NLP team porject - emotional classification

Bing-Zhi, Ke
Student ID : 0716027

Chen, Yeh
Student ID : 0716081

Tzu-Han, Lo
Student ID : 0716083

1 Introduction

The main goal of this project is to predict a type of emotion through a conversation. The conversion is divided into two parts: prompt and utterance. Prompt represents the summary or the key point of the conversation. Utterance represents the the words that the speaker or the listener said.

Our goal is to decide the emotion of the conversation out of all 32 emotions such as joyful, sad, terrified, disgusted... etc.

2 Related Work

Natural language processing is a studied filed with lots of mature methods. The methods can help the computer understand the meaning of sentences and quantify human's language to machine language. We take the advantage of these methods to do the emotional classification.

2.1 TF-IDF

TF-IDF is the recommended method in simple baseline. Based on statistics, it scores a value representing the relevance between the word and the article. It is composed of term frequency tf and inverse-document frequency IDF. The former one means the frequency of the word x in the article y. The higher $tf_{x,y}$ is, the importance the word x is in the article y. In contract, the latter one means the frequency of the word x in different articles. It first calculates df_x which is how many articles contains x, and then divides the number of total articles. Here comes the function:

$$W_{x,y} = tf_{x,y} \times \log(\frac{N}{df_x})$$

After we get the scores of all words in each articles, we can construct a sparse matrix and pass it to the following classifier.

2.2 Word2Vec & GloVe

Here comes the recommended method in medium baseline. Both of the methods convert words into a vectors. The cosine distance of the vectors represent the similarity of the words. The smaller the distance is, the closer the two words are. The difference between the two methods is that Word2Vec is a predictive model and GloVe is a counted-based model. The former one calculates the cross-entropy of the transformed vector as the loss which controls the model updating. The latter one needs first build a co-occurrence matrix, and takes the use of the matrix to calculate the loss by least-square method.

2.3 BERT

The hard baseline we use is BERT which is a widely used method in NLP field. With an additional output layer to learn, the pre-trained deep bidirectional model can reach state-of-the-art results in many tasks such as question answering and Multi-Genre Natural Language Inference. Therefore, it is a good choice to start with BERT on the NLP task, which can help users roughly know the performances of an unknown dataset.

BERT takes the advantage of masked language model(MLM) and bi-directional transformer. Before training, we need to generate three embedding from the input sentence. They are respectively token embedding, segment embedding, and positional embedding. Then, we can pass the three embedding into BERT model. The BERT model randomly hides about 15% tokens in training process, and use the context to predict the hidden token. It helps a lot on pretraining the model and keeping the features of the contexts.

2.4 CNN/RNN-based model

Convolutional neural network (CNN) do a great improvement on the visual recognition field. It

can also be used in the natural language processing. The procedure of CNN contains generating features from convolution layer, reducing the noise by pooling layer, and flattening the network result with a fully connected layer.

Recurrent neural network (RNN) overcomes the shortage of CNN-based model. It can keep the features in the passed time stamps by its hidden state, which is the also the concept of all RNN-based models. In NLP, sentences is somehow a sequential data, and the words in sentence are contextual. We human can even predict what other is going to say by its last word. Therefore, the RNN-based models are widely used in this field.

3 Data

The data are stored as csv files, and there are five columns in each csv file such as conv_id, utterance_idx, prompt, utterance, and label.

Conv_id indicates the index of each conversation. Utterance_id indicates the index of each utterance in one conversation. Prompt records the scenario or the summary of the conversation. Utterance records the words that the speaker or the listener said. Finally, the label column records the emotion of the conversation.What has to be mentioned is that a prompt may consist of several utterances, but all utterances of the same prompt only correspond to one emotion.

There are 84196 rows in training dataset, and 12078 rows in validation dataset, and 10973 rows in testing dataset.

4 Method

This section introduces the method we used to solve the problem,including the process of implementation. This section is divided into two parts. The first part is about TF-IDF, Word2Vec, GloVe, and BERT. The second part is talking about all details of improvement of BERT.

4.1 Model Testing

There are three kinds of baselines with some supported models in this project. We wonder how strong the models are. We chose the model that we learned in this semester, which are TF-IDF, Word2Vec, GloVe, BERT, and combined them with the NN model as the classifier after we convert words to needed features. It means that in our opinion, the model except the NN models are more likely a kind of feature extractor. And our target is to find the upper-bound of the accuracy of these feature extractor.

Due to our purpose in this part is find out the best model, but not the best accuracy. We only use the prompt column as our input data. Doing simple testing on filtering the prompt, such as stop word removing, punctuation removing and stemming and adjusting the parameter of the models by the recommend steps. The accurate adjustment will be completed in the next section.

4.1.1 TF-IDF

When testing on TF-IDF, we first put all the prompts into a list, and then feed the list into TF-IDF Vectorizer. The vectorizer will return a sparse matrix that includes all the score of each word in the word bag for each prompt. Then we feed this sparse matrix into the classifier we want to use, such as random forest or CNN.

4.1.2 Word2Vec

There are two methods for Word2Vec testing, respectively CBOW and skip-gram. The former uses contextual words to estimate the center word. The slide winder helps to update the evaluation vector of contextual words in the same gradient. On the contrast, skip-gram uses the center word to estimate the context. It is more useful on evaluating the low-frequency words due to the multi-gradient. And, it is also the method we tried to apply on Word2Vec model.

4.1.3 GloVe

For GloVe, we use the pre-trained GloVe embeddings to represent the words. There are several types of GloVe embeddings such as 50-dimensional, 100-dimensional, and 200-dimensional. Each type of GloVe embedding uses different number of dimension to represent a word. The more dimension the embedding use, the more information the embedding include. However, more dimension also means more need of memory and training time.Considering our hardware and and efficiency, we chose 100-dimensional GloVe, i.e. use a 100-dimensional vector to represent a word.

Here we only use prompt as our input feature. Using the word-to-vector GloVe dictionary, we can transform every word in prompt to a 100-dimensional vector. However, the length of every prompt are different, and the input to the model should be the same length, so we use the technique

of padding to let every feature to become the same length. We first find the maximum length among all prompts, and then pad 100-dimensional zero vectors to other prompts in order to satisfy the maximum length.

Finally, after embedding and padding, we can feed the (100*maxlen)-dimensional features to the CNN to complete our task, that is emotional classification.

4.1.4 BERT

For BERT, we directly modified the code from the fourth hand-on project. Without changing the last dense layer, which is viewed as the classifier. The prediction will be index which has the highest value from the output of the last layer. In our expectation, BERT must be the best model of all. Due to the well pre-trained parameter in "BERT-english-uncase" package.

4.2 BERT Improving

Now we are going to discuss about the experiments we did for BERT in order to improve the performance. In the experiments, we have focused on three main elements, which are input data types, stop words, and the voting mechanism. These main elements are decided after our discussions.

The first element is input data type. In our first attempt, we used only prompts as input features. However, using only prompts didn't give us a satisfying result. Therefore, we tried to concatenate each utterance and prompt, and using all of them. We also tried to only concatenate some utterances and prompt.

The second element is about stop words removal. Stop words removal means that the words that seems not helpful for the task will be dropped out from the dataset. We found that the words with negative meanings, such as not, can't... are included in the stop words. However, these words are important for us and the classifier to understand the true meaning of the sentences. So we collected these words and build a white list based on these words, which means the words in the white list should be kept in the sentence while doing stop words removal.

The last element is the voting mechanism. We found that the classifier didn't give each utterance in the same dialogue the same label. So we checked the labels of each utterance in the same dialogue, and changed their labels into the same one by majority vote, which means that the label of the dialogue is determined by the label that most of the utterances have.

5 Experiments

5.1 Model Testing Result

In this section, we will show the results of the experiments using the metrics TF-IDF, Word2Vec, GloVe, and BERT with different classifiers in Table 1. All the models have been adjusted to a set of well parameters which cannot be improved for a short time. It can be said that the results is somehow the upper-bound of the input data.

As the results shown in the table, models using random forest classifier is worse than using CNN/RNN-based model. It means that CNN/RNN-based model can really catch more potential features from the word embedding model. Furthermore, using BERT is much better than other metrics even if the classifier is a simple linear layer. We think that it is because the task is a non-context-free task. But other metrics except BERT are often used for context-free tasks. With the result showing that BERT is the most outstanding model, we try to improve BERT in our next step.

Metric	Classifier	Accu.
TF-IDF	Random Forest	0.3662
TF-IDF	CNN	0.3951
Word2Vec	Random Forest	0.3488
Word2Vec	LSTM	0.3974
Word2Vec	GRU	0.4090
GloVe	CNN	0.4672
BERT	linear layer	0.5323

Table 1: The result of Model testing with the random choosing combination.

5.2 BERT Improving

In this section, we will show the results of the BERT experiments by combining different elements introduced above in Table 2.

In the table, we can see that doing stop words removal got worse results than not doing it, and it is the most important element that affects the accuracy the most.

We think that it is because BERT was pre-trained on huge raw datasets, and read the data with the input order, which is very important for BERT. This helps it to understand the meaning of the input sentences. Also, as we mentioned above, there are

some "white list" that will affect the meaning of the sentences if they are removed. Such as the negative words and prepositions.

Due to the above two reasons, if we do stop words removal, but remove the words that are actually important for BERT, BERT will not able to read the input with the original order. This makes BERT hard to understand the meanings. As a result, we think that doing stop words removal is not helpful for BERT. However, if we can find out the proper stop words and remove them, it might be useful for BERT.

I.D.T.	S.W.	Voting	Accu.
P	no	no	0.5323
$P + V$	yes (with W.L.)	no	0.5376
$P + V$	yes (no W.L.)	no	0.5049
$P + U$	no	no	0.5571
$P + U$	no	yes	0.5636

Table 2: Different improvement on BERT and its corresponding accuracy.
(I.D.T.: Input Data Type, S.W.: Stop Words, P: all of prompts, U: all of utterances, $V \subseteq U$, W.L.: white list)

6 Conclusion

Our conclusion is divided into two parts: conclusion and future works.

6.1 Conclusion

6.1.1 Performances

First, we found that the result of Bert is better than other three methods, that is TF-IDF, GloVe, and Word2Vec. The differences may caused by the methods of embedding.
For instance, TF-IDF consider the frequency of words to be the features. Instead, GloVe and Word2Vec get the embedding through training with other words. Finally, Bert takes in three different vectors to represent a sentence, which is also generated by training. So we think that the reason why Bert can work the best is that the embedding consist of the most information which helps the prediction.

6.1.2 Necessity of preprocessing

Secondly, we think that the texts in Bert need no preprocessing such as stemming or removing stop words. Speaking of stemming, we may regard it of a method to simplify the complexity of the sentence. However, what we have to be aware of is that Bert is a pre-trained model, so if we input the words that is not in the original training dataset, some unexpected error may occur. For instance, the words "I remember going" become "I rememb go" after stemming. Nevertheless, the word "rememb" made the mistake. If we call the function that transforms the tokens to ids, and then transform the ids to tokens back, we found that the word became "re me mb" three words, which is not we expected.

As for removing stop words, we found that the accuracy we got before removing stop words is higher than after removing stop words. We suppose that removing stop words result in the removal of some useful information, so we think it is not necessary for the project.

6.1.3 Length of utterance

Third, we found that the length of utterance has little to do with the accuracy. In the experiments, we concatenated the prompt and one utterance as our input feature. Moreover, we set a hyperparameter called "maxlen" limiting the max length of the feature. If the length of the feature exceeds the limit, the rest of the feature will be truncated. To our surprise, we found that there is little difference between the results with different max lengths. For instance, the difference of accuracy between maxlen = 50, maxlen = 70, maxlen = 100 is less than 0.1

6.2 Future Works

6.2.1 Model improvement

First, we can try other stronger model such as BERT large or RoBERTa as we only BERT for this project. The differences between BERT and RoBERTa include longer training time, bigger batches, training on longer sequence, bigger training dataset, and dynamically generating mask of MLM. As the name of RoBERTa (Robustly optimized BERT approach) indicated, we are convinced that through these adjustments, the performance of RoBERTa could be better than BERT.

6.2.2 Utterance grouping

Secondly, we may try approaches to group our utterances. In our experiments, we only tried concatenation prompt and one utterance as one input feature. After discussion, we assumed that only one utterance provide too few information to be classified. For instance, there are some meaningless utterance such as "Hi!" or "Hello!", which

didn't help the classification at all. Perhaps we can try to concatenate prompt and two or even all utterances as one input feature in order to provide more information for classification.

6.2.3 Hardware

Last, we are also looking forward the upgrade of hardware such as GPU. Better GPU means higher calculation speed and bigger memory space, which allow us to train bigger, stronger model.

7 Work Division

Ke: Word2Vec, BERT, report writing(2, 4.1, 5.1, 8)
Lo: TF-IDF, BERT, report writing (4.2, 5.2, 7, 8)
Yeh: GloVe, BERT, report writing (1, 3, 6, 9)

8 Question and Answer

Q1. *How did you decide the parameters of the models at first?*

A1. For Word2Vec, we adjusted the parameters by the importance given by the official documents. For TF-IDF and BERT, we just used the default parameters of the models.

Q2. *How much accuracy does the voting mechanism increase in your experiments?*

A2. The voting accuracy has increased the accuracy for about 1%. Even though that the 1% improvement might be the bias of our project, we think that the idea is valuable. If the model can kick the useless sentences out before training such as "Bye.", "I think so.", as well as "Well...". The voting mechanism might be more reliable.

Q3. *Which models did you use to try the voting mechanism?*

A3. We only tried the voting idea in BERT and there is no complex steps in our voting mechanism. Actually, the voting mechanism is just to consider all the predictions from the same conversation, and choose the most frequently predicted class as our final prediction of the conversation.

Q4. *Did you do grid search?*

A4. No, we didn't do grid search. No matter which model it is, we can find a conventional procedure to adjust the arguments. Furthermore, our purpose is preliminary testing whether the models can perform as well as the project spec said in the first. We do not need to reach the highest accuracy of all models, but the good enough and stable one.

9 References

- Introduction to TF-IDF
- Methods of adjusting parameters of Word2Vec
- Packages realted to NLP fields
- Influences of stop words removal
- GloVe embeddings