Received June 13, 2020, accepted July 3, 2020, date of publication July 14, 2020, date of current version July 23, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3009244

Deep Learning Based Fusion Approach for Hate Speech Detection

YANLING ZHOU[1,2], YANYAN YANG[2], HAN LIU[3], (Member, IEEE), XIUFENG LIU[4], AND NICK SAVAGE[2], (Member, IEEE)
[1]School of Computer Science and Information Engineering, Hubei University, Hubei 430062, China
[2]School of Computing, University of Portsmouth, Portsmouth PO1 3HE, U.K.
[3]College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China
[4]Department of Management Engineering, Technical University of Denmark, 2800 Kongens Lyngby, Denmark

Corresponding author: Yanyan Yang (linda.yang@port.ac.uk)

This work was supported by the China Scholarship Council under Grant 201808420357.

ABSTRACT In recent years, the increasing prevalence of hate speech in social media has been considered as a serious problem worldwide. Many governments and organizations have made significant investment in hate speech detection techniques, which have also attracted the attention of the scientific community. Although plenty of literature focusing on this issue is available, it remains difficult to assess the performances of each proposed method, as each has its own advantages and disadvantages. A general way to improve the overall results of classification by fusing the various classifiers results is a meaningful attempt. We first focus on several famous machine learning methods for text classification such as Embeddings from Language Models (ELMo), Bidirectional Encoder Representation from Transformers (BERT) and Convolutional Neural Network (CNN), and apply these methods to the data sets of the SemEval 2019 Task 5. We then adopt some fusion strategies to combine the classifiers to improve the overall classification performance. The results show that the accuracy and F1-score of the classification are significantly improved.

INDEX TERMS Hate speech, machine learning, Bert, CNN, classifiers fusion.

I. INTRODUCTION

The popularity of social media platforms such as Facebook, Twitter and YouTube, etc. provide channels for internet users to express their opinions and share comments that are visible to all. Some people express aggressive, hateful or threatening speech online arbitrarily. Hate speech is commonly defined as any public speech that expresses disparagement to a person or a group on the basis of some characteristics such as race, color, ethnicity, gender, sexual orientation, nationality, religion, or other characteristics [1], [2]. Social networks encourage the interactions between people to be more indirect and anonymous thus providing anonymity for some people making them feel safer even though they express hate speech. It can easily lead to disruptive anti-social outcomes if it continues to be unregulated and uncontrolled. Hate speech is therefore considered as a serious problem worldwide, and many countries and organizations resolutely resist it [3]. The polarity detection of speech on platforms is the first step and is critical to government departments, social security services, law enforcement and social media companies which expect to remove accounts with offensive content from their websites [4].

Compared with manual filtering which is very time consuming, automatic identification of hate speech will enable the platform to detect the hate speech and remove them much more quickly and efficiently. The problem of online hate speech detection has raised interest in both the scientific community and the business world. There have been many research efforts aimed at automating the process which is usually modeled as a supervised classification problem. Recently, machine learning approach which can learn the different associations between pieces of text, and that a particular output is expected for a particular input by using pre-labeled examples as training data is popular in scientific studies for hate speech detection. Among various machine learning methods, deep learning which is a subset of machine learning, is very prominent in Natural Language Processing (NLP) to tackle the issue of text classification [5], [6].

Although great contributions have been made in this area of work, each research method has its own advantages and disadvantages. It is still difficult to compare their performance, largely due to the use of different datasets and different feature extraction techniques. Different methods usually

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

provide disparate suitability on feature sets, even for the same datasets. The most important question, perhaps, is not which method is the best, but how the results can be better used in general. It is therefore more worthwhile to find a way to improve the results of each classification. Ensemble learning tries to improve the overall performance of the system efficiently by combining the outputs from various candidate systems [7]. The underlying idea of ensemble learning is that even if one weak classifier gets the wrong prediction, other classifiers can correct the error back to some extent. The two most common ways of ensemble learning are bagging and boosting [8]. However, these two methods are unsuitable for ensemble learning between different classifiers. Adopting some simple algebraic rules of fusion for multiple classifiers results may prove meaningful.

In this paper, we focus on several famous deep learning methods. We then apply fusion methods to combine the classifiers to improve the overall classification performance.

The remainder of this paper is structured as follows: in Section II we introduce the terminology and some state–of-the-art methods. In Section III we propose the framework of classifier fusion and introduce the factors affecting fusion results. In Section IV we conduct experiments and discuss the experimental results. Section V we conclude the paper and propose possible directions for future work.

II. RELATED WORK

In this section we introduce the terminology of hate speech and the principles of state-of-the-art deep learning methods.

A. TERMINOLOGY

Hate speech leads to discrimination against particular categories of people and undermines equality. Immigrants and women are usually the main targets. Over the past few decades, hate against immigrants has grown rapidly as a result of the refugee crisis and political changes. Some governments and policy makers are currently trying to address this issue, and making a special effort for the identification and monitoring of hate speech against immigrants. Hate against the female gender is a well-known form of discrimination, ongoing for a long time and generally manifested in the form of abuse, belittling and discrimination against women in the work, social and family environments.

Hate speech is subject to a form of conceptual decomposition and involves some concepts: (1) it targets groups or classes of people by targeting particular characteristics; (2) it demonstrates emotions, feelings, or attitudes of hate or hatred [9]. Hate speech detection belongs to the category of sentiment and emotion analysis, which can be expressed explicitly or implicitly [10]. It is generally expressed as negative opinion, abusive messages, stereotypes, humor, irony and sarcasm. For example, "*go back to your own country*" and "*Women's views don't count*" these words are classified as hate speech. However, in some cases some words may be negative but they may not hate speech in the whole context. For example, "*I hate them wasting time*", even though there is the word "*hate*" employed here, the given sentence is not hate speech.

B. STATE OF THE ART IN DEEP LEARNING

Many approaches have been proposed to detect hate speech. These methods can be divided into two categories: traditional machine learning methods and deep learning methods. The former mainly depends on manual feature extractions, which are then consumed by classification algorithms such as Naïve Bayes, Logistic Regression [11] and Support Vector Machine [12]. The latter employs multi-layers neural networks to abstract useful features from the input raw data automatically. Recently, more and more researchers have addressed hate speech detection problem by using deep learning based models [5].

The deep learning methods can be roughly divided into two categories: one focuses on front-end processing which optimizes the word embedding technology, and the other on mid-end processing which usually uses simple word or character based embedding technology and pays more attention to the middle neural networks processing. The most famous methods focused on front-end processing are Embeddings from Language Models (ELMo) [6], [13], which trains word vectors with context, and Bidirectional Encoder Representation from Transformers (BERT) [14], [15]. BERT is the first deeply bidirectional, unsupervised language representation from unlabeled text by jointly conditioning on both left and right context in all layers. It shows overwhelmingly good performance and has attracted great attention. The most popular network architectures focused on neural networks processing are typically based on long short-term memory networks, such as Convolutional Neural Network (CNN) [16], [17], Recurrent Neutral Network (RNN) and some processing versions of them [18]. As mentioned above, firstly, we focus on several well-behaved deep learning methods in this paper.

1) ELMo

Text representation is the first pivotal step in NLP because the digitization of text features is fundamental to enabling automated processing. At the beginning, discrete representations like one-hot coding method and bag-of-words model are used. They are simple and easy to implement. However, the representation is sparse with high dimensions and does not consider the semantic information of words in the sentence. Then word embeddings which are obtained by training a language model on a large-scale corpus are widely used. One of the most well-known representative works is Word2Vec [19]. Word2Vec showed that we can use a vector to properly represent words in a way that captures semantic or meaning-related relationships as well as syntactic, or grammar-based, relationships.

The distributed representation in the low-dimensional space in Word2Vec not only mitigates the dimensional problem but also implicates the association between words. So Word2Vec has the potential to improve the semantic accuracy of vectors. However, it has one shortcoming that it is unable to express the polysemy in different vectors. It means that, for the same word, even if it has different meanings in the context, its vector is unchanged. It is unacceptable for the higher accuracy requirements for many NLP missions. Therefore, training word vectors with context is proposed, such as ELMo [13]. Instead of using a fixed embedding for each word, ELMo introduces context as new features to dynamically adjust embeddings of words.

ELMo is a deep contextualized word representation that models both complex characteristics of word use and how these uses vary across linguistic contexts. The word vectors that represent contextual features of the input text are learned functions of the internal states of a deep bidirectional language model which is pre-trained on a large text corpus and then can be used as a component in other specific task models. ELMo captures semantic or meaning-related relationships as well as syntactic or grammar-based, relationships. It can achieve good results in solving the problem of polysemous words and outperform previously existing word embeddings like Word2Vec.

ELMo provided a significant step towards pre-training in NLP. It includes two training processes: firstly, generating context-independent word embeddings based on character-level embeddings, and then using a bi-directional language model to generate context-dependent word embeddings.

There are three ways to use the pre-trained ELMo model. We can adopt the version based on pytorch and released by ELMo official AllenNLP, the tensorflow-based version released by ELMo officially and the version implemented by Google based on tensorflow in tensorflow-hub.

2) BERT

In fact, from the perspective of computer vision, regardless of Word2Vec or ELMo, it repeatedly shows the value of transfer learning through a pre-training model. Therefore, NLP researchers have proposed some methods which aim at pre-training a neural network using a language modeling objective and then fine-tunes it onto a target task with supervision [20], [21]. It is a breakthrough in the NLP area, since it makes NLP applications much easier, especially for the users who do not have sufficient data or equipment, since it can save a lot of time and computing resources.

The most popular pre-training general language model is BERT. BERT is the first deeply bidirectional, unsupervised language representation from unlabeled text by jointly conditioning on both left and right context in all layers. There are two steps in BERT framework: pre-training and fine-tuning. During pre-training, an existing unlabeled corpus is used to train a language model. At this stage, Google has invested in large-scale corpora and expensive machines to complete the pre-training process. It is pre-trained with the Books Corpus (800M words) and Wikipedia (2,500M words) on 4 Cloud TPUs in Pod configuration (16 TPU chips total) for 4 days. The second stage: fine-tuning, using pre-trained language models to complete specific NLP downstream tasks. For the fine-tuning, the BERT model is first initialized with the pre-trained parameters, and all of the parameters are fine-tuned using labeled data from the downstream tasks. Each downstream task has a separate fine-tuned model, even though they are initialized with the same pre-trained parameters. With only an additional output layer, the pre-trained BERT model can be fine-tuned to create the state-of-the-art model for the various tasks without too many task-specific architectural changes.

BERT is a model that broke several records for how well it can handle language-based tasks. BERT has two parameter intensive settings [14]: $\text{BERT}_{\text{base}}$ and $\text{BERT}_{\text{large}}$. $\text{BERT}_{\text{base}}$ contains an encoder with 12 Transformer blocks, 12 self-attention heads, and 110 million parameters whereas $\text{BERT}_{\text{large}}$ has 24 layers, 16 attention heads, and 340 million parameters. BERT takes a sequence of tokens with a maximum length of 512 and produces a representation of the sequence in a 768-dimensional vector. The $\text{BERT}_{\text{large}}$ model requires significantly more memory than the $\text{BERT}_{\text{base}}$. As a result, the max batch size for $\text{BERT}_{\text{large}}$ is so small on a normal GPU with 12GB of RAM that it actually hurts the model accuracy, regardless of the learning rate. Therefore, we select the $\text{BERT}_{\text{base}}$ as the base model for further processing.

3) CNN

In recent years, deep neural networks have been widely used and have led to several breakthroughs in some NLP tasks. Convolutional Neural Network (CNN) is a class of deep, feed-forward artificial neural networks and uses a variation of multi-layer perceptron designed to require minimal preprocessing. The CNN can be considered as a feature-extraction architecture which is basically just several layers of convolutions with nonlinear activation functions, but is still the pivotal building block of a larger network. It needs to be trained together with a classification layer in order to produce some useful results.

During the early days, CNN was most commonly applied to image classification. In 2014 Kim proposed the CNN model for sentence classification [16]. Through verification experiments and consensus of the industry, it is generally believed that the CNN model is an ideal model with both efficiency and quality in text classification tasks [22]. The whole model consists of four parts: input layer, convolutional layer, pooling layer and full connection layer. The input layer is a sentence comprised of concatenated word embeddings, followed by a convolutional layer with multiple filters. During the training phase CNN automatically learns the values of its filters based on the specific task that a user wants to perform. Each filter performs convolution on the sentence matrix and generates variable-length feature maps. Then the pooling layer is performed over each map and the largest or mean number from each feature map is recorded. Thus, a univariate feature vector is generated from all maps, and these features are concatenated to form a feature vector for the penultimate layer. Then the final softmax layer classifies the text according to the received feature vector from the last layer.

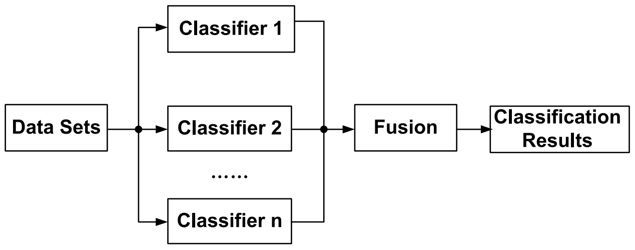

FIGURE 1. Proposed framework of classifier fusion.

In a word, ELMo, BERT and CNN have their own characteristics as mentioned above. ELMo and BERT focus on word embedding, while CNN focuses on neural networks processing. All of these methods demonstrate great significance in the history of NLP and have their own advantages. It is no easy task to decide which method is the best. The specific performance also depends on the specific data set.

III. METHODOLOGY

In this section we propose the framework and combination rules of classifier fusion, and analyze the factors affecting the fusion results.

A. CLASSIFIER FUSION

Generally speaking each classifier method, even the same methods with different parameters, focuses on different angles and emphasizes certain aspects, and each learning algorithm having its own strengths and weaknesses. Therefore, it is difficult to judge which learning algorithm results in a better performance on all feature sets. The idea of ensemble learning in machine learning can advance the overall classification performance and improve the overall accuracy in prediction. Bagging and boosting are classical ensemble learning methods. However, bagging or boosting is based on the same classification algorithm, focusing on the diversity of the data samples, and is short of diversity creation through different algorithms [8]. In addition, feature extraction based on various classifiers is more significant in NLP [23]. Therefore we choose the fixed rules for multiple different classifiers fusion here. This is considering that the back-end fusion of the classification results of different methods is beneficial to improve the overall results, and is also of practical significance. The proposed framework is illustrated in FIGURE 1.

The same data sets may be suitable for different algorithms to learn effectively. In the above framework, Classifier 1, 2, ..., n can be a completely different classification methods or the same method with different parameters. Theoretically, the proposed fusion framework can be applied to all classifiers based on traditional machine learning and deep learning.

There are some fusion methods at the measurement level. One direct rule of combination is referred to as vote, which simply counts the votes for each class and outputs the class that obtains the most votes. Some common algebraic rules of combination include mean, max and product.

Given a n-class classification problem:

$y \in \{c_1, c_2, \ldots, c_n\}$, m classifier $\{h_1, h_2, \ldots, h_m\}$ are trained in a feature space $D : \{x_1, x_2, \ldots, x_k\}$, the combination rules are defined as follows.

$$P_{\text{mean}}(c_i|x_1, x_2, \ldots, x_k) = \frac{1}{m}\sum_{j=1}^{m} P_{h_j}(c_i|x_1, x_2, \ldots, x_k) \quad (1)$$

$$P_{\max}(c_i|x_1, x_2, \ldots, x_k) = \max_{j=1}^{m} P_{h_j}(c_i|x_1, x_2, \ldots, x_k) \quad (2)$$

$$P_{\text{product}}(c_i|x_1, x_2, \ldots, x_k) = \Pi_{j=1}^{m} P_{h_j}(c_i|x_1, x_2, \ldots, x_k) \quad (3)$$

where P_{h_j} is the posterior probability of the classifier h_j, P_{mean} is the posterior probability after averaging fusion for m classifiers, $P_{\max}$ is the posterior probability after maximizing for m classifiers, $P_{product}$ is the posterior probability after production for m classifiers.

Firstly, we can obtain the posterior probabilities P_{h_j} of the classifier h_j from the classification results which show in the form of probabilities labeled 0 and 1 for binary classification. Once more than two groups of classification results are achieved and we choose the appropriate fusion strategies, for example, vote, meaning, maximizing, and production. Then according to the rules of the fusion strategies, which are shown in the formula as above, we can obtain the prediction posterior probability for each text after fusion. The final prediction label value is 0 or 1 depending on whether the corresponding posterior probability is greater or less than 0.5.

By using the simple algebraic rules for classifier fusion, the estimation error of posterior probability and the risk of over fitting can be reduced. Moreover, these algebraic fusion rules are of low computational complexity.

B. FACTORS AFFECTING THE FUSION RESULTS

The final effect of fusion mainly depends on the performance of each classifier and the diversity of classification results.

Firstly, each classifier needs to have good performance. It means that if the performance of one method is much worse than the others, it will pull down the fusion performance. The performance measure can be directly derived from performance analysis indicators such as accuracy and F1-score. Therefore, in this paper we will choose several state-of–the-art and comparable deep learning methods to implement the classifiers.

Secondly, the results of each classifier should be highly diverse. This means that different classifiers should result in different sets of incorrectly classified instances. The diversity can be measured by the Q-statistics [24] which is defined as follows:

$$Q_{ij} = \frac{ad - bc}{ad + bc} \quad (4)$$

where a represents the number of instances that both classifiers i and j give the positive prediction, d represents the number of instances that both classifiers i and j give the negative prediction, b represents the number of instances that classifier i gives the negative class while classifier j gives the positive class; and c represents the number of instances in the opposite case. In other words, a, d are the numbers of the same predicted results in the two classifiers, b, c are the numbers of different predicted results in the two classifiers. In general, with two classifiers that provide good performance the majority of predicted results are the same, the Q_{ij} is a positive number. bc reflects the different prediction results of the two classifiers and indicates the diversity. It can be seen that the smaller Q_{ij}, the greater the diversity.

IV. EXPERIMENTS AND RESULTS ANALYSIS

A. DATASET DESCRIPTION

Model verification is carried out by employing the dataset of SemEval 2019 Task 5 [25] about the detection of hate speech against immigrants and women in English messages extracted from Twitter. From the perspective of simplicity without losing principle we choose a two-class classification Subtask A which demands to identify whether a tweet in English with a particular target (women or immigrants) is hateful or not hateful. The format of an annotated tweet in the training set has the following pattern:

ID Tweet-text HS: Where *ID* is a progressive number denoting the tweet within the dataset, *Tweet-text* is the given text of the tweet, *HS* which is given in the training data and to be predicted in the test set, if the *Tweet-text* is hate speech the value is 1, otherwise the value is 0.

The original English datasets consisted of 9000 tweets for training, 1000 for development, and 3000 for testing. The distribution is basically balanced among hate (42%) and no-hate (58%) tweets. Due to a lack of label answers for test data sets, 10-fold cross validation is used here for training and validation, and the 1000 development set used as the test data.

During the machine learning methods, preprocessing of raw tweets is usually the first step. Data preprocessing usually consists of the removal of URLs, numbers, users, times, date, email, percentages because these symbols may not contain information that can determine whether the text is hate speech. However, data preprocessing did not deliver significant improvement. This is primarily because we have adopted deep learning methods. Keeping the original information of the text can reserve all the information and may be more conducive to the text classification in deep learning. Based on this, for all the methods in this paper, the input data is the same raw data.

B. EXPERIMENTAL RESULTS

In order to provide a measure that is independent of the class size, the results will be evaluated by accuracy and F1-score which are the official metric used for this binary classification task and the key performances indicators. Accuracy is the proportion of correct classifications among all classifications. F1-score gives the harmonic mean of precision and recall. Two experiments from different angles were designed for analysis.

TABLE 1. The performances of ELMo, BERT and CNN.

Classifier	Accuracy	F1-score
ELMo	0.702	0.636
BERT	0.701	0.623
CNN	0.732	0.698

TABLE 2. The diversity measures of ELMo, BERT and CNN.

Classifier Pair	Q-statistics
ELMo, BERT	0.697
ELMo, CNN	0.768
BERT, CNN	0.723

1) EXPERIMENT 1

During this experiment, we adopted three excellent deep learning methods as described above, which are ELMo, BERT and CNN.

For ELMo, we selected the ELMo model implemented and pre-trained by Google based on tensorflow. The weight parameters of the model can be trained in the specific task.

Considering the memory of the GPU, the $\text{BERT}_{\text{base}}$ uncased model was chosen for fine tuning to avoid the memory overflow of the GPU. According to the official suggestion of BERT, we set the maximum sentence length as 64, the mini-batch as 32, the learning rate as 2e-5 and the number of training epochs as 3.0.

Building a CNN architecture means that there are many hyperparameters to choose from, such as the input representations, number and sizes of convolution filters, pooling strategies, activation functions and so on. A few results highlight that max-pooling always beats average pooling and the ideal filter sizes are important but task-dependent. In this experiment several specific hyper parameters are set as suggestions, the dimensionality of character embedding is 128, the comma-separated filter size is '3,4,5', and the number of filters per filter size is 128.

After designing the three classifiers, we applied them to the same data set to obtain the classification results. The results of classification are given in the form of the predicted probability of label 0 and label 1, which can be used for the subsequent classifiers fusion and converted to the predicted labels. The performance indicators can be calculated statistically by comparing the labels.

The separate classification results of each classifier are shown in TABLE 1. We can see from the table that CNN performed the best followed by ELMo when the three methods BERT, ELMo, CNN were classified independently. However, the differences in performance indicators such as accuracy and F-measure are small. It means that the three methods are relatively balanced in performance.

The diversity measures of ELMo, BERT and CNN are shown in TABLE 2. The Q-statistics values are smaller than 1, meaning there is still some room for improvement in the fusion performance. Due to different deep learning algorithms, the diversity of the three separate classification results is relatively good.

TABLE 3. The final fusion results for ELMo, BERT and CNN.

Fusion Method	Accuracy	F1-score
vote	0.741	0.698
mean	**0.750**	**0.704**
max	0.746	0.700
product	0.749	0.704

TABLE 4. The performances of CNN0, CNN1 and CNN2.

Method	Accuracy	F1-score
CNN0	0.732	0.698
CNN1	0.735	0.672
CNN2	0.733	0.690

Then the basic results are fused through the combination rules by voting, meaning, maximizing and production, and the fusion results are shown in TABLE 3. It can be seen from TABLE 3 that these performances are better than the performances of the original three methods, and the performance of the mean fusion method is the best. To be specific, the best accuracy value in TABLE 3 is improved by 6.99% compared with the minimum accuracy in TABLE 1; and improved by 2.46% compared with the maximum accuracy in TABLE 1. The best F1-score in TABLE 3 is improved by 13.00% compared with the minimum F1-score in TABLE 1; and improved by 0.86% compared with the maximum F1-score in TABLE 1. ELMo, Bert and CNN are the most advanced methods which can be considered as baseline methods in recent years. Based on the same dataset, the performances comparison between TABLE 1 and TABLE 3 testify the effectiveness of the fusion method.

2) EXPERIMENT 2

During the first test, the CNN text classification performs best in the original three methods. And it inspired us that we can change the experimental parameters of the CNN to get different results. Thus in this experiment we adopt three classifiers based on CNN.

In this experiment the CNN0 is from the first test, in which several specific hyperparameters are set as suggestion, the dimensionality of character embedding is 128, the comma-separated filter sizes is '3,4,5' and the number of filters per filter size is 128. For CNN1 we change the dimensionality of character embedding as 256, the other parameters remain the same as CNN0. For CNN2 the comma-separated filter sizes are changed to '2,3,4,5', whilst the other parameters remain the same as in CNN0.

The separate classification results of CNN0, CNN1 and CNN2 are shown in TABLE 4. We can see from Table 4 that the text classification performances are changed after changing the parameters of the CNN classification. However, the overall performances are comparable.

The diversity measures of the CNN0, CNN1 and CNN2 are shown in TABLE 5. The Q-statistics values are bigger than those in TABLE 2; it means that the diversity between the classification results is lower than the first experiment. Compared with the first experiment, the diversity decreases mainly because the classifier algorithms in this experiment belong to the same class with the parameters changed.

TABLE 5. The diversity measures of CNN0, CNN1 and CNN2.

Classifier Pair	Q-statistics
CNN0, CNN1	0.897
CNN0, CNN2	0.914
CNN1, CNN2	0.941

TABLE 6. The final fusion results for CNN0, CNN1, CNN2.

Fusion Method	Accuracy	F1-score
vote	0.752	0.705
mean	**0.753**	0.708
max	0.746	**0.712**
product	0.751	0.706

The final classification results are shown in TABLE 6 by fusing according to the combination rules. We can see that all the performance indicators are improved after fusing. The best accuracy is obtained in the mean fusing method and the best F1-score is obtained in the max fusing method. Although the diversity in this test is worse than in the first test, the results after fusing are still better than the first test, it is mainly because the three separate results of the second experiment are better than the first experiment. To be specific, the best accuracy value in TABLE 6 is improved by 2.87% compared with the minimum accuracy in TABLE 4; and improved by 2.45% compared with the maximum accuracy in TABLE 4. The best F1-score in TABLE 6 is improved by 5.95% compared with the minimum F1-score in TABLE 1; and improved by 2.00% compared with the maximum F1-score in TABLE 4.

According to the above two sets of experiments, we can conclude that whether by designing several different classifiers or using the same type of classifier with different parameters, the fusion of different classifier results can improve the classification accuracy and F1-score. And this approach improves the results with little additional cost.

V. CONCLUSION

This paper presented the principle of three types of text classification methods, ELMo, BERT and CNN, and applied them to hate speech detection, then improved the performance by fusion from two perspectives: the fusion of the classification results of ELMo, BERT and CNN, and the fusion of the classification results of three CNN classifiers with different parameters. The results showed that fusion processing is a viable way to improve the performance of hate speech detection. It can be deemed reasonable to achieve the practical significance of performance at a little extra cost.

This paper focuses on the fusion after separate classification; the degree of integration is not deep enough. In the future we will pay more attention to the early cooperation before classification. We will try to replace the basic word vector expression in CNN with the embedding technologies in ELMo or BERT. This can integrate the advantages of excellent word embedding and powerful neural networks deeply.

REFERENCES

[1] O. de Gibert, N. Perez, A. García-Pablos, and M. Cuadros, "Hate speech dataset from a white supremacy forum," in *Proc. 2nd Workshop Abusive Lang. Online (ALW2)*, 2018, pp. 11–20.

[2] T. Davidson, D. Bhattacharya, and I. Weber, "Racial bias in hate speech and abusive language detection datasets," in *Proc. 3rd Workshop Abusive Lang. Online*, 2019, pp. 25–35.

[3] T. Davidson, D. Warmsley, M. Macy, and I. Weber, "Automated hate speech detection and the problem of offensive language," in *Proc. ICWSM*, May 2017, pp. 512–515.

[4] E. Cambria, "Affective computing and sentiment analysis," *IEEE Intell. Syst.*, vol. 31, no. 2, pp. 102–107, Mar./Apr. 2016.

[5] P. Badjatiya, S. Gupta, M. Gupta, and V. Varma, "Deep learning for hate speech detection in tweets," in *Proc. 26th Int. Conf. World Wide Web Companion WWW Companion*, 2017, pp. 759–760.

[6] M. Bojkovský and M. Pikuliak, "STUFIIT at SemEval-2019 task 5: Multilingual hate speech detection on Twitter with MUSE and ELMo embeddings," in *Proc. 13th Int. Workshop Semantic Eval.*, 2019, pp. 464–468.

[7] M. S. Akhtar, A. Ekbal, and E. Cambria, "How intense are you? Predicting intensities of emotions and sentiments using stacked ensemble [application notes]," *IEEE Comput. Intell. Mag.*, vol. 15, no. 1, pp. 64–75, Feb. 2020.

[8] H. Liu and L. Zhang, "Advancing ensemble learning performance through data transformation and classifiers fusion in granular computing context," *Expert Syst. Appl.*, vol. 131, pp. 20–29, Oct. 2019.

[9] H. Watanabe, M. Bouazizi, and T. Ohtsuki, "Hate speech on Twitter: A pragmatic approach to collect hateful and offensive expressions and perform hate speech detection," *IEEE Access*, vol. 6, pp. 13825–13835, 2018.

[10] Z. Wang, S.-B. Ho, and E. Cambria, "A review of emotion sensing: Categorization models and algorithms," *Multimedia Tools Appl.*, pp. 1–30, Jan. 2020, doi: 10.1007/s11042-019-08328-z.

[11] Z. Waseem and D. Hovy, "Hateful symbols or hateful people? Predictive features for hate speech detection on Twitter," in *Proc. NAACL Student Res. Workshop*, Jun. 2016, pp. 88–93.

[12] P. Burnap and M. L. Williams, "Cyber hate speech on Twitter: An application of machine classification and statistical modeling for policy and decision making," *Policy Internet*, vol. 7, no. 2, pp. 223–242, Jun. 2015.

[13] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol., (Long Papers)*, vol. 1, 2018, pp. 2227–2237.

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, vol. 1, Jun. 2019, pp. 4171–4186.

[15] M. Mozafari, R. Farahbakhsh, and N. Crespi, "A BERT-based transfer learning approach for hate speech detection in online social media," in *Int. Conf. Complex Netw. Their Appl., Dec.*, vol. 2019, pp. 928–940.

[16] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. EMNLP*, Oct. 2014, pp. 1746–1751.

[17] B. Gambäck and U. K. Sikdar, "Using convolutional neural networks to classify hate-speech," in *Proc. 1st Workshop Abusive Lang. Online*, 2017, pp. 85–90.

[18] Z. Zhang, D. Robinson, and J. Tepper, "Detecting hate speech on Twitter using a Convolution-GRU based deep neural network," in *Proc. Eur. Semantic Web Conf.*, Jun. 2018, pp. 745–760.

[19] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proc. ICLR Workshop Papers*, 2013, pp. 1–12.

[20] X. Yin, Y. Huang, B. Zhou, A. Li, L. Lan, and Y. Jia, "Deep entity linking via eliminating semantic ambiguity with BERT," *IEEE Access*, vol. 7, pp. 169434–169445, 2019.

[21] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics (Long Papers)*, vol. 1, 2018, pp. 328–339.

[22] Y. Goldberg, "Neural network methods for natural language processing," *Synth. Lectures Human Lang. Technol.*, vol. 10, no. 1, pp. 1–309, Apr. 2017.

[23] M. Sammons, C. Christodoulopoulos, P. Kordjamshidi, D. Khashabi, V. Srikumar, and D. Roth, "Edison: Feature extraction for nlp, simplified," in *Proc. 10th Int. Conf. Lang. Resour. Eval.*, May 2016, pp. 4085–4092.

[24] L. I. Kuncheva and C. J. Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy," *Mach. Learn.*, vol. 51, no. 2, pp. 181–207, 2003.

[25] V. Basile, C. Bosco, E. Fersini, D. Nozza, V. Patti, F. M. R. Pardo, P. Rosso, and M. Sanguinetti, "SemEval-2019 task 5: Multilingual detection of hate speech against immigrants and women in Twitter," in *Proc. 13th Int. Workshop Semantic Eval.*, Jun. 2019, pp. 54–63.

YANLING ZHOU received the B.S. degree in communication engineering from Hubei University, China, in 2000, and the M.S. and Ph.D. degrees from the Huazhong University of Science and Technology, China, in 2004 and 2012, respectively. From 2013 to 2015, she was a Lecturer with Hubei University, where she has been an Associate Professor, since 2015. Her research interests include machine learning and navigation signal processing.

YANYAN YANG received the B.E. and M.S. degrees in computer software, and the Ph.D. degree in computer science from Peking University, China, in 1993, 1996, and 1999, respectively. She was a Lecturer with Robert Gordon University and a Senior Research Fellow of the University of St. Andrews, U.K., and Cardiff University, U.K. She is currently a Senior Lecturer with the University of Portsmouth, U.K. She has worked on U.K. and European funded research projects and has published over 50 papers in refereed journals and conferences. Her research interests include recommendation systems, information retrieval, and data mining as well as their applications in social networks, health informatics, and business. She is a member of the IET, WES, and HEA.

HAN LIU (Member, IEEE) received the B.Sc. degree in computing from the University of Portsmouth, in 2011, the M.Sc. degree in software engineering from the University of Southampton, in 2012, and the Ph.D. degree in machine learning from the University of Portsmouth, in 2015. He has published two research monographs in Springer and over 65 articles in areas such as data mining, machine learning, and intelligent systems. His research interests include data mining, machine learning, rule-based systems, intelligent systems, fuzzy systems, pattern recognition, big data, granular computing, and computational intelligence.

XIUFENG LIU received the bachelor's and master's degrees in computer science from Tsinghua University, China, in 2001 and 2005, respectively, and the Ph.D. degree in computer science from Aalborg University, Denmark, in 2012. He is currently a Senior Researcher with the Department of Management Engineering, Technical University of Denmark. He is also a member of the research group of climate change and sustainable development within the university. He has published over 60 papers in refereed journals and conferences. His research interests include smart meter data analytics, data management (in particular, data warehousing), big data, data mining, and machine learning.

NICK SAVAGE (Member, IEEE) received the Ph.D. degree from the University of Portsmouth. He joined the University of Portsmouth, in 2000, to work as a Researcher on an EPSRC project characterizing the indoor communications channel and a radio communications agency (now OFCOM) sponsored project characterizing radio wave propagation through vegetation. He started his work as a Lecturer with the Department of Electronics and Computer Engineering, University of Portsmouth, in 2002. He is currently a Principal Lecturer with the School of Computing, University of Portsmouth. His research interests include the development of secure systems, identifying vulnerabilities in systems, utilizing features to identify users, and network protocol analysis.

• • •