YUN-HPCC at SemEval-2019 Task 3: Multi-Step Ensemble Neural Network for Sentiment Analysis in Textual Conversation

Dawei Li, Jin Wang and **Xuejie Zhang**
School of Information Science and Engineering
Yunnan University
Kunming, P.R. China
`Contact: xjzhang@ynu.edu.cn`

Abstract

This paper describes our approach to the emotion detection of Twitter textual conversations based on deep learning. We analyze the syntax, abbreviations, and informal-writing of Twitter; and perform perfect data preprocessing on the data to convert them to normative text. We apply a multi-step ensemble strategy to solve the problem of extremely unbalanced data in the training set. This is achieved by taking the GloVe and ELMo word vectors as input into a combination model with four different deep neural networks. The experimental results from the development dataset demonstrate that the proposed model exhibits a strong generalization ability. For evaluation on the test dataset, we integrated the results using the stacking ensemble learning approach and achieved competitive results. According to the final official review, the results of our model achieved micro-F_1 score of about 0.7588 on the final evaluation.

1 Introduction

Over the past 10 years, short microblogging communication methods, such as Tweets and Weibo, have been widely adopted. Numerous emotions exist in the dialogue of texts, and there is great commercial value for the detection of such emotions. For example, in the customer service field, the feedback time limit is divided according to the emotion.

Text conversation emotion detection is a challenging issue without facial expressions and mood information being available. Moreover, the data of sadness, anger, and happiness in the current context, as well as the extremely unbalanced scale, natural language ambiguity, and rapidly growing online language, further exacerbate the challenges of sentiment detection.

In this paper, we describe our work on SemEval 2019 Task 3, EmoContext: Contextual Emotion Detection in Text (Chatterjee et al., 2019). The main challenge of the task lies in imbalanced data distribution. Previously proposed methods for solving data imbalance include over-sampling, under-sampling (Weiss, 2004), and Synthetic Minority Oversampling Technique (SMOTE) (Chawla et al., 2002). over-sampling is copying from a smaller number of categories, which may lead to over-fitting. Under-sampling discards potentially useful information, which can degrade the performance of the classifier (Drummond et al., 2003). SMOTE is down-sampling first, and then integration. We propose a multi-step integration approach similar to SMOTE for this task to alleviate the data imbalance problem. In the first step, we use five-fold cross-validation to train five sub-neural networks with different data distributions. Thereafter, soft-voting is used for integrating the results from these sub-neural networks. Soft-voting integration is applied because it not only alleviates the problem of unstable prediction results caused by data imbalance, but also improves the effective prediction accuracy of the single model. The second step of stacking integration further enhances the global effective prediction accuracy. The experimental results indicate that the proposed model alleviates the problem of data imbalance and significantly improves the effective prediction accuracy.

The remainder of this paper is organized as follows. In section 2, we describe the system architecture. Section 3 explains the data processing and parameter tuning. The conclusions and future work are presented in Section 4.

2 System Architecture

The data with the label *others* comprise 49.56% of the training set. The model trained by the conventional method exhibits a poor generalization a-

Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019), pages 360–364
Minneapolis, Minnesota, USA, June 6–7, 2019. ©2019 Association for Computational Linguistics

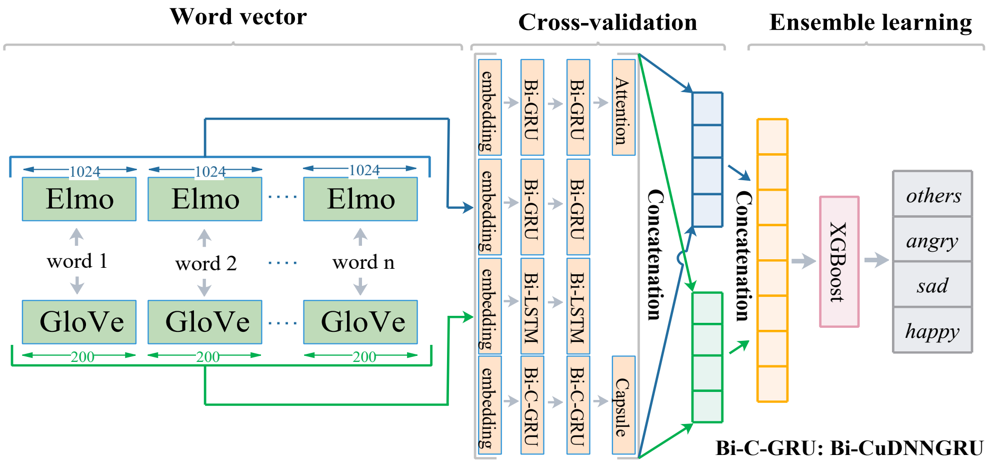

Figure 1: System architecture.

bility and is prone to over-fitting. Sampling verification may alleviate the problem of data imbalance (He and Garcia, 2008). In order to enable the model to learn the data characteristics of small samples, we use five-fold cross-validation to verify the model and test its robustness. The system architecture is illustrated in Figure 1.

2.1 Embedding

We use a GloVe (Pennington et al., 2014) pre-trained word vector: the Twitter 200-dimensional word vector. GloVe is a word representation tool based on the count base and overall statistics. It expresses a word as a vector of real numbers, and captures the semantic properties of words, such as similarity and analogy. Meanwhile, the ELMo algorithm (Peters et al., 2018) is used to train word vectors. ELMo simulates not only the complex features of vocabulary use, but also the changes in these usages in different language contexts. To train the ELMo word vector, we use the aforementioned processed text as input, including both the training and development set. The text is processed into a lookup table, and the words are passed into the ELMo algorithm one by one to generate a 1024-dimensional word vector.

For the feature extraction step, we use Keras (Francois and Chollet, 2015) to convert the text into a vector form of the word embeddings.

2.2 Model

Conventionally, the deep learning model is used in the natural language processing field. We use four superior-performance model components. By combining two different word embedding models, we obtain eight different models. We use four deep learning models, namely LSTM (Hochreiter and Schmidhuber, 1997; Mikolov, 2010), GRU (Cho et al., 2014), Capsule-Net (Sabour et al., 2017; Zhao et al., 2018), and Self-Attention (Luong et al., 2015).

We use Dropout (Salakhutdinov et al., 2014) to aid with improved model convergence. Finally, at each model output, we output the four predicted categories of probabilities, instead of the predicted results.

2.3 Ensemble Learning

According to the dataset characteristics, we design a multi-step ensemble neural network for the four-category emotion detection task.

The first step of integration consists of randomly dividing the entire dataset into 5-folds. In each round, four folds are used for training and the remaining fold is used to validate the model. Moreover, the model is used to predict the development and test sets. We use softmax activation function to get the probability distribution. At the end of the 5-fold cross-validation, five predicted probability values of train set, development set and test set are obtained. We apply a soft-voting mechanism to integrate the prediction probability on five predicted probability of development and test sets. The second step of integration involves combining different word vectors (GloVe and ELMo) with different models. The results of the first step of multiple models are horizontally concatenated as input for the second step of integration. The parameters are tuned and the predicted results are output.

Codes are publicly available at https://github.com/L-Maybe/SemEval-2019-task3-EmoContext

	others	*angry*	*sad*	*happy*
Training	0.4956	0.1826	0.1811	0.1407
Development	0.8486	0.0544	0.0454	0.0515

Table 1: Percentage of categories in dataset.

In the final integration phase, we use the XGBoost (Chen and Guestrin, 2016) toolkit, which utilizes CPU multithreading for parallelism and exhibits strong classification performance. Furthermore, the toolkit can set different weights for unbalanced datasets, which is the most important reason for its use as the final predictive classifier. Following the ensemble learning of the soft-voting in the first step, we obtain eight groups of classification probability values. Owing to the different inputs, the final outputs of the four models differ, which meets the requirements of integrated learning. We horizontally concatenate the eight sets of probability prediction values into a new feature matrix and use the XGBoost tool to learn the new feature matrix to obtain the final prediction result.

3 Experiments and Results

3.1 Datastes and Official Evaluation Metrics

Datasets were provided by SemEval 2019 Task 3, EmoContext: Contextual Emotion Detection in Text (Chatterjee et al., 2019). The participants were asked to predict the emotions of a three-turn conversation. The task considered three emotion classes, namely *happy*, *sad*, and *angry*, along with an *others* category. The number of training and development sets was 30160 and 2755, respectively. The categories of the training and development sets are displayed in Table 1. Owing to the imbalance of the training set data, the official evaluation matrics is the micro-average F_1-score.

3.2 Preprocessing

For the data preprocessing, cleaning, and tokenization, as well as for most of the training sets, we used the Python Scikit-learn (Pedregosa et al., 2013) and Ekphrasis (Baziotis et al., 2017; Gimpel et al., 2011). The data is different from regular text, with substantial amounts of irregular grammar, logograms, and abbreviations, among others. Moreover, the emojis in text have an influence on the emotions detection. We studied the abbreviations of the text, determined the comparison table of abbreviations and words, and added English abbreviations to the abbreviated words. Moreover, the special emojis in text were counted, and a comparison table of expressions and corresponding explanations was generated. The processing steps are as follows:

- Process multiple consecutive punctuation points or emojis in a text into a punctuation or emoji.
- Splice the dialogue into a single text, and segment each round of dialogue with ’<eos>’.
- Use regular expressions to convert English mis-spelled words with similar rules into the correct words (for example, convert ’goooooood’ to ’good’, and ’yesssss’ to ’yes’).
- Use the Ekphrasis tool to segment texts. This tool is used to separate special emoticons (for example, convert ’:-(’ to ’unhappy face’, and ’:)’ to ’smiley face’), which is effective for the next step of expression processing.
- Traverse the word segmentation and comparison table one by one, replacing logograms, abbreviations, and emojis (for example, ’ ☹’ to ’unhappy face’ and convert ’☺’ to ’smiley face’,).

3.3 Parameter Optimization

In order to search the optimal parameters for each model, we used the Scikit-Learn toolkit to perform a grid search on the training set. In the single model tuning phase, we output the probability value of the four classifications and then performed integration. We used the micro-averaged F_1-score to evaluate the results of the soft-voting and to tune the optimal parameters.

After adjusting the single model parameters, we obtain the eight best performing models. As the training set was randomly scrambled, the training set for each cross-validation differs. Therefore, in the final integration, eight models will be run together, and the results of the eight models will be integrated to obtain the predicted results. The classifier for integrated learning is XGBoost, which uses the grid search method in Scikit-Learn to tune the optimal parameters. As the data to be learned are unbalanced, cross-validation is used to adjust the parameters. This step is applied to the training and development sets, and finally predicts the test set results.

Embeding	Dataset	Micro-Average-F_1(%)			
		Attention GRU	**Capsule-Net**	**LSTM**	**GRU**
GloVe	Dev	69.81	70.16	68.76	66.27
	Test	71.26	70.24	69.05	69.43
ELMo	Dev	71.36	70.18	70.32	70.12
	Test	70.04	70.95	70.92	70.1
Ensemble	Dev	76.05			
	Test	75.88			

Table 2: Final submission of development sets and tests.

	Precision	***Recall***	***F_1***
happy	0.717	0.687	0.701
sad	0.802	0.824	0.813
angry	0.720	0.0.819	0.766

Table 3: The result for each emotion class of test dataset.

The parameters are described as follows: model 1 and model 2 are two layers of LSTM and GRU respectively. Then there is the Dropout layer, and finally the softmax function is used to output the probability value. Model 3 and model 4 add the Attention layer and the Capsule-Net layer respectively after the stacked bidirectional GRU. The output is the same as model 1 or 2. The dropout in the GRU component is 0.25. The hidden dimension using the GloVe word vector is 120 while using the ELMO word vector is 400.The optimizer is rmsprop with a learning rate of 0.3. The loss function is categorical cross-entropy. The bacth size of model is 256. The Routings is 5 of Capsule-Net while the number of capsule is 10 and the capsule dimension is 32.

The super parameters of XGBoost are as follows: the learning rate is 0.09, the estimators are 18, the maximum depth is 4, gamma is 1.7, subsample is 0.15, colsample_bytree is 0.75, reg_alpha is 0.01, and seed is 6.

3.4 Results and Analysis

The proposed system is trained on the EmoContext training set. There are eight models, and each single model is trained using five-fold cross-validation combined with a soft-voting method. The stacking integration algorithm is used to output the final predicted results. The results of the development and test sets are presented in Table 2. the result for each emotion class of test dataset in Table 3.

It is worth noting that the results of developing the ELMo word vector on the set are significantly superior to the results of the GloVe word vector. We did not discard the GloVe word vector, and the results are not ineffective. Moreover, the performance will be improved by integrating multiple different models. Based on the results of the test set, the GloVe word vector results will be superior to those of the development set in the final prediction. In the overall comparison of the test set and development set results, the model offers a strong generalization ability.

Following the ensemble learning, our final result is 75.88%. Throughout the experiment, we found that, although we used cross-validation and applied a soft-voting mechanism, the experimental results were not very stable, the main reason for which is that the data were not balanced.

4 Conlusion

Our system won 10^{th} place in SemEval-2019 Task 3, EmoContext: Contextual Emotion Detection in Text. The main challenge for this task was data imbalance. We achieved a competitive result using a multi-step ensemble neural network. The use of cross-validation in a single model mitigates the effects of data imbalance. A soft-voting mechanism was incorporated into the process to improve the model stability further. The results of the eight models were integrated using stacking integration. According to the final official review, our system is certainly effective. In future work, we will study how to enable the model to learn more features and improve our proposed model in the case of data imbalance.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (NSFC) under Grants No.61702443 and No.61762091. We also thank anonymous reviewers for their comments.

References

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. DataStories at SemEval-2017 Task 4: Deep LSTM with Attention for Message-level and Topic-based Sentiment Analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, 1, pages 747–754.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

Nitesh Chawla, Kevin Bowyer, Lawrence O. Hall, and W Philip Kegelmeyer. 2002. SMOTE: Synthetic Minority Over-sampling Technique. *J. Artif. Intell. Res. (JAIR)*, 16:321–357.

Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794.

Kyunghyun Cho, Bart van Merri?nboer, Dzmitry Bahdanau, and Y Bengio. 2014. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. *Computer Science*.

Chris Drummond, Robert C Holte, et al. 2003. C4. 5, class imbalance, and cost sensitivity: why undersampling beats over-sampling. In *Workshop on learning from imbalanced datasets II*, volume 11, pages 1–8. Citeseer.

Francois and Chollet. 2015. Keras: Deep learning library for theano and tensorflow.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-Speech Tagging for Twitter: Annotation, Features, and Experiments. *Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers*.

H. He and E. A. Garcia. 2008. Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412—-1421.

Tomas Mikolov. 2010. Recurrent neural network based language model. *Interspeech*, 2:3.

Fabian Pedregosa, Gal Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, and Vincent Dubourg. 2013. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(10):2825–2830.

Jeffrey Pennington, Richard Socher, and Christoper Manning. 2014. GloVe: Global Vectors for Word Representation. *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532—-1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *The North American Chapter of the Association for Computational Linguistics 2019*, abs/1802.0.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules. pages 3856–3866.

Nitish Srivastava Salakhutdinov, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Gary M Weiss. 2004. Mining with rarity: a unifying framework. *ACM Sigkdd Explorations Newsletter*, 6(1):7–19.

Wei Zhao, Jianbo Ye, Min Yang, Zeyang Lei, Suofei Zhang, and Zhou Zhao. 2018. Investigating Capsule Networks with Dynamic Routing for Text Classification. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3110–3119.