NLP at IEST 2018: BiLSTM-Attention and LSTM-Attention via Soft Voting in Emotion Classification

Qimin Zhou, Zhengxin Zhang, Hao Wu*
School of Information Science and Engineering, Yunnan University
Chenggong Campus, Kunming, P.R. China
{zqmynu,zzxynu}@gmail.com, haowu@ynu.edu.cn

Abstract

This paper describes our method that competed at WASSA2018 *Implicit Emotion Shared Task*. The goal of this task is to classify the emotions of excluded words in tweets into six different classes: sad, joy, disgust, surprise, anger and fear. For this, we examine a BiLSTM architecture with attention mechanism (BiLSTM-Attention) and a LSTM architecture with attention mechanism (LSTM-Attention), and try different dropout rates based on these two models. We then exploit an ensemble of these methods to give the final prediction which improves the model performance significantly compared with the baseline model. The proposed method achieves 7^{th} position out of 30 teams and outperforms the baseline method by 12.5% in terms of *macro F1*.

1 Introduction

Sentiment analysis is a hot and vital research area in the field of natural language processing. It aims at detecting the sentiment expressed in the context written by the authors. Many advanced deep learning models have been exploited to address this issue in recent years (Cambria, 2016; Kim, 2014). The rise of social media, such as twitter and facebook, has fueled the interest of researchers in this field. Twitter is one of the most popular and influential social media all over the world, which attracts over more than 300 million users with over 500 million tweets every day [1]. Therefore, it has received great attention in research communities as a data source due to its easy accessibility of data and diversity of the content (Pak and Paroubek, 2010).

In this shared task, given tweets are incomplete because that certain emotion words are removed from these tweets. These words belong to one of the following classes: sad, happy, disgusted, surprised, anger and afraid, or a synonym of one of them. The goal of the task of WASSA2018 is to classify the emotion of the excluded words into one of the above-mentioned emotions according to the incomplete tweets. All the data given by WASSA2018 are in English.

For this task, we put forward to two different models: one is LSTM-Attention which mainly consists of LSTM (Li and Qian, 2016) and attention mechanism (Bahdanau et al., 2014; Lai et al., 2015), the other is BiLSTM-Attention which mainly consists of BiLSTM and attention mechanism. We have tried different dropout (Srivastava et al., 2014) rates to get different classification results on each model. To further better the predictive performance, our final method employs an ensemble of these models, with a strategy called *soft voting*.

The remainder of the paper is structured as follows: we provide the detailed architecture of proposed methods in Section 2. We present evaluation metrics and experimental results in Section 3. And we conclude our works and point to the future works in Section 4.

2 Methodology

In this section, we describe the details of our proposed methods, including data preprocessing, neural networks and ensemble strategy.

2.1 Data Preprocessing

As data released by WASSA2018 is crawled from the internet, raw tweets may contain a lot of useless (even misleading) information, such as some punctuations and abbreviations. Therefore, we perform a few preprocessing steps to improve the quality of raw data for the ongoing study: (1) The positions in the tweets where the emotion words have been removed are marked with [#TRIG-

[1] http://www.internetlivestats.com/twitter-statistics/

Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, pages 189–194
Brussels, Belgium, October 31, 2018.
https://doi.org/10.18653/v1/P17

GERWORD#] (see Figure 1), so we remove them from the raw data. (2) We remove the useless link "$http://url.removed$" and some meaningless punctuations such as semicolon and colon. (3) We restore some abbreviations in the tweets, e.g., substituting *"have"* for *"'ve"*. (4) All characters are then transformed into lowercase. (5) The TweetTokenizer [2] tool is used to split tweets into a list of words. We try to remove stopwords via nltk.corpus [3], but there is no performance improvement, so we ignore this processing.

```
I was absolutely [#TRIGGERWORD#] when I heard about
what happened yesterday. It astounds me that some
people··· http://url.removed
```

Figure 1: An example of raw tweets

2.2 Neural Networks

Our models consist of an embedding layer, a LSTM or BiLSTM layer, an attention layer and two dense layers. Figure 2 shows the architecture of the BiLSTM-Attention model. For the LSTM-Attention model, it shares the same architecture with the BiLSTM-Attention model, except that the BiLSTM layer is replaced with the LSTM layer.

2.2.1 Embedding Layer

To extract the semantic information of tweets, each tweet is firstly represented as a sequence of word embeddings. Denote s as a tweet with n words and each word is mapping to a global vector (Mikolov et al., 2013), then we have:

$$s = [\vec{e_1} \parallel \vec{e_2} \parallel \vec{e_3} \parallel ... \parallel \vec{e_n}], \quad (1)$$

where vector $\vec{e_i}$ represents the vector of i-th word with a dimension of d. The vectors of word embeddings are concatenated together to maintain the order of words in a tweets. Consequently, it can overcome deficits of *bag-of-words* techniques. For our methods, Word2vec-twitter-model, a pre-trained word embedding model using Word2vec technique (Mikolov et al., 2013) on tweets is exploited. The embedding dimension of Word2vec-twitter-model is d=400.

2.2.2 LSTM/Bidirectional-LSTM Layer

In this emotion classification task, we model the twitter messages using *Recurrent Neural Network* (RNN), to be exact, we respectively examine LSTM and Bidirectional LSTM (Zeng et al., 2016) to process the tweets. LSTM firstly introduced by (Hochreiter and Schmidhuber, 1997) has proven to be stable and powerful for modeling long-time dependencies in various scenarios such as speech recognition and machine translations. Bidirectional LSTM (Graves and Schmidhuber, 2005; Graves et al., 2013) is an extension of traditional LSTM to train two LSTMs on the input sequence. The second LSTM is a reversed copy of the first one, so that we can take full advantage of both past and future input features for a specific time step. We train both LSTM and Bidirectional LSTM networks using back-propagation through time (BPTT) (Chen and Huo, 2016). After the embedding layer, the sequence of word vectors is fed into a single-layer LSTM or Bidirectional LSTM to achieve another representation of $h = LSTM/BiLSTM(s)$. In order to maintain consistency of dimensions, the number of neurons is configured as 400 in both the LSTM Layer and the BiLSTM Layer.

2.2.3 Attention layer

Generally, not all words in a tweet contribute equally to the representation of tweet, so we leverage word attention mechanism to capture the distinguished influence of the words on the emotion of tweet, and then form a dense vector (Yang et al., 2017) considering the weights of different word vectors. Specifically, we have:

$$\begin{aligned} u_{ti} &= \tanh(W h_{ti} + b), \\ \alpha_{ti} &= \frac{\exp(u_{ti}^T u_w)}{\sum_{j=1}^{n} \exp(u_{tj}^T u_w)}, \\ s_t &= \textstyle\sum_i \alpha_{ti} h_{ti}. \end{aligned} \quad (2)$$

t represents t-th tweet, i represents i-th word in the tweet and n is the number of words in a tweet. h_{ti} represents the word annotation of the i-th word in the t-th tweet which fed to a one-layer MLP to get u_{ti} as a hidden representation of h_{ti}. More specifically h_{ti} is the concatenation output of the LSTM/BiLSTM layer in our model. W is a weight matrix of the MLP, and b is a bias vector of the MLP. Then we measure the importance of words through the similarity between u_{ti} and a word level context vector u_w which is randomly initialized. And after that, we get a normalized importance weight α_{ti} through a softmax function. α_{ti} is the weight of the i-th word in the t-th tweet. The bigger α_{ti} is, the more important the i-th word is

[2] http://www.nltk.org/api/nltk.tokenize.html
[3] http://www.nltk.org/api/nltk.corpus.html

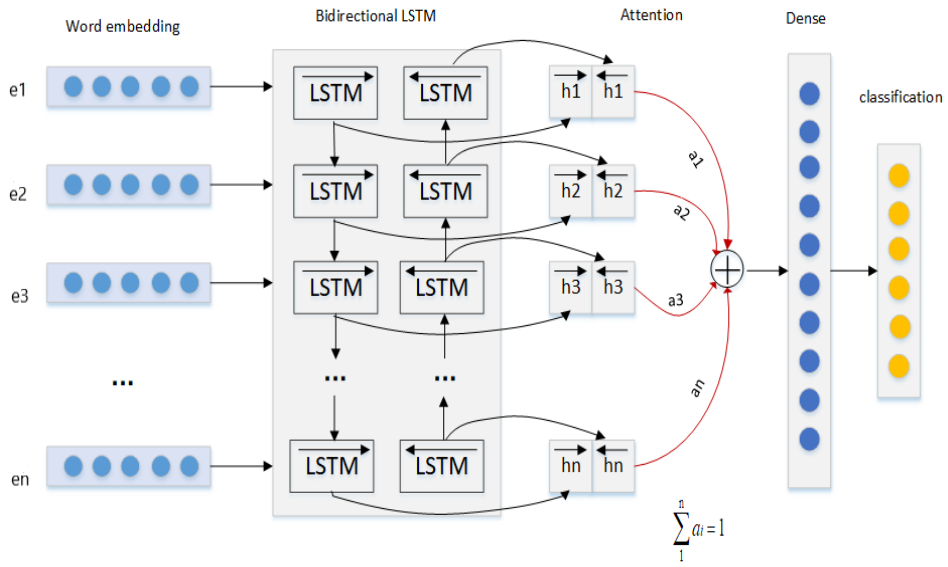

Figure 2: The architecture of BiLSTM-Attention model

for emotion representation. Finally, we represent the sentence vector s_t as a weighted sum of the word annotations.

2.2.4 Dense Layers

The attention layer is followed by two dense layers with different sizes of neurons. The output of attention layer is fed into the first dense layer with 400 hidden neurons. The activation function of this layer is *tanh*. And in order to avoid potential overfitting problem, *dropout* is utilized between these two dense layers. And we try different dropout rates to find the best configurations. The output is then fed into the second dense layer with 6 hidden neurons, and the activation function in this layer is *softmax*. So we can obtain the probability that the excluded word belongs to each of the six classes.

2.3 Ensemble Strategy

Ensemble strategies (Dietterich, 2000) have been widely used in various research fields because of their ascendant performance. Ensemble strategies train multiple learners and then combine them to achieve a better predictive performance. Many ensemble strategies have been proposed, such as Voting, Bagging, Boosting, Blending, etc [4]. In our methods, a simple but efficient ensemble strategy called soft voting is utilized. It means that for a classification problem, soft voting returns the class label of the maximum of the weighted sum of the predicted probabilities. We assign a weight equally to each classifier, then the probability that a sample belongs to a certain class is the weighted sum of probabilities that this sample belongs to this class predicted by all classifiers. And the class with the highest probability is the final classification result. It can be defined as Eq.3 (Zhou, 2012):

$$H^j(x) = \frac{1}{T}\sum_{i=1}^{T} h_i^j(x). \quad (3)$$

i represents i-th classifier, T is the total number of classifier. j is the class label where j is an integer between 0 and 5, because there are 6 classes in our task. x is a sample. $h_i^j(x)$ represents the i-th classifier's predictive probability towards the sample x on the j-th class label, it is a probability which is between 0 and 1. Finally, $H^j(x)$ represents the probability that the sample x belongs to j-th class after ensembling.

3 Experiments

3.1 Evaluation Metrics

To evaluate the classification performance, there are two available metrics: *macro average* and *micro average*. In this task, we use macro average to measure the performance of proposed methods. Macro average is the arithmetic mean of the performance metrics for each class, e.g. precision and recall (Ting, 2011). *Precision* is the fraction of relevant instances among the retrieved instances, while *recall* is the fraction of relevant instances

[4] http://scikit-learn.org/stable/modules/ensemble.html

that have been retrieved over the total amount of relevant instances [5]. More specifically, *macro F1* score is utilized as a measurable indicator of classification performance. The F1 score can be interpreted as a weighted average of the precision and recall. The relative contributions of precision and recall to the F1 score are equal. The formula of F1 score can be defined as Eq.4:

$$F1 = \frac{2 * precision * recall}{precision + recall}. \quad (4)$$

3.2 Experiment Results

Our system is implemented on Keras with a Tensorflow backend [6]. For experiments, we use the datasets downloaded from WASSA2018, they mainly include three splits: 153,383 tweets in the training set, 9,591 tweets in the validation set and 28,757 tweets in the test set. We train our model on the training set, and then tune the hyper parameters of models on the validation set.

For training, the mini-batch size is set at 128 and the max length of sentences (namely, the number of words in a tweet) is configured as 37 to ensure the same length of each tweet. Namely, if the length of a tweet is less than 37, it will be padded with zero; otherwise, it will be truncated from the tail. And the dropout rates that we have tried are ranged from 0.1 to 0.6 with a step of 0.1. In our models, the categorical-crossentropy based loss function and the gradient descent algorithm with *Adaptive Moment Estimation* (Kingma and Ba, 2014) are used to learn the model parameters of neural networks as well as the word vectors. The default parameters of *Adaptive Moment Estimation* is $learning\_rate$=0.001, $beta\_1$=0.9, $beta\_2$=0.999, $eposilon$=1e-08.

The experimental results on different emotion classes are shown in Table 1. There are three evaluation metrics of each emotion class, namely precision, recall and F1 score. Apparently, our system works best on the emotion class named joy in term of all the metrics. And the F1 score on the class called anger is the lowest.

The experimental results of different models in our system are shown in Table 2. Obviously, all of our models outperform the baseline model dramatically. More specifically, the BiLSTM-Attention model performs slightly better than the LSTM-Attention model because BiLSTM can learn more features than LSTM.

[5]https://en.wikipedia.org/wiki/Precision-and-recall
[6]https://keras.io

Classes	Precision	Recall	Macro average F1
sad	0.685	0.622	0.652
joy	0.773	0.778	0.776
disgust	0.701	0.673	0.687
surprise	0.620	0.683	0.650
anger	0.618	0.627	0.623
fear	0.722	0.722	0.722

Table 1: Experimental Results on Different Classes

Model	Dropout	Macro average F1
Baseline	-	0.599
BiLSTM-Attention	0.1	0.659
	0.2	0.662
	0.3	0.655
	0.4	0.659
	0.5	0.658
	0.6	0.659
LSTM-Attention	0.1	0.656
	0.2	0.659
	0.3	0.661
	0.4	0.660
	0.5	0.652
	0.6	0.654
Ensemble	-	0.685

Table 2: Experimental Results on Different Models

To achieve better performance, we utilize a simple ensemble method called *soft voting*. Briefly speaking, if a class gets the highest weighted sum of probabilities from various models, then it is the final class which our sample belongs to. After ensembling the LSTM-Attention model and the BiLSTM-Attention model with different dropout rates, the macro F1 score reaches to 0.685. These results demonstrate that the ensemble approach boosts the classification performance dramatically.

Figure 3 shows the predictive accuracy which is measured by F1 score as a function of epoch when dropout rate is 0.2. The accuracy of our model is optimal when the epoch is equal to 2, so we set the epoch to 2 in our final model.

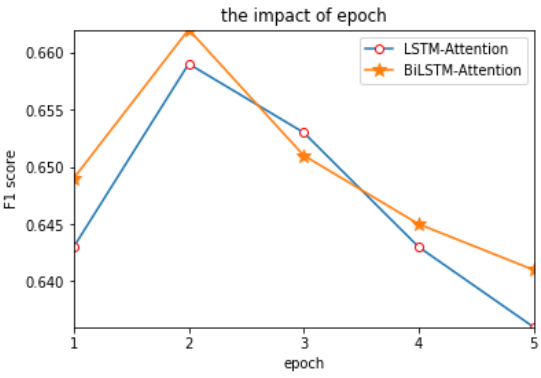

Figure 3: impact of epoch

4 Conclusion & Future work

We have presented a deep learning based approach for implicit emotion analysis task which can be seen as a classification task. We explore LSTM model and BiLSTM model both equipped with attention mechanism using different dropout rates and leverage ensemble method to boost the classification performance. Experimental results demonstrate that our system is effective for this implicit emotion classification task.

As for future works, it can follow three directions. Firstly, we intend to try different ensemble methods like hard voting and stacking to find which one is the most suitable for our task. Secondly, we would like to combine word embedding and char embedding (Santos and Guimaraes, 2015) together with different weights. Also we can utilize some new embedding algorithms like ELMo embeddings [7]. Finally, we plan to explore more textual features like emotion icons to gain better performance.

Acknowledgments

This work is partially supported by the National Natural Science Foundation of China (61562090) and the Graduate Research Innovation Found Projects of Yunnan University (YDY17113).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Erik Cambria. 2016. Affective computing and sentiment analysis. *IEEE Intelligent Systems*, 31(2):102–107.

Kai Chen and Qiang Huo. 2016. Training deep bidirectional lstm acoustic model for lvcsr by a context-sensitive-chunk bptt approach. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 24(7):1185–1193.

Thomas G Dietterich. 2000. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE.

[7] https://allennlp.org/elmo

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Lstm can solve hard long time lag problems. In *Advances in neural information processing systems*, pages 473–479.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

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

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*, volume 333, pages 2267–2273.

Dan Li and Jiang Qian. 2016. Text sentiment analysis based on long short-term memory. In *IEEE International Conference on Computer Communication and the Internet*, pages 471–475.

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

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *LREc*, volume 10, pages 1320–1326.

Cicero Nogueira dos Santos and Victor Guimaraes. 2015. Boosting named entity recognition with neural character embeddings. *arXiv preprint arXiv:1505.05008*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Kai Ming Ting. 2011. Precision and recall. In *Encyclopedia of machine learning*, pages 781–781. Springer.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2017. Hierarchical attention networks for document classification. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Ying Zeng, Honghui Yang, Yansong Feng, Zheng Wang, and Dongyan Zhao. 2016. A convolution bilstm neural network model for chinese event extraction. In *Natural Language Understanding and Intelligent Applications*, pages 275–287. Springer.

Zhi-Hua Zhou. 2012. *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC.