Lab 2 Report

113033628 邱一恩

I. Introduction

In this Lab, we are working on a dataset containing lots of Twitter message text, and our goal is to predict the emotions in the document. There are eight types of label emotion categories in total: 'anticipation', 'sadness', 'fear', 'joy', 'anger', 'trust', 'disgust', 'surprise'. We first checked the data distribution, as shown in Figure 1. which shows a large degree of imbalance.

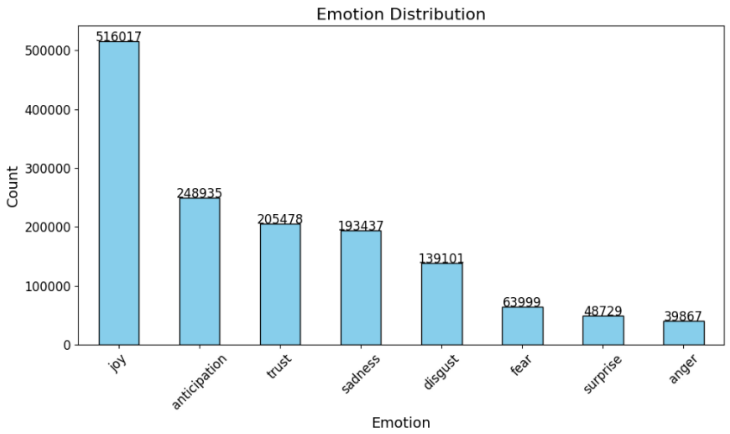

Figure 1. data distribution

Therefore, in the following research, each different method is tested on down sampling dataset (balanced) and the original dataset, mind that we will separate it into 0.7 and 0.3 during training procedure.

II. Method

In this work, we've tried six different ways to predict the emotion.

- TF-IDF Matrix with Multinomial Naïve Bayes Classifier
- TF-IDF Matrix with Decision Tree
- Augmentation with PAMI
- Word2vec embeddings with K Nearest Neighbor
- TF-IDF Matrix with Deep Neural Network
- LLM embeddings with K Nearest Neighbor

i. **TF-IDF Matrix with Multinomial Naïve Bayes Classifier:**

In this section, we first use the TF-IDF vectorizer to change dataset into term-document sparse matrix form. Due to the limitation of RAM, we can only get a maximum of 40000 features. But after several rounds of attempts, 20000 is a good choice. According to what was taught in class, the distribution of Multinomial is very consistent with the distribution of words in the text, so we try this method first. At the end, the original dataset performs 0.52 accuracy on validation. But only 0.39 on Kaggle public leaderboard.

ii. **TF-IDF Matrix with Decision Tree**

In this section, we first use the TF-IDF vectorizer and get 20000 features term-document matrix. Since Decision tree is suffering from overfitting issue. We performed a lot of hyperparameter tuning such as max_depth, min_samples_split, min_samples_leaf, etc. Finally, the original dataset get 0.48 accuracy on validation and 0.33 on Kaggle public leaderboard.

iii. **Augmentation with PAMI**

In this section, We use PAMI try to find the implicit pattern in dataset. However we suffer form RAM limitations and some text decode issue, which takes us lot of time and we gave up this attempt int the end unfortunately.

iv. **Word2vec embeddings with K Nearest Neighbor**

In this section, We try to use pretrain Word2vec model to get word embedding to augment the dataset. First we apply Word2vec model from GoogleNews to get each word's embedding in a document. And then we use the pre-calculated TF-IDF to weighted sum all words in a text to form a unified format features. Unfortunately, we can only apply this preprocessing on down sampling dataset because of RAM limitation issue. After applying this preprocessing, we utilized K Nearest Neighbor Classification to predict the emotion because KNN is work suitable for embedding data. At the end, It only got 0.28 accuracy on validation, so I haven't upload it to Kaggle.

v. **TF-IDF Matrix with Deep Neural Network**

In this section, We use TF-IDF term-document matrix on Deep neural network. Similarly, we also tested the number of features of TF-IDF and

found that 30,000 is a good choice. In this attempt model perform well on original dataset. To deal with overfitting issue, we try on several network structure and utilize early stopping and model checkpoint method. Eventually, We got 0.57 accuracy on validation and 0.45 accuracy on Kaggle public leaderboard, which finally lead to rank 54 on private leaderboard with accuracy 0.44.

Layer (type)	Output Shape	Param #
input_layer_7 (InputLayer)	(None, 30000)	0
dense_22 (Dense)	(None, 128)	3,840,128
re_lu_15 (ReLU)	(None, 128)	0
dense_23 (Dense)	(None, 128)	16,512
re_lu_16 (ReLU)	(None, 128)	0
dense_24 (Dense)	(None, 64)	8,256
re_lu_17 (ReLU)	(None, 64)	0
dense_25 (Dense)	(None, 8)	520
softmax_7 (Softmax)	(None, 8)	0

Total params: 3,865,416 (14.75 MB)
Trainable params: 3,865,416 (14.75 MB)
Non-trainable params: 0 (0.00 B)

Figure 2. model structure

vi. LLM embeddings with K Nearest Neighbor

In this section, we use llama3.2:1b LLM model to help us embed the document. Similarly, due to the RAM limitation issue, we can only apply this method on down sampling dataset. And I still utilize KNN model to predict emotion. At the end we got 0.4 accuracy on validation and 0.36 on Kaggle public leaderboard.

III. Conclusion

I found that the imbalance data still works well in this lab, maybe because the private test dataset distribution is also imbalanced. Based on my observations from trying, the number of parameters of the model does not seem to have much relationship with overfitting. Even if I adjust the model to be very lightweight, overfitting still occurs, and the performance of the model is worse than that of a large model. In future work, I should try other types of pruning such as weight pruning, etc.