THU-HCSI at SemEval-2019 Task 3: Hierarchical Ensemble Classification of Contextual Emotion

Xihao Liang, Ye Ma, Mingxing Xu
Department of Computer Science & Technology
Tsinghua University, Beijing, China
(liangxh16, ma-y17)@mails.tsinghua.edu.cn, xumx@tsinghua.edu.cn

Abstract

In this paper, we describe our hierarchical ensemble system designed for the SemEval-2019 task3, *EmoContext*. In our framework, we trained three set of classifiers for all-four-emotion classification, Happy-Sad-Angry classification and Others-or-not classification respectively. Through three steps of voting, the predicted labels of these three sets of classifiers are combined to make the final prediction. Experiment results show that the ensemble approach manages to improve the predictive performance in comparison with using only single classifier and our system has achieved the top 10 performance in the final evaluation of *EmoContext*.

1 Introduction

Sentiment analysis is a task to identify the emotion conveyed by written language. With the popularization of the Internet and instant message applications, text has become one of the most familiar media that people use to express their ideas and communicate with each other. Automatic emotion classification can help people and robots better understand the messages or comments written by other people and make proper responses, which makes this study field increasingly important.

Approaches to sentiment analysis can be classified into three categories: rule-based approaches, non-neural machine learning approaches and deep learning approaches. Significantly, neural networks have been used to achieve state-of-art performance in many recent studies. Gupta et al. (2017) used a LSTM-based model to identify the emotion in 3-turn conversations in Twitter but the turn information is not manipulated. For emotion detection on TV show transcripts, a sequence-based convolution neural network which can make use of information in the previous lines was designed by Zahiri and Choi (2017). Hazarika et al. (2018) proposed a conversational memory network based on both CNN and GRU to recognize emotion in dyadic dialogue videos, featuring its ability to manipulate the information of different speakers. Designing a proper structure of neural network to capture the information in different types of text data still remains a valuable topic.

In this paper, we describe our approach to SemEval-2019 task3, *EmoContext*, which aims to encourage more research of contextual emotion detection in textual conversation. A dataset of 3-turn conversations is provided and the participating systems are required to predict the contextual emotion of the last turn of each 3-turn conversation: *Happy*, *Sad*, *Angry* or *Others*. The performance of each system is evaluated by the micro-averaged F1-score for *Happy*, *Sad*, *Angry* on the given Test set. Our system is composed of three sets of CNN-based neural networks trained for all-four-emotion classification, Happy-Angry-Sad classification and Others-or-not classification respectively. Through three steps of voting, the predicted labels of these three sets of classifiers are combined to make the final prediction. Our system has achieved F1-score of 0.7616 in the final evaluation on Test set, which is the top 10 performance out of 165 participating systems.

This paper is organized as follows. In Section 2, we describe the dataset provided by the task organizers. In Section 3, our system and the strategies used in model training are detailed. In Section 4, we present and discuss the evaluation results of our system and its base classifiers. Conclusion is given in Section 5.

2 Data

Task organizers have released three datasets. Each sample in these datasets contains a 3-turn conversation, in which the first turn was written by User

Turn1	Turn2	Turn3	label
You are funny	LOL I know that. :p	:)	happy
Yeah exactly	Like you said, like brother like sister ;)	Not in the least	others

Table 1: Samples in the datasets.

Dataset	Others	Happy	Sad	Angry
Train	14948	4243	5463	5506
Dev	2338	142	125	150
Test	4677	284	250	298

Table 2: Class distribution in each dataset.

1, the second turn is User 2's reply to the first turn and the third turn is User 1's reply to User 2 (the second turn). The emotion label of the third turn is manually annotated for each conversation, which is one of the four emotions: *Happy*, *Sad*, *Angry* and *Others*. Two samples are demonstrated in Table 1 and the class distributions of these three datasets are shown in Table 2. As the Train set and Dev set are both allowed to be used for model training at the phase of final evaluation, we combine these two datasets (hereinafter referred as the training set) to train our classifiers.

3 System Description

3.1 Preprocessing

Through our observation of the 3-turn conversations in the datasets, we realize the writing style resembles that of the tweets and comments in Twitter, featuring emoticons, informal usage of language, spelling errors and so on. Hence, we utilized the tweet processor, *ekphrasis*[1] (Baziotis et al., 2017). The preprocessing steps we used include (1) Twitter-specific tokenization, (2) spell correction, (3) word normalization for numbers and dates, (4) annotation for all-capital words, elongated words and repeated punctuations, (5) conversion of emoticons to emotion labels.

3.2 Word Embeddings

Word embedding is to capture semantic, syntactic and sentiment information of a word or token and represent it by a vector. We use the pretrained embedding model provided by Baziotis et al. (2018) [2] (hereinafter called NtuaW2V) in our system, which contains 300-dimensional vectors for 800 thousands words and tokens trained on English Twitter messages that are also preprocessed by *ekphrasis*. In addition, we tried the pretrained model provided by Mikolov et al. (2013) [3] (hereinafter called GoogleW2V), which containes 300-dimensional vectors for 3 million words and phrases trained on Google News dataset, in order to get an insight into the effect of different embedding models.

Name	Value
std. of Gaussian noise	0.1
kernel size of CNN	5
filter number of CNN	128
# of cells of the first dense layer	32
dropout keep prob.	0.5
l_2	0.2
initial learning rate	0.005
decay of learning rate	0.9

Table 3: Hyper-parameters of the classifiers.

3.3 Classifier

Our system is composed of classifiers with the same architecture (Fig 1). The input to the classifier is a 3-turn conversation, treated as 3 sequences of tokens after the preprocessing step. An embedding layer is used to project the input to 3 sequences of vectors. This layer is initialized by the pretrained word embeddings mentioned in Section 3.2. For tokens which are not covered in the embedding model but occur in no less than two training samples, we randomly initialize the embedding vectors.

Each sequence of vectors is then fed into a 1D-CNN layer and a max-pooling layer respectively to get a vector as the representation of the corresponding turn. Three vectors are concatenated as the feature vector of the 3-turn conversation and it is then fed into a dense layer with rectified linear unit (ReLU) and another dense layer with softmax to get the probability distribution over all predicted classes.

3.4 Hierarchical Ensemble

In order the improve the predictive performance based on individual classifiers, we design three steps of voting to combine their predicted labels

[1] https://github.com/cbaziotis/ekphrasis

[2] https://github.com/cbaziotis/ntua-slp-semeval2018

[3] https://code.google.com/archive/p/word2vec/

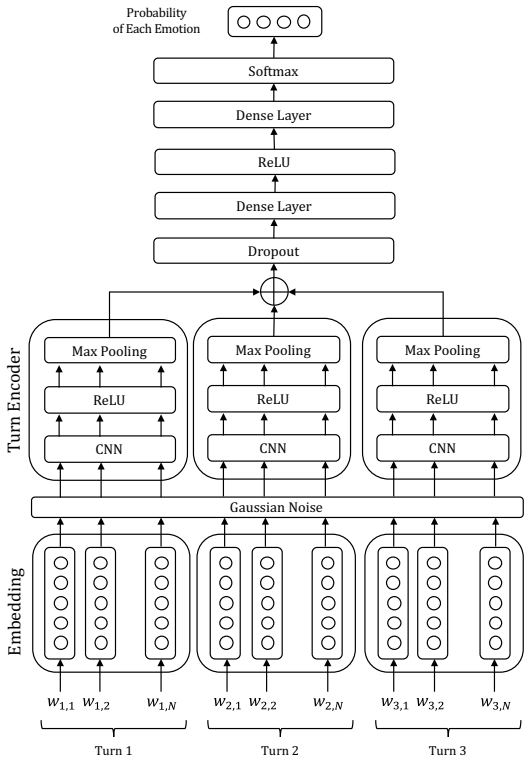

Figure 1: The architecture of CNN-based classifiers in our system.

and make the final prediction. Three sets of classifiers are trained. Set A is a set of all-4-emotion classifier trained by the given dataset and the original labels. Set B is a set of triple classifiers which are trained only by the samples of three classes, i.e. *Happy*, *Sad* and *Angry*. Classifiers in Set C are binary classifiers, which are also trained by the given dataset but the labels for samples of *Happy*, *Sad* and *Angry* are changed to *Not Others*.

First step is the majority voting of the classifiers in Set A to get the base predicted labels (hereinafter referred as Prediction I). At the second step, majority voting is also applied to the classifiers in Set B and we change the label in Prediction I to the major vote if (1) the label in Prediction I is not *Others*, (2) the voting count of the majorly voted class exceeded a given threshold thr_{II}, otherwise the label stay the same as Prediction I. This set of predicted labels are hereinafter referred as Prediction II. At the third step, classifiers in Set C vote for each sample whether its label should be *Others* or not. Then we change the predicted label in Prediction II to *Others* if more than thr_{III} classifiers in Set C vote for *Others*. This set of predicted labels are used as the final prediction (hereinafter referred as Prediction III).

3.5 Regularization

In order to alleviate the problem of overfitting, following strategies are applied.

- Early stopping is used. For each classifier, we randomly select 90% of the training set for training and the 10% left for validation so that the information captured by each base classifier will diverse. Although the evaluation metric is micro-averaged F1 score for *Happy*, *Sad* and *Angry*, we choose the set of neural network weights that achieves the best micro-averaged precision on the validation set. As we notice that precision is significantly sensitive to the class distribution and the task organizers have emphasized the difference between the class distributions of the provides datasets in advance, we want the precision of the base classifiers to be as high as possible.

- New samples of *Others* are automatically created. For each sample of the classs *Others* in the selected 90% training set, we create a new *Others* sample by randomly remove one token in one of the turns to simulate the situation when the user misspell a word and that misspelled token is not known to our embedding models. As we observe that most of the *Others* samples in the training set still belong to *Others* even if one of the tokens is missing or misspelled. However, we do not generate samples for *Happy*, *Sad* and *Angry* automatically in case the discriminative token is removed and a missleading sample will be made.

- Embedding layer is not fine-tuned as the relative positions of the embedding vectors of the tokens in the training set will be changed but those of the tokens in Test set but not in the training set are not adjusted, which may lead to wrong representation of the meaning of these tokens in the new embedding space.

- Gaussian noise is added after the embedding layer, by which the model is more robust when dealing with tokens of similar meaning, as they are supposed to be projected to adjacent embedding vectors.

- Dropout layer is added before the stacked dense layers (Fig 1).

Classifier	Acc	Prec	Rec	F1
Base	**0.9244**	0.7247	0.7661	**0.7445**
no extra Others	0.9206	0.6974	0.7932	0.7420
Early stopping				
F1-score	0.9112	0.6521	0.8093	0.7222
Recall	0.9020	0.6190	**0.8253**	0.7073
Emb				
GoogleW2V	0.9226	**0.7421**	0.7163	0.7286

Table 4: Average performance of our classifiers and the variants on Test set.

- L2 regularization is applied to the weights of the stacked dense layers.

4 Experiments and Discussion

4.1 Implementation Details

Tensorflow (Abadi et al., 2015) is used to develop our models. For network optimization, we choose Adam algorithm (Kingma and Ba, 2014), with minibatches of size 100. Hyper-parameters of the classifiers are shown in Table 3.

4.2 Results and Discussion

Table 4 shows the performance of our base classifiers and the variants in Set A on the Test set. Note that **Base** refers to the models trained as mentioned in Section 3 and **Prec**, **Rec** and **F1** refers to the micro-averaged ones over *Happy*, *Sad* and *Angry*

According to Table 4, automatically generated extra *Others* samples improve both the accuracy and the F1-score, as the classifier intend to predict more samples as *Others* and less *Others* samples will be miss-predicted thus raising the precision at the cost of recall.

We also observe the effect of different metric when used as the indicator for choosing the network weights in early stopping, which is rarely discussed but the results show that the effect on accuracy and the F1-score is significant. It is because of the remarkable difference between the class distributions of the training set and the Test set.

On the other hand, NtuaW2V and GoogleW2V, the embedding models used to initialize the embedding layer, are compared. Results show that using NtuaW2V can achieve better F1-score while using GoogleW2V achieve the best precision but also the worst recall among the variants, which shows the importance of choosing pretrained embedding models. The data on which the embedding model is trained and how the data are preprocessed should be the crucial keys to it.

		Pred			
		Others	**Happy**	**Sad**	**Angry**
Gold	**Others**	4423	74	70	110
	Happy	77	200	5	2
	Sad	29	1	207	13
	Angry	40	1	2	255

Table 5: Sample of confusion matrix on Test set.

Prediction	Acc	Prec	Rec	F1
(Base)	0.9244	0.7247	0.7661	0.7445
I	0.9278	0.7360	0.7740	0.7545
II	0.9281	0.7383	**0.7764**	0.7569
III	**0.9305**	**0.7553**	0.7680	**0.7616**

Table 6: Performance of our system after each step of voting on Test set.

We have analyzed the base classifier's confusion matrix on Test set (Table 5) and two phenomenons are observed. First, the crucial problem that affect the performance of our classifiers is that many *Others* samples are falsely predicted, which severely lowers the precision as the proportion of *Others* are remarkably high in Test set. The second phenomenon is that our classifiers are relatively effective to classify samples among *Happy*, *Sad* and *Angry*.

Table 6 illustrates the performance of our system after each step of voting. Results show that the first two steps bring slight improvement on all four metrics and the final step improves F1-score by raising the precision at the cost of recall.

5 Conclusion and Future Work

In this paper, we present our system used for SemEval2019 Task3, *EmoContext*. This system is composed of three sets of CNN-based neural network models trained for all-four-emotion classification, Happy-Sad-Angry classification and Others-or-not classification respectively. Three steps of voting are used to combine the predicted labels of the base classifiers and make the final prediction. Experiment results show that our training strategies are effective and the ensemble approach manages to improve the performance based on the base classifiers. We also notice the importance of picking the right metric as indicator for choosing network weights in early stopping. In order to achieve better result based on our system, improving the performance of base classifier is crucial and the structures of neural networks and the features used as the input should be the fields that worth further exploration.

References

Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, G.s Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, and Xiaoqiang Zheng. 2015. Tensorflow : Large-scale machine learning on heterogeneous distributed systems.

Christos Baziotis, Nikos Athanasiou, Pinelopi Papalampidi, Athanasia Kolovou, Georgios Paraskevopoulos, Nikolaos Ellinas, and Alexandros Potamianos. 2018. Ntua-slp at semeval-2018 task 3: Tracking ironic tweets using ensembles of word and character level attentive rnns.

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. pages 747–754.

Umang Gupta, Ankush Chatterjee, Radhakrishnan Srikanth, and Puneet Agrawal. 2017. A sentiment-and-semantics-based approach for emotion detection in textual conversations.

Devamanyu Hazarika, Soujanya Poria, Amir Zadeh, Erik Cambria, Louis-Philippe Morency, and Roger Zimmermann. 2018. Conversational memory network for emotion recognition in dyadic dialogue videos. pages 2122–2132.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Tomas Mikolov, Ilya Sutskever, Chen Kai, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 26:3111–3119.

Sayyed M. Zahiri and Jinho D. Choi. 2017. Emotion detection on tv show transcripts with sequence-based convolutional neural networks.