extracts features of multi-granular phrases with variable-size convolution filters. And Poria [15] considered combining CNN and Support Vector Machine in order to import nonlinearity into the model.

In all CNN-based methods, CNN-non-static, a single-layer and single-channel sentence model proposed by Kim, is the simplest method and has quite satisfactory performance. In order to achieve better performance, most of the researches focus on how to extract more effective category information by increasing parameters or updating the architecture, such

This research is supported by the National Natural Science Foundation of China (Grant No. 61602475) , the National Key Research and Development Plan of China (Grant No. 2016YFB0800102).

as using various word embeddings, increasing the number of layers, or applying new method of pooling. Although they perform better than CNN-non-static, they spend much more time in training. According to these analyses, we consider combining statistics-based and CNN-based information instead of extracting more CNN-based information by complex architecture, which will not only improve classification performance, but also need less time in training model.

This paper proposes two simple and effective methods which combine statistics-based and CNN-based information for improving sentence classification performance with little time cost. The first method is S-SFCNN, which combines statistical features and CNN-based probabilistic features of classification to build feature vectors, and then the vectors are used to train the logistic regression (LR) classifiers. And the second method is C-SFCNN, which combines CNN-based features and statistics-based probabilistic features of classification to build feature vectores. In two methods, Naive Bayes log-count ratios [1] are selected as text statistical features and CNN-non-static [10], a single-layer and single-channel sentence model, is used as our CNN architecture. The testing results executed on 7 tasks show that our methods can achieve better performance than many other complex CNN models. In addition, we summarized the main factors influencing the performance of our methods though experiment.

In remaining parts, Section 2 presents the background knowledge of CNN architecture. Section 3 gives the details of our models. Section 4 reports the experiment. And our work is concluded in Section 5.

II. Background

Our methods select the CNN-non-static architecture which is a single-layer and single-channel CNN-based sentence model proposed by Kim [10]. This section will describe the main process, which is shown in Fig.1.

In the CNN-non-static, each word in one sentence is replaced with its vector representation, thus the sentence generates a sentence matrix $\mathbf{A} \in \mathbb{R}^{s \times d}$ where s is sentence length with zero padding and d is the dimensionality of the word embeddings. And then $\mathbf{A}$ is convolved with some linear filters to get some convolved feature vectors $\mathbf{v}$. Each filter window is defined as a matrix $\mathbf{W}_i \in \mathbb{R}^{h \times d}$ where h denotes the number of words involved in each operation. And 1-max pooling operation is applied to extract largest element from each convolved feature vector $\mathbf{v}$. All largest elements finally constitute a CNN feature vector $\mathbf{c} \in \mathbb{R}^k$ where k is the number of filters. Finally, $\mathbf{c}$ is fed to fully connected layer with dropout [16] and goes through a softmax function to get a classification probabilistic vector $\mathbf{p} \in \mathbb{R}^n$ where n is the number of labels.

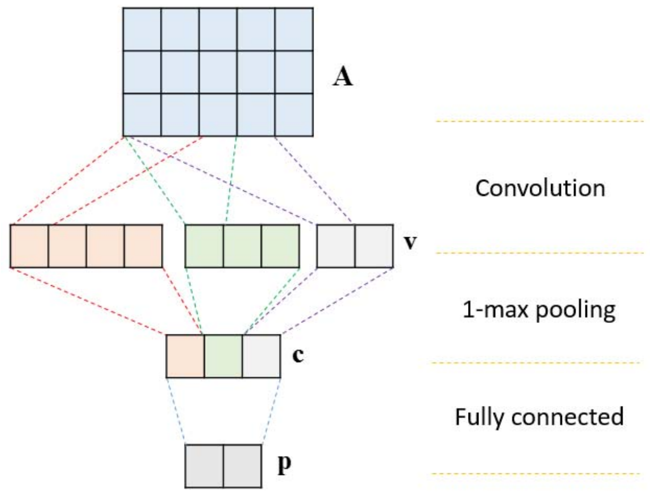


Figure 1: CNN-non-static where length of sentence s is 5, dimension d of word vector is equal to 3, and the sizes of filters are 2×3, 3×3, 4×3, and each size include one filter respectively.

III. Model Architecture

In this section, we will focus on describing how to extract text statistical features and building S-SFCNN and C-SFCNN that are two methods to combine statistics-based and CNN-based information to improve sentence classification performance with little time cost.

A. Extracting statistical features

Wang and Manning proposed a novel method to use NB log-count ratios as feature vectors to train classifiers in [1], and achieved good performance on binary classification tasks. Thus, we use NB log-count ratios as our statistical features, and improve it to adapt to multi-class tasks.

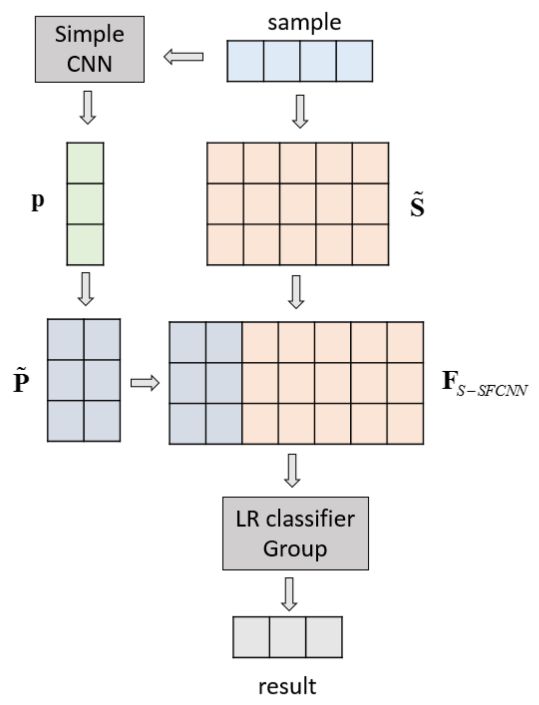
We define a set $\mathbf{V}$ which includes all word unigrams and bigrams of the training corpus. Let $\mathbf{s}^{(i)} \in \mathbb{R}^{|\mathbf{V}|}$ be the word count vector for sample i with label $y^{(i)} \in \{1, 2, ..., n\}$ where n is the number of labels, and $\mathbf{s}_j^{(i)}$ represents the number of occurrence of word $\mathbf{V}_j$ in sentence i. And then we can get a count matrix $\mathbf{M} = [\mathbf{m}_1, \mathbf{m}_2, ..., \mathbf{m}_n]$, and $\mathbf{m}_l$ is described as:

$$\mathbf{m}_l = 1 + \sum_{i:y^{(i)}=l} \mathbf{s}^{(i)} \tag{1}$$

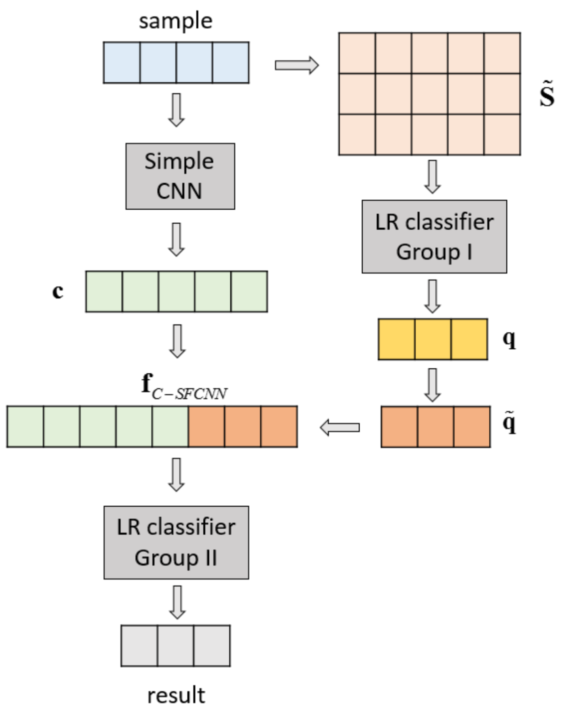
Then, we can get the NB log-count ratio matrix $\mathbf{R} = [\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_n]$ where $\mathbf{r}_l$ is:

$$\mathbf{r}_l = log\left(\frac{\mathbf{m}_l/||\mathbf{m}_l||_1}{\sum_{j \neq l}^{n} \mathbf{m}_j/||\mathbf{m}_j||_1}\right) \tag{2}$$

In order to get a statistical feature vector for sample i, we define an indicator vector $\hat{\mathbf{s}}^{(i)} = \mathbf{1}\left\{\mathbf{s}^{(i)} > 0\right\}$ where $\mathbf{1}$ is an



(a) S-SFCNN



(b) C-SFCNN

Figure 2: Three-class model architecture of S-SFCNN and C-SFCNN for an example sentence.

indicator function. Finally, we can extract statistical feature vector $\tilde{\mathbf{s}}_l^{(i)}$ of i for each label $y^{(i)} = l$, and it is:

$$\tilde{\mathbf{s}}_l^{(i)} = \mathbf{r}_l \circ \hat{\mathbf{s}}^{(i)} \tag{3}$$

where symbol $\circ$ is the element-wise multiplication operation. Merging all $\tilde{\mathbf{s}}_l^{(i)}$, we can get a statistical feature matrix $\tilde{\mathbf{S}}^{(i)} = \left[\tilde{\mathbf{s}}_1^{(i)}, \tilde{\mathbf{s}}_2^{(i)}, ..., \tilde{\mathbf{s}}_n^{(i)}\right]^T$ for sample i. Using the method, we can extract the statistical features for all sentences. Since the operation can be completed before training model, it just costs little time.

B. S-SFCNN

One of our methods is S-SFCNN, which combines s-tatistical features and CNN-based probabilistic features of classification to build feature vectors, and then the vectors are applied to train LR classifier group. Fig.2(a) describes the classification process on three-classification task.

For sample i, Simple CNN generates a classification probabilistic vector $\mathbf{p}^{(i)}$ which is mapped into a directive probabilistic matrix $\tilde{\mathbf{P}}^{(i)}$. In order to explain the mapping process, we define $\mathbf{p}^{(i)} = \left[p_1^{(i)}, p_2^{(i)}, ..., p_n^{(i)}\right]^T$ where $p_l^{(i)}$ is the probability belonging to label l and n is the number of labels, and $\mathbf{p}_{\neg l}^{(i)}$ is the vector $\mathbf{p}^{(i)}$ without $p_l^{(i)}$. Then, we need to obtain a directive probabilistic matrix $\tilde{\mathbf{P}}^{(i)} = \left[\tilde{\mathbf{p}}_1^{(i)}, \tilde{\mathbf{p}}_2^{(i)}, ..., \tilde{\mathbf{p}}_n^{(i)}\right]^T$, the $\tilde{\mathbf{p}}_l$ is described as:

$$\tilde{\mathbf{p}}_l = \phi\left(\frac{p_l^{(i)} - \mathbf{p}_{\neg l}^{(i)}}{p_l^{(i)} - \mathbf{p}_{\neg l}^{(i)}}\right) \tag{4}$$

where the function $\phi(\mathbf{x})$ carries out element-wise operation. For each element x_j in $\mathbf{x}$, $\phi(x_j)$ is defined as:

$$\phi(x_j) = \begin{cases} -kx_j^2 & \text{if } x_j < 0 \\ kx_j^2 & \text{others} \end{cases} \tag{5}$$

where k is a parameter which can be regarded as the weight of directive probabilistic features and can control the impact of directive probabilistic features on classification. Since different datasets have different sensitivities for directive probabilistic features, its value need to be adjusted within limits in order to get best performance for different datasets.

By combining $\tilde{\mathbf{P}}^{(i)}$ with statistical feature matrix $\tilde{\mathbf{S}}^{(i)}$ that generated by NB log-count ratio matrix $\mathbf{R}$ and directive vector $\hat{\mathbf{s}}^{(i)}$ of sample i, we can get the S-SFCNN feature matrix $\mathbf{F}_{S-SFCNN}^{(i)}$ as follows:

$$\mathbf{F}_{S-SFCNN}^{(i)} = \begin{bmatrix} \tilde{\mathbf{s}}_1^{(i)}, \tilde{\mathbf{s}}_2^{(i)}, ..., \tilde{\mathbf{s}}_n^{(i)} \\ \tilde{\mathbf{p}}_1^{(i)}, \tilde{\mathbf{p}}_2^{(i)}, ..., \tilde{\mathbf{p}}_n^{(i)} \end{bmatrix}^T \tag{6}$$

And then for multi-class task, we use one-vs-rest strategy which assigns each row of $\mathbf{F}^{(i)}_{S-SFCNN}$ to one classifier of LR classifier group to gain the probability belonging to corresponding label. For example, the l-th row of $\mathbf{F}^{(i)}_{S-SFCNN}$ is fed to l-th classifier in classifier group to gain the probability of sample i belonging to label l. The LR classifier group is composed of several binary-class logistic regression classifiers, and the number of classifiers is equal to the number of categories. And for binary-class task, classifier group just need one classifier, and we only use one row of $\mathbf{F}^{(i)}_{S-SFCNN}$ to train. There are two reasons why we select LR classifiers. One is that it can greatly reflect the effect of weight k on classification. The other is that the training speed of LR is quite fast. After normalized operation for all results of LR classifiers, we can gain final classification probabilistic vector.

C. C-SFCNN

The other method is S-SFCNN, which combines CNN-based features and statistics-based probabilistic features of classification to build feature vectors, and then the vectors are applied to train LR classifier group. In this method, the statistics-based class probabilities is generated by training another LR classifier group. We show the classification process in Fig.1(b) which is a three-class task.

For sample i, We use statistical feature matrix $\tilde{\mathbf{S}}^{(i)}$ as the input of LR classifier group I to get the classification probabilistic vector $\mathbf{q}^{(i)}$. The process is the same as S-SFCNN. By processing $\mathbf{q}^{(i)}$, the directive probabilistic vector $\tilde{\mathbf{q}}^{(i)}$ can be created:

$$\tilde{\mathbf{q}}^(i) = \phi\left(\mathbf{q}^{(i)} - \frac{1}{n}\right) \tag{7}$$

where the function $\phi(\mathbf{x})$ is the same as that in S-SFCNN and n is the number of labels. In function $\phi(x)$, we also use parameter k to control the impact of directive probabilistic features. Since the process of operating features is different from S-SFCNN, parameter k has different value range. Concatenating $\tilde{\mathbf{q}}^{(i)}$ and the Simple CNN feature vector $\mathbf{c}^{(i)}$, we can build C-SFCNN feature vector:

$$\mathbf{f}^{(i)}_{C-SFCNN} = \begin{bmatrix} \mathbf{c}^{(i)} \\ \tilde{\mathbf{q}}^{(i)} \end{bmatrix}^T \tag{8}$$

And $\mathbf{f}^{(i)}_{C-SFCNN}$ is fed to LR classifier II with one-vs-rest strategy to get final result for sample i.

IV. Experiments

A. Datasets

We select 7 datasets to test the performance of our methods:

- **MR**: movie reviews dataset. There is one sentence per review, and the task is to classify positive/negative reviews [17].[1]

[1]https://www.cs.cornell.edu/people/pabo/movie-review-data/

Table I: Summary Statistics of Datasets

Data	c	l	N	$\lvert V\rvert$	$Test$
MR	2	20	10662	18765	CV
SST-1	5	18	11855	17836	2210
SST-2	2	19	9613	16185	1821
CR	2	19	3775	5340	CV
SUBJ	2	23	10000	21323	CV
TREC	6	10	5952	9592	500
MPQA	2	3	10606	6246	CV

- **SST-1**: Stanford Sentiment Treebank with fined-grained labels (very positive, positive, neutral, negative, very negative). It is an extension of MR and has train/dev/test splits [18].[2]
- **SST-2**: Same as SST-1 but with neutral reviews removed and binary labels.
- **CR**: customer reviews of various products dataset. Classification involves detecting positive/negative reviews [19].[3]
- **SUBJ**: subjectivity database with subjective reviews and objective plot summaries [20].
- **TREC**: question dataset involving six question type [21].[4]
- **MPQA**: Opinion polarity prediction of the MPQA dataset [22].[5]

Summary statistics of the datasets are in Table 1 where c is number of labels, l is average sentence length, N represents dataset size. $|V|$ is vocabulary size, and $Test$ describes the size of test set (CV means there is no train/test split and need cross validation).

B. Training

1) Settings of CNN: we select CNN-no-static, a singel-layer and single-channel sentence model proposed by Kim (2014), as our Simple CNN. And the hyperparameters settings are same as him. We use filter windows of 3, 4, 5 with 100 features maps each, dropout rate of 0.5, l_2 constraint of 3, and employ word2vec vectors trained on 100 billion words from Google News (Mikolov et al., 2013)[6] to initialize the word embeddings, and words not in the words set are initialized randomly. AdaDelta [23] is used as stochastic gradient descent (SGD) update rule with mini-batch size of 50.

2) Classifiers: we apply logistic regression classifier implemented by scikit-learn[7] which called the interfaces of liblinear[8], and set the inverse of regularization strength C

[2]http://nlp.stanford.edu/sentiment/
[3]http://www.cs.uic.edu/ liub/FBS/sentiment-analysis.html
[4]http://cogcomp.cs.illinois.edu/Data/QA/QC/
[5]http://www.cs.pitt.edu/mpqa/
[6]https://code.google.com/p/word2vec/
[7]http://scikit-learn.org/stable/
[8]http://www.csie.ntu.edu.tw/ cjlin/liblinear/

to be 1 for all dataset except SST-1. For SST-1 dataset, the default value of C can cause overfitting, so we set it to be 0.01 for classifiers in S-SFCNN's classifier group and C-SFCNN's group I, and set it to be 0.001 in C-SFCNN's classifier group II.

3) Parameter k: it represents the importance of directive probabilistic features $\tilde{\mathbf{p}}$ or $\tilde{\mathbf{q}}$, and depends on the characteristics of datasets. By adjusting it, we can find the appropriate value which can take full advantage of probabilistic information and reduce interference for other features. Since $\tilde{\mathbf{p}}$ and $\tilde{\mathbf{q}}$ have different operations, it has different value ranges in S-SFCNN and C-SFCNN. And the best values of k for the 7 datasets are shown in table 2.

Table II: Parameter k for Datasets

Data	S-SFCNN	C-SFCNN
MR	0.1	1.5
SST-1	0.5	1
SST-2	0.1	1.5
CR	0.5	1
SUBJ	0.25	2
TREC	0.5	1
MPQA	0.25	1

4) Others: since CNN don't reach the best result when our models achieve best result in most cases, we need to test our models using dev set at each iteration of CNN. It can't spend much more time than single-layer CNN because the time of training LR classifier is much less than CNN and our models need less iterations. For datasets without a standard dev set we use 10-fold cross validation, which randomly select 10% of the training data as the dev set.

C. Experiment Results and Discussion

The experiment results of our methods are listed in table 3. We select two baseline models which can be regarded as the submodules of our methods. **Simple CNN**: a CNN-based single-layer and single-channel sentence model that is the same as CNN-non-static. **SF**: the method only using NB log-count ratios as the statistical feature vectors to train LR classifiers. We also compare our methods with other models which include several CNN-based methods and other effective methods. **CNN-rand/static/non-static/multichannel**: CNN with word embeddings randomly initialized / initialized by word2vec vectors and keep static during training / initialized by word2vec vectors that can be fine-turned for each task / initialized by two sets of word vectors [10]. **DCNN**: Dynamic Convolutional Neural Network with k-max pooling [11]. **MVCNN**: Multichannel Variable-size CNN architecture [13]. **Paragraph-Vec**: Logistic regression on top of paragraph vectors [24]. **NBSVM/MNB**: Navie Bayes SVM and Multinormal Navie Bayes with uni-bigrams [1]. **G-Dropout/F-Dropout**: Gaussion Dropout and Fast Dropout [25]. **SVMs**: SVM with uni-bi-trigrams, wh word, head word, POS, parser, hypernyms, and 60 hand-coded rules as features [26]. **Tree-CRF**: Dependency tree with Conditional Random Fields [2]. **CRF-PR**: Conditional Random Fields with Posterior Regularization [3].

In the rest of the part, we will discuss the performance of our methods from two aspects. Firstly, we will compare S-SFCNN/C-SFCNN with baseline models (SF and Simple CNN), and give the conclusion about our two methods' characteristics. Then, we will make a comparison between our methods and other models to show the advantages of our methods.

1) S/C-SFCNN vs SF/Simple CNN: we compared our methods with baselines SF and Simple CNN. S-SFCNN performs better than SF and Simple CNN on MR, SST-2, CR, SUBJ, MPQA which are binary-class tasks. But we also note it is weak on SST-1 and TREC which are multi-class problems. On SST-1 and TREC, the performance of S-SFCNN are only slightly better than SF but worse than Simple CNN. C-SFCNN has better performance than SF and Simple CNN on 6 of 7 datasets. Although it performs slightly worse than S-SFCNN on binary-class datasets except SUBJ, it has much better performance than S-SFCNN on multi-class tasks which are SST-1 and TREC. The results show that our methods can take full advantage of information both extracted from text statistics and C-NN to get better performance. Besides, the two methods have different characteristics. S-SFCNN is more suitable for binary classification tasks but is weaken on multi-class tasks, whereas C-SFCNN has more stable performance and behaves well on multi-class tasks.

2) Our methods vs Other methods: in comparison with other methods, our methods also have obvious advantage and achieve best results on five tasks. We pay attention to MVCNN, a quite effective CNN architecture in sentence classification, that outperforms other models on SUBJ, SST-1 and SST-2 datasets. Although it has good performance, it is worth notice that MVCNN has a much more complex process that involves pre-training and mutual-learning steps, usually causing hours to train. Comparing with MVCNN, our methods just use on the order of minutes, which is much less time, to get similar performance on SUBJ and SST-1. The reduction of our methods' training time are contributed mainly by two reasons. One is that we make each part of our methods as simple as possible. For the Simple CNN in our methods, we use simplest architecture which applies only one set of word embeddings as input and has only one convolution layer. For selection of classifiers, LR not only conveniently obtain classification probabilities but also spend little training time. Besides, the statistical features can be obtained before training models. The other is that our methods need fewer iterations to get best model's performance by mutual guidance between statistics-based and CNN-based information. All of these can reduce the

Table III: Results of our CNN models against other methods (%)

Model	MR	SST-1	SST-2	CR	SUBJ	TREC	MPQA
SF	78.7	40.8	82.0	81.6	91.4	90.6	84.5
Simple CNN	81.2	48.1	86.9	84.3	93.2	93.6	89.8
S-SFCNN	**82.7**	41.4	88.3	**85.8**	93.8	91.6	**89.9**
C-SFCNN	82.4	**49.7**	87.9	85.2	**93.9**	93.8	89.6
CNN-rand	76.1	45.0	82.7	79.8	89.6	91.2	83.4
CNN-static	81.0	45.5	86.8	84.7	93.0	92.8	89.6
CNN-non-static	81.5	48.0	87.2	84.3	93.4	93.6	89.5
CNN-multichannel	81.1	47.4	88.1	85.0	93.2	92.2	89.4
DCNN	-	48.5	86.8	-	-	93.0	-
MVCNN	-	49.6	**89.4**	-	**93.9**	-	-
Paragraph-Vec	-	48.7	87.8	-	-	-	-
NBSVM	79.4	-	-	81.8	93.2	-	86.3
MNB	79.0	-	-	80.0	93.6	-	86.3
G-Dropout	79.0	-	-	82.1	93.4	-	86.1
F-Dropout	79.1	-	-	81.9	93.6	-	86.3
SVMS	-	-	-	-	-	**95.0**	-
Tree-CRF	77.3	-	-	81.4	-	-	86.1
CRF-PR	-	-	-	82.7	-	-	-

training time to the greatest extent.

D. Model Performance Analysis

Analyzing the results, we note our methods have different performance on different datasets. For example, they have obviously positive effect on MR and CR, but have little improvement on TREC and MPQA. In order to find out the main reasons, we calculate the ratio of sentences where SF gives correct labels while Simple CNN gives wrong labels (r_1) and SF gives wrong labels while Simple CNN gives correct labels (r_2) on each test set, and also list the difference between best accuracy of our models and the accuracy of Simple CNN (d). The size of d can represent the influence of our methods on different datasets to some extent, which is shown in table 4.

Table IV: Parameter k for datasets (%)

Data	r_1	r_2	$r_2 - r_1$	d
MR	7.7	10.2	2.5	1.5
SST-1	12.7	20.7	8.0	1.6
SST-2	5.4	10.1	4.7	1.4
CR	6.7	9.5	2.8	1.5
SUBJ	2.8	4.5	1.7	0.7
TREC	2.2	5.4	3.2	0.2
MPQA	3.3	8.8	5.5	0.1

From this table, we try to seek the relationship between d and other items. Since Simple CNN has better performance than SF, r_1 is lower than r_2 on all datasets, which is the foundation of all the following analysis. For MR, SST-1, SST-2 and CR, r_1 is relatively large and d exceeds 1.4%. For SUBJ, TREC and MPQA, r_1 is relatively small and d is below 1%. This phenomenon indicates that there is an approximate positive correlation between r_1 and d. If d is only affected by r_1, d should have similar values in SUBJ, TREC and MPQA since r_1 has similar values in the three datasets. However, d in TREC and MPQA is much smaller than SUBJ, especiall in MPQA where r_1 is highest among them but d is approximately equal to 0. This indicates there exists some other factors that influences d. Observing the difference between r_2 and r_1, MPQA and TREC have much higher values of $r_2 - r_1$ than SUBJ, which implies d and $r_2 - r_1$ have negative correlation at least when r_1 is relatively small.

According to the above discussion, we can get an important conclusion that the difference of classification between our baseline models (SF and Simple CNN) has much effect on the performance of our methods. On the condition that Simple CNN has better performance than SF, our methods will behave better when there is a higher proportion of sentences that SF gives correct labels while Simple CNN gives wrong labels and a lower ratio of sentences that SF gives wrong labels while Simple CNN gives correct labels. It is coincident with our intuitive understanding that the more useful information we can extract from two baseline models, the better our methods perform.

V. Conclusion

In this paper, we proposed two CNN-based methods, known as S-SFCNN and C-SFCNN, for sentence classification. They combine statistical features and CNN-based probabilistic features of classification, CNN-based features and statistics-based probabilistic features of classification to make decisions respectively. Through experiments, it can be seen that our methods are able to achieve satisfactory classification performance with relatively small amount of training time. Besides, we can draw a conclusion that S-SFCNN is good at binary-class tasks and C-SFCNN has obvious advantage for multi-class tasks. In addition, we found the difference of classification between our baseline models has much effect on the performance of our methods.

Acknowledgment

We would like to thank Kim and the anonymous reviewers for their helpful feedback and suggestions.

References

[1] S. Wang and C. Manning, "Baselines and bigrams: Simple, good sentiment and topic classification," in *ACL*, Jeju Island, Korea, July 2012, pp. 90–94.

[2] T. Nakagawa, K. Inui, and S. Kurohashi, "Dependency tree-based sentiment classification using crfs with hidden variables," in *ACL*, Los Angeles, California, June 2010, pp. 786–794.

[3] B. Yang and C. Cardie, "Context-aware learning for sentence-level sentiment analysis with posterior regularization," in *ACL*, Baltimore, Maryland, June 2014, pp. 325–335.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, December 2012, pp. 1097–1105.

[5] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *ICASSP*, Vancouver, Canada, May 2013, pp. 6645–6649.

[6] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *JMLR*, vol. 3, no. February, pp. 1137–1155, 2003.

[7] W.-t. Yih, K. Toutanova, J. C. Platt, and C. Meek, "Learning discriminative projections for text similarity measures," in *CoNLL*, Portland, Oregon, USA, June 2011, pp. 247–256.

[8] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, South Lake Tahoe, Nevada, USA, December 2013, pp. 3111–3119.

[9] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *JMLR*, vol. 12, pp. 2493–2537, August 2011.

[10] Y. Kim, "Convolutional neural networks for sentence classification," in *EMNLP*, Doha, Qatar, October 2014, pp. 1746–1751.

[11] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," in *ACL*, Baltimore, Maryland, June 2014, pp. 655–665.

[12] Y. Goldberg, "A primer on neural network models for natural language processing," *CoRR*, vol. abs/1510.00726, November 2015.

[13] W. Yin and H. Schütze, "Multichannel variable-size convolution for sentence classification," in *CoNLL*, Beijing, China, July 2015, pp. 204–214.

[14] Y. Zhang and B. Wallace, "A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification," *CoRR*, vol. abs/1510.03820, June 2015.

[15] S. Poria, E. Cambria, and A. Gelbukh, "Deep convolutional neural network textual features and multiple kernel learning for utterance-level multimodal sentiment analysis," in *EMNLP*, Lisbon, Portugal, September 2015, pp. 2539–2544.

[16] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, vol. abs/1207.0580, October 2012.

[17] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," in *ACL*, Ann Arbor, Michigan, June 2005, pp. 115–124.

[18] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *EMNLP*, Seattle, Washington, USA, October 2013, pp. 1631–1642.

[19] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *SIGKDD*, Seattle, Washington, USA, August 2004, pp. 168–177.

[20] W. Pang and L. Lee, "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts," in *ACL*, Barcelona, Spain, July 2004, pp. 271–278.

[21] X. Li and D. Roth, "Learning question classifiers," in *COLING*, Taipei, Taiwan, August 2002, pp. 1–7.

[22] J. Wiebe, T. Wilson, and C. Cardie, "Annotating expressions of opinions and emotions in language," *Language Resources and Evaluation*, vol. 39, pp. 165–210, May 2005.

[23] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *CoRR*, vol. abs/1212.5701, January 2012.

[24] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents." in *ICML*, Beijing, China, June 2014, pp. 1188–1196.

[25] S. I. Wang and C. D. Manning, "Fast dropout training." in *ICML*, Atlanta, GA, United states, June 2013, pp. 118–126.

[26] J. Silva, L. Coheur, A. C. Mendes, and A. Wichert, "From symbolic to sub-symbolic information in question classification," *Artificial Intelligence Review*, vol. 35, pp. 137–154, February 2011.