EmotionX-Antenna: An Emotion Detector with Residual GRU and Text CNN

Zhuoran Yu[*] , **Yuhong Wang** , **Zhanhao Liu** and **Xiang Cheng**
Georgia Institute of Technology
{zhuoranyu, yuhong.wang, zhanhao.liu, cxworks}@gatech.edu

Abstract

In this paper, we present an emotion detector built with residual GRU and text CNN, which predicts emotion associated with each utterance within dialogues. The model is used for the competition of emotionX 2019.
Deep learning models have achieved wide success in many natural language processing tasks such as part-of-speech tagging and question answering in recent years. However, with the increase in performance of NLP tasks, the model complexity has also increased dramatically, which usually requires a large amount of training time and GPU usage. Many recent models, such as BERT are impossible to train from scratch on the local GPU. To discover how well small networks can do on emotion detection task, we show in this paper that our model with a rather simpler structure which makes it easy to train can still give a reasonable performance on this emotion detection task. Code is available at: https://github.com/ZhuoranYu/emotionX2019

1 Introduction

Detecting emotion in dialogues is one of the most challenging tasks in the area of natural language processing, which is also important for building dialogue systems[Chen *et al.*, 2018]. A similar task called sentiment analysis had been studied in natural language community for years. Unlike sentiment analysis for regular documents, detecting emotion in dialogues is relatively harder due to the following reasons. First, different people may feel differently in response to the same text, especially for online chatting or texting. Second, people tend to use emoji and abbreviation when texting to each other and use idioms for face-to-face conversation. Moreover, information such as facial expressions, body figures, and tones are lost when dialogues are translated into plain text. With these challenges, emotion detection in dialogues does not make too much progress in the current stage of research.
Emotion detection is a subproblem of text classification. With the boosting effect of deep learning models, various models have also been used for text classification such as Convolutional Neural Networks(CNN)[Kim, 2014] and different variations of Recurrent Neural Networks(RNN)[Lai *et al.*, 2015]. In recent years, more complex models such as Transformers[Vaswani *et al.*, 2017] and BERT[Devlin *et al.*, 2018], which outperforms many previous frameworks in natural language processing tasks at the time of publishing. Although these models have excellent performance in many tasks, the model complexity has also increased dramatically. In this paper, we present an emotion detector with a rather simple structure which is built by a combination of textCNN[Kim, 2014] and bidirectional Gated Recurrent Unit(GRU)[Cho *et al.*, 2014] for the EmotionX 2019 competition. These two component are used for different purposes: we use textCNN to capture local information within each utterance and use GRU to capture relationships between sentences. Furthermore, since the majority utterances in the corpus are annotated as "neutral", the "neutral" class would dominate the loss, which makes training phase difficult. To tackle this issue, we use focal loss[Lin *et al.*, 2017], which is originally proposed in the community of computer vision to reduce the effect of background class in object detection. We would show in the experiment section that with this rather simple architecture, our model can still give a fair performance in the evaluation phase.
The rest of paper is organized as follows: Section 2 we describe data preprocessing steps of datasets associated with the competition. In section 3, we present an overview of our model and how each component is combined. In section 4, we describe the strategy we used to tackle the class imbalance issue. Section 5 presents the evaluation result of our model.

2 Preprocessing

To compute with text, a common practice is to use vectors to represent each word. Pretrained embeddings such as Glove[Pennington *et al.*, 2014] are widely used in the past. Since BERT[Devlin *et al.*, 2018] has shown its power on many NLP tasks, we use BERT to extract vector features for our corpus.
We take each dialogue as an individual document by expanding utterances in order. To make the model encode more contextual meanings on our corpus, we first do a fine-tuning on BERT-Base, which has 12 layers and 12 heads with hidden dimension 768. We then extract vector features for each ut-

[*]Contact Author

terance in dialogues by taking the last layer of the model, which gives us a sentence vector indicated by *[CLS]* tag and word vectors for each word in the utterance.
The open source code from BERT has its tokenizer, and we did not include any additional preprocessing before passing the data into the tokenizer.
With the help of BERT, for each dialogue, we can get a sequence of sentence vectors and sequence of word matrices whose rows represent word vector of words in utterances. Sentence vectors and word vectors are processed separately by the model that is shown in the next section.

3 Model Overview

After preprocessing, each dialogue is converted to a 2D matrix(sentMatrix) whose dimension is (number of utterances, embedding dimension) and a 3D tensor(wordTensor) whose dimension is (number of utterances, number of words, embedding dimension). For the sake of alignment, we only consider the first 15 words in each utterance and do a zero padding if the total number of words in an utterance is less than 15.
Then, wordTensor is passed to TextCNN first, and the output of TextCNN passed together with sentMatrix to the bidirectional residual GRU. Finally, the output of bi-residual GRU is fed into a fully connected layer, and predictions are made correspondingly. Figure 1 shows the general architecture of our model. As illustrated by the figure, our model has a pretty clean and simple structure which can be easily fed into local GPUs. We now go through each component in details.

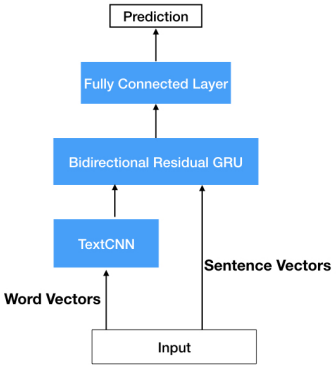

Figure 1: An Overview of Model Architecture.

3.1 TextCNN

We include TextCNN[Kim, 2014] to process word vectors. From our preprocessing step, each utterance is converted to a 15 x 768 matrix where 15 is the max utterance length, and 768 is the embedding dimension. A straightforward demonstration of our variation of TextCNN is presented in Figure 2. To capture local dependencies between words, we include three types of n-gram convolution filters, namely, 2-gram, 3-gram, and 4-gram. For each filter type, we include

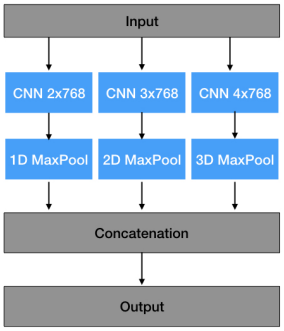

Figure 2: Our variation of TextCNN.

256 filters to increase diversity. The output from convolution layers is passed to a 1D Max-Pooling layer. Finally, the output from pooling layers is flattened and concatenated together as the final output. To avoid overfitting, a dropout layer with a rate of 0.4 is included before the output layer.

3.2 Bidirectional Residual Block

The output from TextCNN is concatenated together with sentMatrix and fed into our main model: bidirectional residual Block.
To capture the relationship between sentences, we decided to use Gated Recurrent Unit[Cho *et al.*, 2014] as our core part. There are also some other choices for top-tier layers such as Long Short-Term Memory(LSTM)[Hochreiter and Schmidhuber, 1997] and self-attentive layers, which are also widely used in many NLP tasks. We do not use self-attentive layers because we believe the emotion of an utterance would only be related to other utterances which are close to the current one. The reason why we choose GRU over LSTM is that the size of our corpus is not large and GRU has a simpler structure so that it converges quicker than LSTM in our scenario.
ResNet[He *et al.*, 2016], which is one of the state-of-the-art models in computer vision, has outperformed many models in computer vision such as VggNet, AlexNet, etc. ResNet exposes its novelty by adding a residual connection between the output of a convolution layer and input. The insight is that when neural networks get deeper, the performance should be at least as good as shallow networks. Inspired by ResNet, we decided to add a residual connection between input and output of the GRU layer. Therefore, the output of each GRU layer can be represented as follows:

$$output = ReLU(GRU(x)) + x \tag{1}$$

Further expand equation (1):

$$\overleftarrow{h_t} = \overleftarrow{GRU}(x_t, \overleftarrow{h_{t-1}}) + x_t \tag{2}$$

$$\overrightarrow{h_t} = \overrightarrow{GRU}(x_t, \overrightarrow{h_{t-1}}) + x_t \tag{3}$$

$$\tag{4}$$

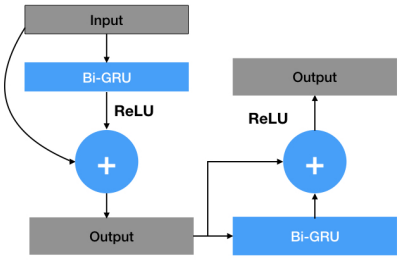

Figure 3: An Illustration of Connection Between Residual GRU Blocks.

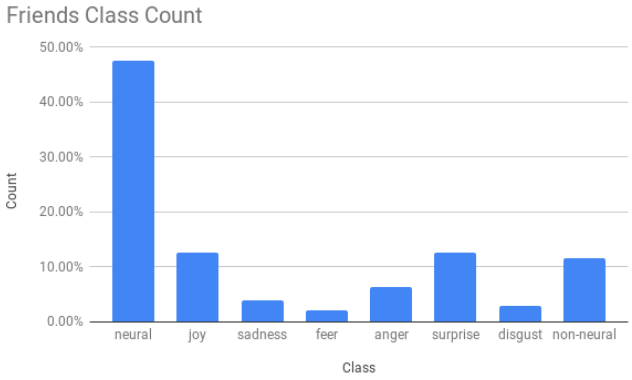

Figure 4: Class Count for Friends

We call one GRU layer with a residual connection a residual GRU block. To increase the performance of our model, more than one GRU block can be stacked together as shown in Figure 3. Our final model includes 3 residual GRU blcoks. To avoid overfitting, we include dropout layers with rate of 0.3 between adjacent residual GRU blocks.

4 Class Imbalance

Our corpus has severely imbalanced classes. As illustrated in Figure 4 and Figure 5, both Friends and EmotionPush are dominated by "neutral", and all other classes have a low count compared to "neutral". Imbalance issue is worse in EmotionPush, entries in 3 classes only below 5%.

To tackle the imbalance issue of classes, we use FocalLoss[Lin *et al.*, 2017] in replacement of regular Cross Entropy Loss for our training phase. Focal Loss is initially introduced to tackle the imbalance between background and foreground in object detection, which boosts the performance of one-state detection algorithms. The original form of Focal Loss in binary classification is given as follows:

$$L = \begin{cases} -\alpha(1-p)^{\gamma} log(p) & \text{if } y = 1 \\ -(1-\alpha)p^{\gamma} log(1-p) & \text{if } y = 0 \end{cases}$$

where p is the confidence score of each class. γ can be viewed

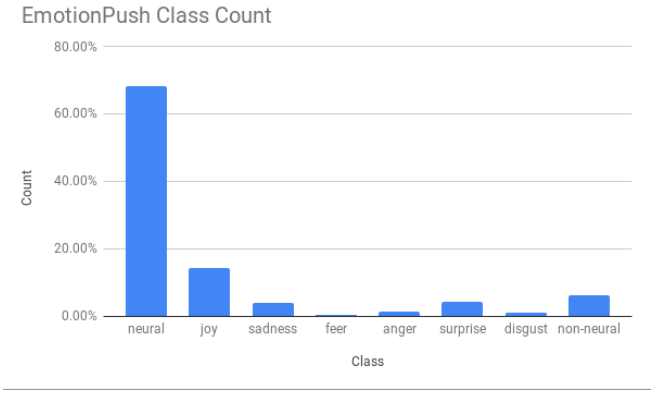

Figure 5: Class Count for EmotionPush

as a hyper parameter which controls the effect confidence score. As we can see from the equation, γ actually controls the effect of $1 - p$, which measures how far the prediction of the class from the truth. When γ is set to 1, FocalLoss reduces to regular Cross Entropy Loss. α is a weight factor works in the same manner as weights in weighted Cross Entropy Loss. We use a multi-class version of focal loss as follows:

$$FL(p_t) = -\alpha_t(1-p_t)^{\gamma} log(p_t) \quad (5)$$

where p_t stands for confidence score of class t.
The output of our model is passed through a softmax layer to get the final scores. Then, Focal Loss is computed and backpropagated as normal.

5 Evaluation

We present our evaluation results and our analysis in this section. Our simple model still gives some reasonable results in evaluation phase.

5.1 Evaluation Results

Table 1 shows the evaluation results of our model. As we can see from the table, our model performs reasonably well on Friends and it does not have too much bias of one class over others. Precisions for Neutral, Joy, and Anger are around 70%, which shows the power of our simple network.
In contrast, for EmotionPush, our model does not perform as good as that for Friends. Only prediction results of Neutral are reasonably good and prediction results of all other classes are not as good as Friends. Predictions for Sadness and Anger are extremely terrible.

5.2 Analysis

Our analysis would focus on the terrible performance on EmotionPush. The first reason we come up with is that quality of text. Since EmotionPush is composed by text messages, there are a large amount of emoji and abbreviations. Since we do not include any additional preprocessing on raw text, these uncommon words and symbols would hurt our model. Another reason can be hyper-parameter tuning. We use the same model structure for Friends and EmotionPush, however, due

Dataset	Unweighted Avg	Metric	Neural	Joy	Sadness	Anger
Friends	68.4	Precision	77.6	77.3	49.2	69.5
		Recall	90.4	58.0	49.6	46.8
		F1	83.5	66.3	49.4	55.9
EmotionPush	35.6	Precision	77.5	42.5	16.2	6.1
		Recall	81.4	20.8	20.0	7.4
		F1	82.2	27.9	17.9	6.7

Table 1: Evaluation Results

to the limit of time, the hyper-parameter tuning is only performed for Friends and we use the same set of parameters to train our model for EmotionPush.

6 Conclusion and Future Work

In this work, we present an emotion detector built with TextCNN and Bidirectional Residual GRU. Although the performance of our model is not perfect, it still shows its potential for emotion detection tasks. It is clear that there is still some possible improvement that can be made on this framework. In the future, we would like to study the following aspects of this task:

1. Depth of residual GRU
2. Data Augmentation

We do not make the depth of our model extremely large because the point of this work is to show the big potential of small models. However, one of the reasons why the residual connection is introduced is that deeper models should at least be as good as shallow models. Therefore, it is interesting to see whether increasing depth could improve performance on emotion detection task. Another aspect is data augmentation, which is already widely used in computer vision but underdiscovered in natural language processing. Our training process does not include any data augmentation step provided by the competition. Data augmentation itself is still not well-studied in the area of natural language processing because of the level of difficulty. Unlike images, augmentation of documents is not as trivial as flipping images. However, we are still curious about how could current data augmentation strategy in natural language processing help us improve our model performance.

References

[Chen *et al.*, 2018] Sheng-Yeh Chen, Chao-Chun Hsu, Chuan-Chun Kuo, Lun-Wei Ku, et al. Emotionlines: An emotion corpus of multi-party conversations. *arXiv preprint arXiv:1802.08379*, 2018.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[Devlin *et al.*, 2018] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[Kim, 2014] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[Lai *et al.*, 2015] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[Lin *et al.*, 2017] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[Pennington *et al.*, 2014] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.