Neural Computing and Applications (2020) 32:10809–10818
https://doi.org/10.1007/s00521-018-3442-0

S.I: INDIA INTL. CONGRESS ON COMPUTATIONAL INTELLIGENCE 2017

Verbal aggression detection on Twitter comments: convolutional neural network for short-text sentiment analysis

Junyi Chen[1] · Shankai Yan[1] · Ka-Chun Wong[1]

Received: 28 November 2017 / Accepted: 16 March 2018 / Published online: 20 March 2018

Abstract
Cyberbullying and hate speeches are common issues in online etiquette. To tackle this highly concerned problem, we propose a text classification model based on convolutional neural networks for the de facto verbal aggression dataset built in our previous work and observe significant improvement, thanks to the proposed 2D TF-IDF features instead of pre-trained methods. Experiments are conducted to demonstrate that the proposed system outperforms our previous methods and other existing methods. A case study of word vectors is carried out to address the difficulty in using pre-trained word vectors for our short-text classification task, demonstrating the necessities of introducing 2D TF-IDF features. Furthermore, we also conduct visual analysis on the convolutional and pooling layers of the convolutional neural networks trained.

Keywords Aggression detection · Sentiment analysis · Machine learning · Convolutional neural network

1 Introduction

Sentiment analysis and opinion mining task [1] is one of the well-studied fields in text mining and natural language processing. It aims at detecting and analyzing human opinions, attitudes, and emotions. Application scenarios of sentiment analysis can stem from product reviews [2], advertisement distribution, stock market [3, 4], social networks [5, 6] or even government intelligence [7].

Many research and famous datasets of sentiment analysis such as IMDB [8] and Yelp [9] acquaint positive or negative opinion as known as PNO [10] about a certain object from user comments. Contrariwise, we focus on the behavior of users by capturing verbal offense which potentially arouses negative feelings among other users. Recently, we built a de facto network comment dataset with 'aggressivity' label and adopted predictive models to detect verbal offenses [11]. The dataset included manually collected paragraphs and paragraphs from 'Sentimen140' [12] with labels renovated. Combined with WordNet [13] lemmatizer and Porter's stemmer [14], support vector machine [15] and logistic regression [16] can achieve decent performance with F1-scores [17] greater than 0.80 on the 783 pieces of aggressive and unaggressive comments without any extensive hyper-parameter tuning. Despite those two methods achieve good results on our verbal offense dataset, we were looking for models that can outperform our previous ones for verbal aggression detection.

Convolutional neural networks (CNN) [18] are originally designed to process and learn information from image features by applying convolution kernels and pooling techniques which are widely adopted for extracting stationary features; for instance, CNN has shown its adaptability in the field of text mining and NLP tasks. Kim et al. reported series of experiments with CNNs [19] that achieve good results on sentence classification and sentiment analysis tasks. Lee et al. propose a weakly supervised CNN architecture [20] to identify discriminating keywords in PNO tasks. Inspired by the successful examples of CNN applications in the field of text classification, we introduce a CNN model to detect verbal offenses from

✉ Ka-Chun Wong
kc.w@cityu.edu.hk

Junyi Chen
junyichen8-c@my.cityu.edu.hk

Shankai Yan
sk.yan@my.cityu.edu.hk

[1] City University of Hong Kong, Kowloon Tong, Kowloon, Hong Kong SAR

the aggression dataset we collected in the previous research to look for performance enhancement.

The contribution of this work is to further improve the performance of the sentiment analysis task we previously proposed by introducing an efficient CNN-based deep learning model. In addition, by testing different kinds of models and methods, we discovered some interesting CNN architectures which can outperform others.

2 System modeling

2.1 Model architecture

The model architecture of CNN in the present paper is derived from Kim [19] and Lee et al. [20]. Motivated by those successful results attained in the aforementioned works, we design the network architectures by referencing former experiences. According to Lee's statement, applying a large number of filters rather than deep architectures is good for text classification. We set 128 filters on the convolution layer, and each of them is 10×2 rectangular-shaped matching our 100×20 inputs. Also, after comparison experiments which will be discussed in the next section, we decide to use mean pooling rather than max pooling as an optional choice in the pooling layer. Finally, a two-layered multilayer perceptron [21] is introduced to be the classification model following the pooling layer. Summarizing these aspects, our model structure is shown in Fig. 1 and detailed settings will be shown in the experiments. Further justification of the architecture will be discussed in the Discussion section.

2.2 Word features

The model mentioned above use word embeddings techniques like Word2Vec [22] or Glove [23] to represent word features. Trained by a self-supervised model, words are transformed into similar vectors if they have similar verbal meanings. However, these approaches highly rely on the article context. Our dataset contains short passages from Twitter and usually contains short sentences in each document. Sentiment analysis on short texts is a difficult business [24]. This organization of data makes Word2Vec and Glove hard to learn information of each word by its 'context.' The situation that our task does not favor context-based pre-trained embedding methods will be shown afterward in the case study part. Another word embedding method presented in Gal et al. [25] is a method that treats word embedding on-the-fly. Encoded one-hot word vectors are linearly combined by an embedding layer whose weights are updated by backpropagation in the network. This is the method adopted in our experiment.

Alternatively, we apply TF-IDF [26] matrix as document-level features in the present work. In such short paragraph text classification problem, we think that word occurrence statistics are appropriate solutions which contain enough information although TF-IDF is a traditional method. Furthermore, to better utilize the CNN property, the 1-dimensional vector is transformed to a 2-dimensional matrix, and hence, CNN filters can convolve word features in a larger field as shown in Fig. 2. Features in 1-dimensional only share the same weight with other features within the window size, while 2-dimensional features can share weights with features in the next row which are unreachable in 1-dimensional case. Theoretically, 2-dimensional features with 2-dimensional kernel can capture features from a larger space than single dimensions.

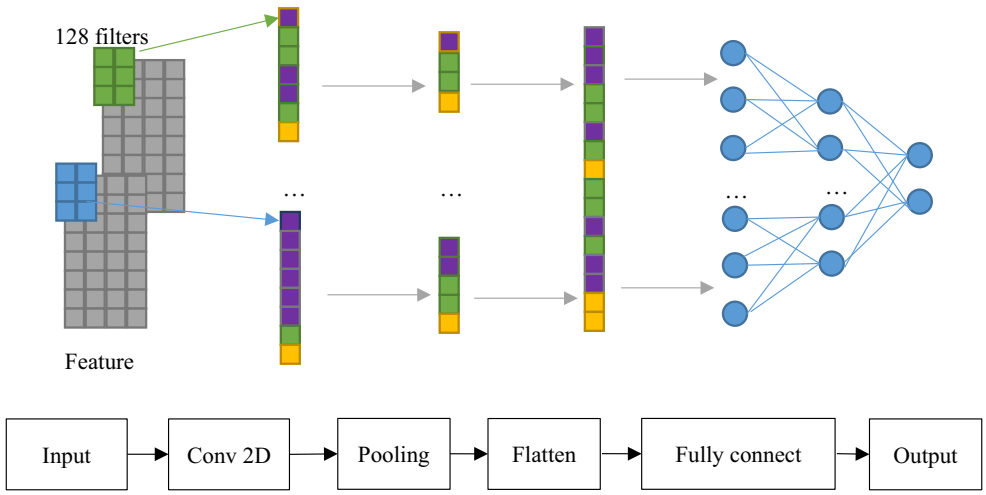

Fig. 1 Model structure of the proposed convolutional neural networks

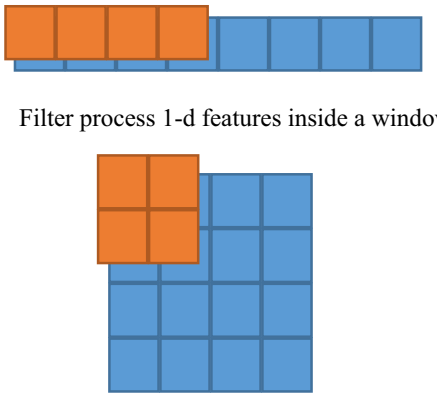

Fig. 2 Comparison between 1-D feature and 2-D feature, each element in the vector/matrix is an attribute from TF-IDF document representation

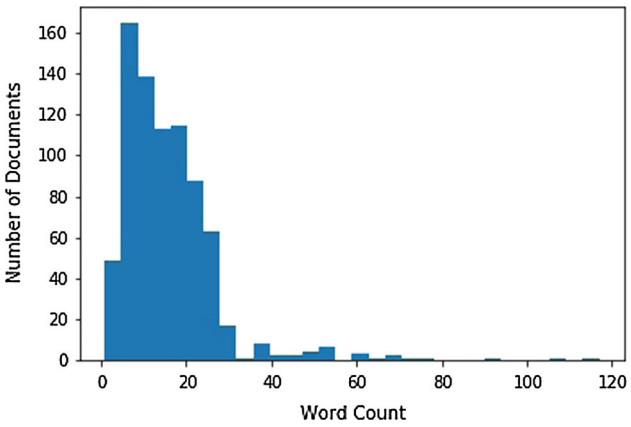

Fig. 3 Word count distribution of documents in the dataset

3 Experiment settings

3.1 Dataset and preprocessing

As it is shown in our previous paper [11], the number of features after document encoding (regardless of lemmatizing methods) is around 2000 tabulated in Table 1. With such dictionary size, we limit the maximum number of features to 2000 features as sorted by their TF-IDF weights (ascending order). To obtain the optimal solution, different word tokenizing methods (Porter and WordNet) are compared.

Apart from the dictionary size, word count is another import statistic of the dataset. Figure 3 is a histogram to show the relation between word count and document count. Under the constraint of 140 words in Twitter, most comments in the dataset contain less than 40 words. Few of them contain 60–80 words, while fewer contain around 100 words. This property will affect the training result of Word2Vec which will be discussed in the case studies.

3.2 Learning algorithms and models

To verify the efficiency of the presented model, baseline models from previous work and other typical frameworks are introduced to the experiment. Settings of those models to be examined are listed in Table 2.

Firstly, the CNN models mentioned in the previous sections which are capable of both TF-IDF matrix and word embeddings are introduced to the experiment. Except for the proposed CNN architecture, two algorithms that can achieve desirable results in the previous research [11], support vector machine and logistic regression with stochastic gradient descent, are included in the experiment. These two methods are baselines for comparison in this study. Thirdly, a modification of recurrent neural network (RNN) [27] called long short-term memory also known as 'LSTM' [28] is also tested for the widely successful applications of recurrent networks. Mikolov et al. [29] introduce recurrent network to language model. Yang et al. [30] design an infrastructure that capture sentence and word-level attentions using gated recurrent units [31] which is also a modified LSTM. Based on these successful attempts, LSTM is also tested in our experiment using settings indicated in Table 2.

Table 1 Feature numbers tokenized by non-lemmatized, WordNet lemmatizer and Porter's stemmer

Lemmatization	Number of features
Non-lemmatized	2257
Porter's	2028
WordNet	2196

3.3 Performance benchmarking

Performance benchmarking is held based on holdout validation method [32] which means the original dataset is randomly divided into training set and test set. In our experiment, 60% of the original data is used to train the model, while 40% of data is used for testing. Performance metrics are accuracy and area under the curve (AUC) of the receiver operating characteristic (ROC) [33].

4 Results

The results of our present approaches against some other experiments are shown as follows. Table 3 and Fig. 4 depict the accuracy values and AUC values of different

Table 2 Hyper-parameters settings of different models

Model	Settings
Convolutional neural network (CNN)	1. Convolution layer
	Filters: 128; kernel size: 10 × 2; activation: RELU, strides: 1 × 1
	2. Average pooling layer pool size: 2 × 2, strides: 2 × 2
	3. Dense layer
	Units: 256, activation: tanh
	4. Dense layer
	Units: 2, activation: soft-max
	Loss: cross-entropy; optimizer: Adam
Long short-term memory (LSTM)	1. LSTM layer 1
	Units: 256; dropout: 0.2; recurrent dropout: 0.2
	2. LSTM layer 2
	Units: 256; dropout: 0.2
	3. 2 dense layers' settings are the same with CNN
	Loss: cross-entropy; optimizer: Adam
Support vector machine (SVM)	Kernel: linear; C: 1.25, degree: 1; tolerance: 0.001
Logistic regression (logistic)	Loss: log; alpha: 0.00075; penalty: l1

measures adopted in the experiment. Deep learning algorithms are expected to gain better performance as evidenced by many other similar studies. Compared with the baseline methods we experimented in the previous research, LSTM model is proved to be enhanced with 0.91 as accuracy and 0.96 as Macro-AUC. By contrast, the proposed CNN model with 2-dimensional TF-IDF matrix results in further improvement with accuracy equals to 0.92 and Macro-AUC equals to 0.98.

Apart from numeral performance benchmarks, Fig. 4 depicts the ROC curves of different methods for illustrations. Magenta dash line is a microaverage curve, navy blue dash line denotes macroaverage [34] curves, while cyan and yellow are curves with respect to each class. The curves stay near the upper-left corner, implying that the corresponding model achieves good performance. We can see that not only accuracy but also ROC curves demonstrate that CNN with 2D TF-IDF matrix is a superior solution to our problem.

4.1 Embedding layer and TF-IDF

Another alternative for feature construction is to introduce embedding layer proposed in [25] by adding a lookup table also known as one-hot word vectors and an embedding weights layer. Document matrices are represented by the concatenation with paddings of word vectors. Embedding approach enables us to learn in word level rather than document level. Table 4 and Fig. 5 show some statistic and analysis of embedding experiments.

As shown in the table and figures above, the accuracy of CNN drops from 0.92 to 0.83, while the accuracy of LSTM drops from 0.91 to 0.72. We can be informed that, although embedding method reserves word level and ordinal information, it compromises the prediction performance. Both CNN and LSTM with embedding layer output worse performance compared with TF-IDF feature at the document level.

4.2 WordNet and Porter

'WordNet' and 'Porter' are two stemming methods to group words by their lexical roots: 'WordNet' group words by meanings while 'Porter' group word by word roots. In the previous paper, we tested two methods on support vector machine and logistic regression and concluded that on these two models. 'WordNet' is an optimal solution of stemming. To our surprises, the present CNN model reveals an opposite conclusion shown in Fig. 6 and Table 5. According to the figure and table, Porter's stemmer outputs 0.85 accuracy and 0.94 Macro-AUC which are 7 and 4% less than 0.92 and 0.98, respectively, using WordNet. Porter's stemming is a more aggressive scheme than WordNet and lose some information in when doing word tokenizing.

4.3 Model generalization analysis

In this section, we discuss the generalization of the model using learning curve. The learning curves are generated by increasing the size of the training set. If testing score decreases along with the increment of training set size, the model tends to suffer from the overfitting problem. According to Fig. 7, the testing accuracy keeps increasing and both accuracies converge at a value larger than 0.9. Hence, the learning curve demonstrates that the network

Table 3 Performances of different models (SVM, logistic, LSTM, CNN)

Method	Accuracy	Micro-AUC	Macro-AUC
Support vector machine (SVM)	0.86	0.91	0.89
Logistic regression (logistic)	0.86	0.92	0.89
Long short-term memory (LSTM + 2D TF-IDF)	0.91	0.96	0.93
Convolutional neural network (CNN + 2D TF-IDF)	0.92	0.98	0.97

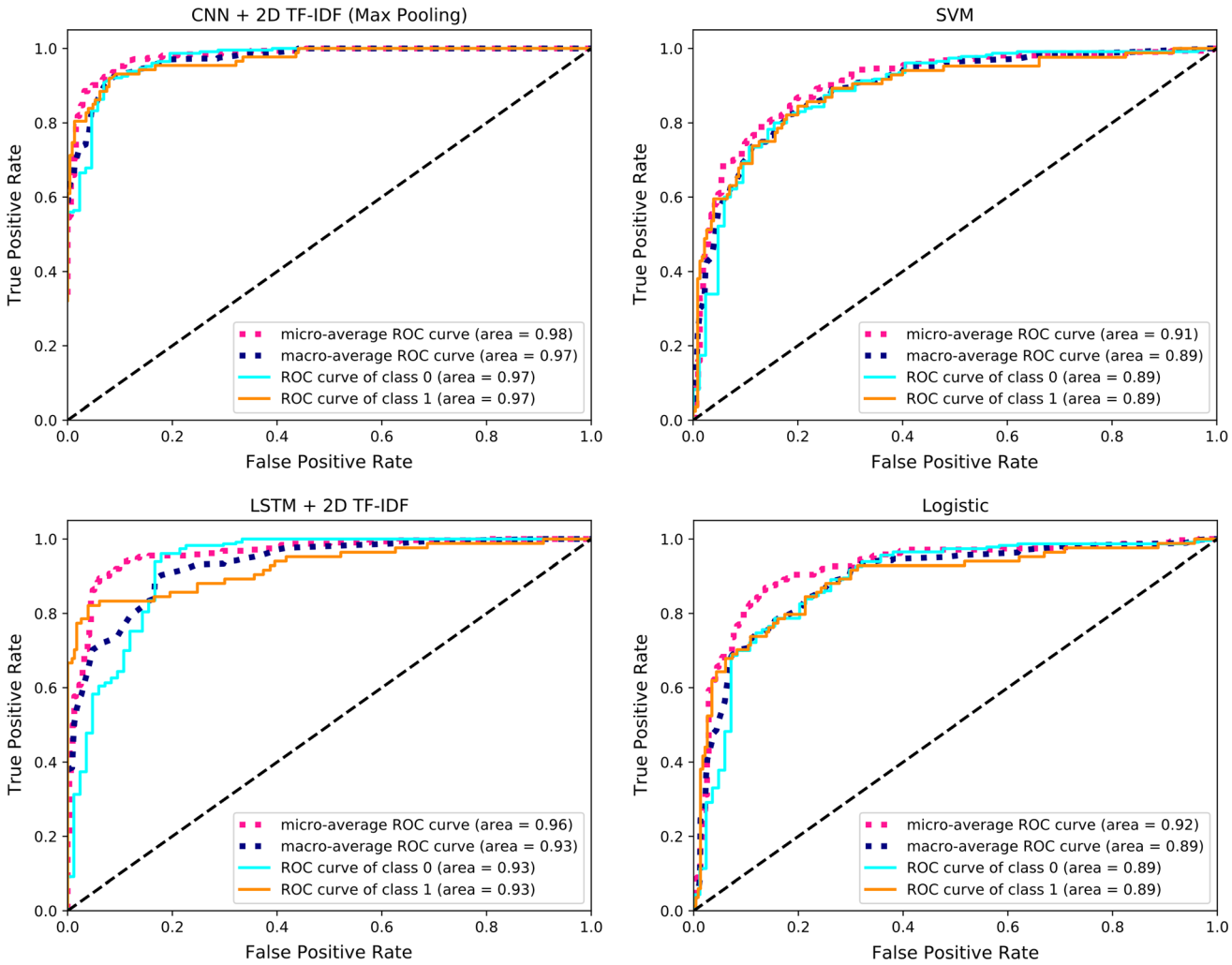

Fig. 4 ROC curves of the convolutional neural network (CNN + 2D TF-IDF, max pooling), long short-term memory (LSTM), support vector machine (SVM) and logistic regression (logistic)

Table 4 Performances of introducing embedding layer to CNN and LSTM

Method	Accuracy	Micro-AUC	Macro-AUC
LSTM + EMBEDDING	0.72	0.81	0.75
CNN + EMBEDDING	0.83	0.88	0.85

achieves decent testing performance as well as a good ability of generalization.

5 Discussion

5.1 Limitations of pre-trained embedding

In this section, we demonstrate a case study on the Word2Vec method presented by Mikolov et al. [22]. Before training the proposed models for experiments, we have finished the word vector pre-trained procedure and observed the generated word vectors. We examined a set of word vectors and found that most of the pre-trained word vectors did not preserve the desired properties of Word2-Vec: consistency between vector similarity and word similarity.

Some sample words and their nearest neighbors ordered by similarity are shown in Table 6. We select words which are positive (successful, happy), negative (worst, lose) and neutral (everyone). Many of the words are not 'well trained' and failed to reserve word meanings. Short paragraph and small dataset (shown by Fig. 3) hinder the performance of Word2Vec training which highly relies on context no matter by the continuous bag of words (predict word by context) or skip gram (predict context by word) [22]. The limitation of the pre-trained model is shown in our dataset.

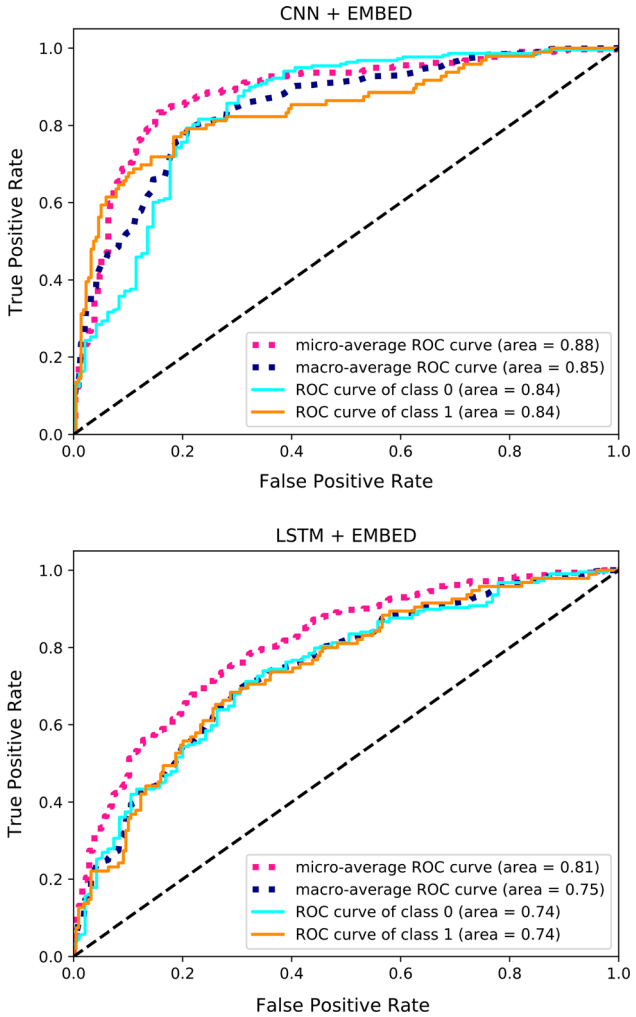

Fig. 5 ROC curves of introducing embedding layer to CNN (up) and LSTM (down)

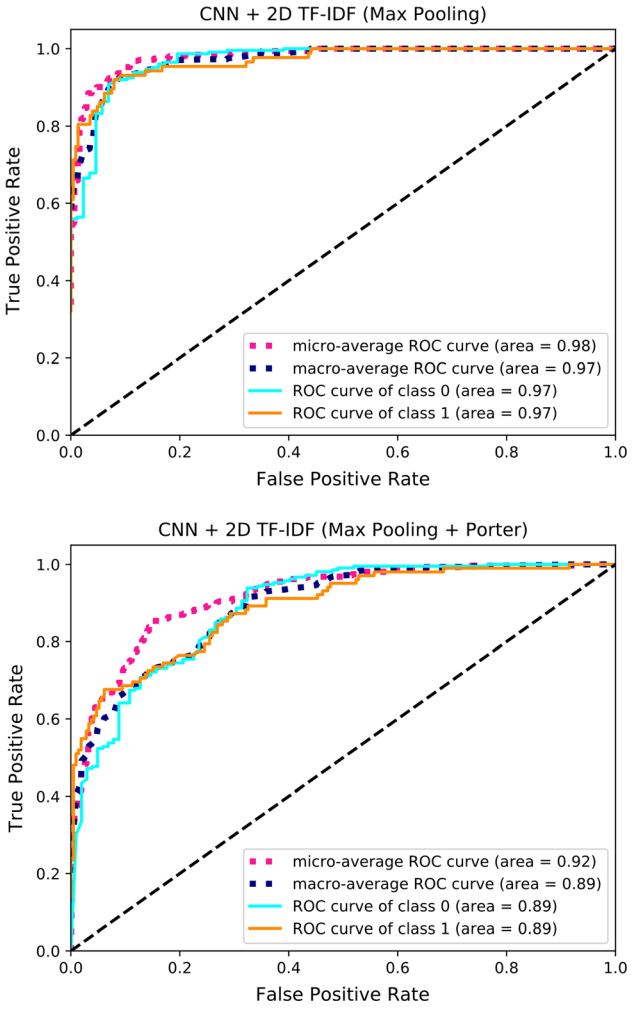

Fig. 6 ROC curves of WordNet lemmatizing (up) and Porter's stemmer (down)

5.2 Visual analysis of convolution layer and pooling layer

After benchmarking of the model, some case studies on our CNN model are carried out. One property of CNN concerns about applying convolution operations with trained kernels on the original feature matrix to filter out new feature sets. Our 2D TF-IDF matrix can be regarded as a single channel picture with 100×20 pixels. To better observe the behavior of CNN, we construct a dictionary document (a document contains every single word once excluding stop words in the dictionary) and see how. The TF-IDF transformed dictionary is shown in Fig. 8. Attributes with higher TF-IDF scores are highlighted in yellow color in the picture. Otherwise, attributes with low TF-IDF scores are purple pixels. Shown by the dictionary figure, features in the TF-IDF matrix are ranked by word counts inside the field of the picture. Intuitively, a dictionary is a color gradient bar change from purple to yellow with a small amount of noise because a word with a larger number of counts does not pick up some examples from guaranteed words to gain a higher TF-IDF score. The picture of the processed dictionary convolved by 128 trained filters is shown in Fig. 8, while some samples before and after going through the pooling layer are also shown.

Table 5 Performances of WordNet lemmatizing and Porter's stemmer

Lemmatizing	Accuracy	Micro-AUC	Macro-AUC
WordNet	0.92	0.98	0.97
Porter's	0.85	0.94	0.91

We conduct a case study to observe pooling effect in terms of text classification. Figure 8 shows features before pooling in the first row while the features after pooling in the second row. Visually, the picture of the feature after

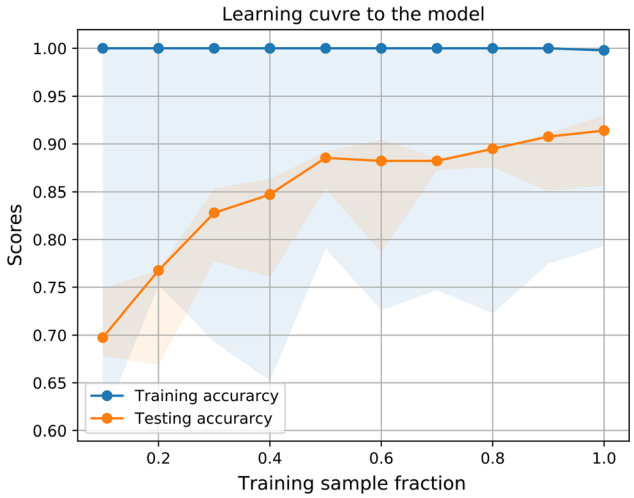

Fig. 7 The learning curves of the proposed CNN model, lower (upper) bound of the filled area is the minimum (maximum) score within 15 epochs

Table 6 Similar word samples

Sample word	Most similar words (order by similarity)
Everyone	Zac, winner, across, freaky, plane…
Worst	Definitely, jail, three, cast, everything…
Lose	Proudly, warren, absolutely, tomorrow…
Successful	Picture, dad, stock, background, apartment…
Happy	Program, wait, foil, belong, may…

pooling is 'brighter' which may imply feature values are amplified after pooling. We also conclude some patterns and pick up some examples of the processed features. The second leftmost column has sparse active features only in some region of the matrix, while most of the values are close to zero. This pattern implies that the convolution layer 'filters out' crucial features. Another pattern shown in the third column is that convolution layer 'wipes out' some features in a certain region (bottom from the picture) from original features. The third discovered pattern in the fourth column is that the outputted feature is almost identical to the original one except 'diluting' the original values. Finally, the patterns from the rightmost column 'highlight' attributes from a certain region while minimizing feature values outside the region. From the scope of the convolved dictionary, we can be implied that features are enriched by the convolution layer and all these new implicit features will be flattened and go through the training process of two fully connected neural network.

To further explore the functionality of pooling layer, we perform experiments by replacing average pooling layer by

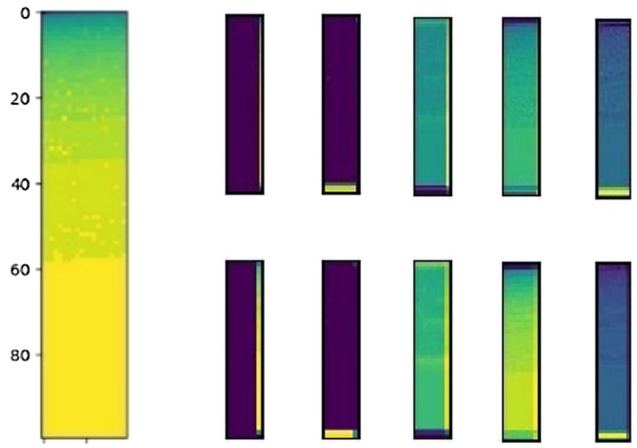

Fig. 8 Dictionary as input feature (left), some convolved features (upper right) and some pooled features (lower right)

max pooling. Figure 9 shows plots of average pooling output and max pooling output; we noted that these outputs came from different training process. Comparing max pooling with average pooling, max pooling outputs more features with zero values. We assume that, with richer features, the average pooling should output superior results than max pooling.

Nonetheless, the scope of performance benchmarking in Table 7 shows a contrary result. The max pooling outputs 0.92 accuracy, 0.98 Micro-AUC and 0.97 Macro-AUC, while average pooling outputs 0.92, 0.96 and 0.95, respectively. The max pooling layer in our experiment shows stronger ability to filter out necessary features to avoid overfitting than the average pooling layer.

Using the analysis method above, we sample one aggressive sentence and one non-aggressive sentence from the dataset with some measurements shown in Table 8 and plot the visualized aggressive sentence in Fig. 10. Since the example sentence covers a small set of words in the whole dictionary, the input features are sparse since only a few areas in the plots are colored.

According to Table 8, bold words are the most informative ones, while italic words are the second. Words that implicit the aggressive sentiment in each sentence obtain high scores using our feature construction approach. Then, the convolutional layer and the pooling layer further extract and filter these features. These functions of the two layers are reflected by the alteration of the colored region shown by the plots in Fig. 10.

5.3 Architecture design experiments

The proposed network architecture is rooted from Kim [19] which trains a convolutional neural network with one layer of convolution. To examine the design principle, we test

Fig. 9 Overview of the 128 trained filters: averaged pooling layer (up) contains more nonzero values, while max pooling layer (down) contains more zero values

Table 7 Performances of max pooling and average pooling

Pooling	Accuracy	Micro-AUC	Macro-AUC
Max 2D	0.92	0.98	0.97
Average 2D	0.92	0.96	0.95

different architectures with alternative numbers of convolutional layers (followed by max pooling layers) and dense layers. The design of architecture is based on the statistics shown in Table 9, and our proposed neural network with 1 convolutional layer and 2 dense layers achieves the best accuracy among all experiment settings.

6 Conclusion

In this paper, we present a new solution to the verbal aggression detection task we aroused in the prerequisite research based on convolutional neural networks (CNN) using 2-dimensional TF-IDF features and observe significant improvement. Firstly, experimental results indicate that CNN model achieves significant improvement compared with the baseline SVM and logistic regression methods in the previous study as well as the newly tested LSTM model in the problem. Moreover, we carried out experiments on the dataset to explain the selection of word lemmatizing. Finally, the problem that pre-trained word

Table 8 Examples of verbal aggression detections and two most informative words

Sentence	Remarks
(1): Not really you voted for him because you re too **intellectually** *lazy* to know better	Aggressive: true 1st score: 5.97 2nd score: 5.57
(2): I wouldn t say I m Wayne Rooney s biggest *fan* but these are the kind of **stories** everyone likes to read He s just made that little lads year Top job	Aggressive: false 1st score: 5.68 2nd score: 5.06

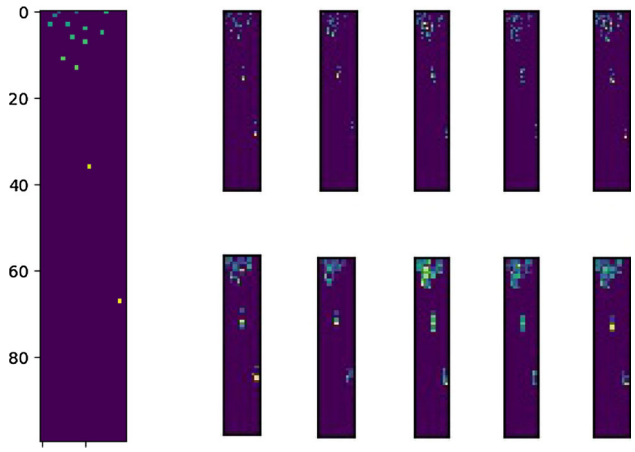

Fig. 10 Aggressive sentence (sentence 1 in Table 8) as input feature (left), some convolved features (upper right) and some max pooled features (lower right)

Table 9 Accuracies of models with different architecture

Convolutional layers	Dense layers	Test accuracy	Number of parameters
1	2	0.92	16,387,458
2	2	0.91	1,109,410
3	2	0.90	302,530
4	2	0.90	175,586
1	3	0.90	16,403,522
1	4	0.90	16,407,682
1	5	0.90	16,411,842

vector method encountered on our dataset is annotated and the preference of pooling strategies is studied by conducting visual analysis on neural layers.

Acknowledgements The work described in this paper was substantially supported by two grants from the Research Grants Council of the Hong Kong Special Administrative Region (CityU 21200816) and (CityU 11203217).

Compliance with ethical standards

Conflict of interest All the authors declare that they have no conflict of interest.

References

1. Pang B, Lee L (2008) Opinion mining and sentiment analysis. Foundations and Trends®. Inf Retrieval 2(1–2):1–135
2. Zhang W, Xu H, Wan W (2012) Weakness Finder: find product weakness from Chinese reviews by using aspects based sentiment analysis. Expert Syst Appl 39(11):10283–10291
3. Long W, Tang Y-R, Tian Y-J (2016) Investor sentiment identification based on the universum SVM. Neural Comput Appl. https://doi.org/10.1007/s00521-016-2684-y
4. Hájek P (2018) Combining bag-of-words and sentiment features of annual reports to predict abnormal stock returns. Neural Comput Appl 29(7):343–358. https://doi.org/10.1007/s00521-017-3194-2
5. Pak A, Paroubek P (2010) Twitter as a corpus for sentiment analysis and opinion mining. In: LREc, vol 10, no. 2010
6. Kouloumpis E, Wilson T, Moore JD (2011) Twitter sentiment analysis: the good the bad and the omg! Icwsm 11(538–541):164
7. Mullen T, Malouf R (2006) A preliminary investigation into sentiment analysis of informal political discourse. In: AAAI spring symposium: computational approaches to analyzing weblogs, pp 159–162
8. Maas AL, Daly RE, Pham PT, Huang D, Ng AY, Potts C (2011) Learning word vectors for sentiment analysis. In: Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies, vol 1. Association for Computational Linguistics, pp 142–150
9. Tang D, Qin B, Liu T (2015) Document modeling with gated recurrent neural network for sentiment classification. In: EMNLP, pp 1422–1432
10. Liu B, Zhang L (2012) A survey of opinion mining and sentiment analysis. In: Mining text data. Springer, New York, pp 415–463
11. Chen J, Yan S, Wong KC (2017). Aggressivity detection on social network comments. In: Proceedings of the 2017 international conference on intelligent systems, metaheuristics & swarm intelligence. ACM, pp 103–107
12. Go A, Bhayani R, Huang L (2009) Twitter sentiment classification using distant supervision. CS224 N Project Report, Stanford, 1(2009), 12
13. Fellbaum C (1998) WordNet. Wiley, New York
14. Porter MF (1980) An algorithm for suffix stripping. Program 14(3):130–137
15. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297
16. Hosmer DW Jr, Lemeshow S, Sturdivant RX (2013) Applied logistic regression, vol 398. Wiley, New York
17. Salton G, McGill MJ (1986) Introduction to modern information retrieval. McGraw-Hill, Inc., New York
18. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
19. Kim Y (2014) Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882

20. Lee G, Jeong J, Seo S, Kim C, Kang P (2017) Sentiment classification with word attention based on weakly supervised leaning. arXiv preprint arXiv:1709.09885
21. Gardner MW, Dorling SR (1998) Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. Atmos Environ 32(14):2627–2636
22. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013). Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems, pp 3111–3119
23. Pennington J, Socher R, Manning C (2014) Glove: global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp 1532–1543
24. Dos Santos CN, Gatti M (2014) Deep convolutional neural networks for sentiment analysis of short texts. In: COLING, pp 69–78
25. Gal Y, Ghahramani Z (2016) A theoretically grounded application of dropout in recurrent neural networks. In Advances in neural information processing systems, pp 1019–1027
26. Salton G, Buckley C (1988) Term-weighting approaches in automatic text retrieval. Inf Process Manag 24(5):513–523
27. Williams RJ, Zipser D (1989) A learning algorithm for continually running fully recurrent neural networks. Neural Comput 1(2):270–280
28. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
29. Mikolov T, Karafiát M, Burget L, Cernocký J, Khudanpur S (2010) Recurrent neural network based language model. In Interspeech, vol 2, p 3
30. Yang Z, Yang D, Dyer C, He X, Smola AJ, Hovy EH (2016) Hierarchical attention networks for document classification. In: HLT-NAACL, pp 1480–1489
31. Bahdanau D, Cho K, Bengio Y (2014) Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473
32. Refaeilzadeh P, Tang L, Liu H (2009) Cross-validation. In: Encyclopedia of database systems. Springer, New York, pp 532–538
33. Fawcett T (2006) An introduction to ROC analysis. Pattern Recognit Lett 27(8):861–874
34. Yang Y (1999) An evaluation of statistical approaches to text categorization. Inf Retrieval 1(1):69–90