Exploring the Efficacy of BERT on a Fine-Grained Emotion Dataset : GoEmotions

Liangshen Yin

Shanghai Jiao Tong University

hnyls2002@sjtu.edu.cn

Abstract

In this paper, We focus on a fine-grained emotions classification dataset : GoEmotions, published by Google Research. The GoEmotions dataset is not so popular, and there are few public results for comparison. We explore the efficacy of BERT, a pre-trained language model, on this dataset. Our work is mainly fine-tuning on a BERT-base model, and we have tried many different hyperparameters and fine-tuning strategies. However, the performance of our model is still far below the state-of-the-art result.

1 Introduction

Fine-grained emotion classification is a challenging task in natural language processing as it requires the model to discern subtle differences between emotions. The GoEmotions dataset **?** contains 58k curated Reddit comments labeled with 27 emotion categories and Neutral, making it an ideal benchmark for evaluation. In this paper, we explore the efficacy of BERT, a pre-trained language model **?**, on this dataset. The code for our experiments is available at `https://github.com/hnyls2002/KFC-emo`.

Given our resource constraints, we adopt a bunch of strategies of fine-tuning the BERT model provided by `HuggingFace` on the GoEmotions dataset, rather than undertaking the daunting task of pre-training on a massive corpus. The pre-trained BERT model we used is `bert-base-uncased`, which has 12 layers, 768 hidden size, 12 attention heads, and 110M parameters.

Our model is simple and consists of a BERT model and one or two fully connected layers. We observe that adding more layers does not necessarily improve performance but increases computation time. Additionally, we experimented with different fine-tuning approaches, such as freezing and unfreezing the BERT model, expanding the dataset and so on. The current best performance we achieved is a macro-F1 score of 0.39 on the test set, which is far below the baseline (also a state-of-the-art result) of 0.46.

Additionally, apart from fine-grained emotion classification in the GoEmotions dataset, we also experimented with two other classification tasks: sentiment grouping and Ekman's taxonomy. The sentiment classification task comprises three classes: `positive, negative, and ambiguous`. On the other hand, Ekman's taxonomy categorizes emotions into six classes: `anger, disgust, fear, joy, sadness, and surprise`.

2 Related Work

2.1 BERT model

Pre-trained language models have recently gained dominance in many NLP tasks, including Question Answering, Natural Language Inference, and Sentiment Analysis. However, in the field of fine-grained emotion classification, the efficacy of pre-trained language models seems to be limited.

Among the pre-trained language models, BERT **?** is the most widely used. It is a multi-layer bidirectional Transformer encoder designed for masked language modeling objectives. The base model comprises approximately 110M parameters while the large model consists of 340M parameters, necessitating enormous amounts of unlabeled data for pretraining. Following the emergence of BERT, various other pre-trained models have been developed to improve its performance or mitigate training time, such as RoBERTa **?**, ALBERT **?**, and so on.

2.2 Baseline for GoEmotions

In the paper **?** detailing the GoEmotions dataset, Google Research authors proposed a baseline model based on BERT. They achieved a macro F1 score of 0.46 on the test set, which is currently the state-of-the-art performance for this dataset. However, with no additional public results available, it is unclear how effective this baseline is.

The baseline model proposed by Google Research is as simple as ours. It just adds one dense output layer on top of the BERT model, keeping most of the hyperparameters unchanged, only modifying the batch size to 16. Moreover, they also provided an additional baseline model, which is a bidirectional LSTM and its performance is not so good.

As the GoEmotions dataset has minimal popularity and does not attract significant attention or participation, there are no other publicly available results for comparison. Therefore, we cannot compare our results with other models.

3 Task

3.1 Features of Dataset

The GoEmotions dataset is collected from Reddit comments, so it contains a lot of informal language and slang, which makes it challenging to classify. And the text classified to be `neutral` comprises a large proportion of the dataset, up to 26.5%. The following figure shows the distribution of each emotion in the GoEmotions dataset (the `neutral` class is not included).

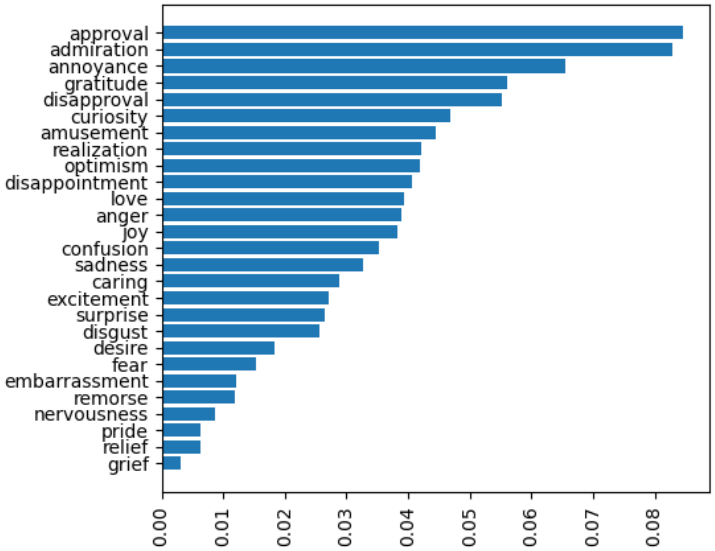

Figure 1: Emotions's distribution

It is worth noting that the text in the GoEmotions dataset can be labeled with multiple emotions. So the classification task is a multi-label classification task, not a multi-class classification task. The following figure shows a few examples of the GoEmotions dataset.

Sample Text	**Label(s)**
Just thinking about all the stupid stuff I did as a kid makes me very afraid for my 5 year old.	`fear`
Wow! You must have pretty low standards for teachers in your state/school! I guess that's why Massachusetts is always first for education.	`admiration, curiosity, nervouness`
I think someone else should open a drive through so I can enjoy the flavours without the dishes aha	`joy`
Lol you're an idiot.	`amusement`
Agree. Not a perfect coach but pretty, pretty good	`admiration`

Table 1: Examples of GoEmotions dataset

4 Approach

4.1 Model

Our model is simply a BERT model with one or two fully connected layers. The BERT model is used to extract features from the text, and the fully connected layer is used to classify the emotion. The following figure shows the structure of our model.

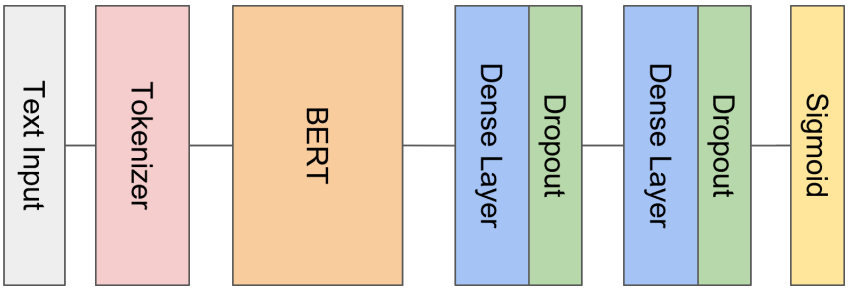

Figure 2: Model structure

4.1.1 Tokenization and BERT Model

The tokenizer and BERT model are both imported from the `HuggingFace.transformers` library. The text input is first tokenized by the tokenizer, and then the tokenized text is fed into the BERT model. All the hyperparameters of the BERT model are kept unchanged, and the output of the BERT model is the last hidden state of the `[CLS]` token.

4.1.2 Inference Layers

In the above figure, our model contains two fully connected layers to do the Inference. However, in the following experiments, we find that adding more layers does not necessarily improve performance but increases computation time. Basically, they are almost the same in performance. So in the next sections of the paper, we would not mainly discuss the number of layers in the inference layers.

4.1.3 Loss Function

As mentioned above, the GoEmotions dataset is a multi-label classification task. So we use the `BCEWithLogitsLoss` loss function provided by `PyTorch` to calculate the loss. Suppose the direct output of the last layer is the digit of each emotion, then the loss function is defined as follows:

$$\text{Loss} = \frac{1}{N}\sum_{i=1}^{N}\left(y_i \log(\sigma(x_i)) + (1-y_i)\log(1-\sigma(x_i))\right)$$

4.1.4 Settings of Hyperparameters

The settings of hyperparameters in the SOTA model are not fully given in the paper **?**, so we have to try different hyperparameters to find the best one.

The batch size is set to 16, larger batch size would cause obviously overfitting. The learning rate is fixed as 5e-5 in the SOTA model, but we use a dynamic learning rate, which is adjusted by the `lr_scheduler.CyclickLR("triangular")`, from 1e-5 to 5e-5.

We do not change the hidden layer dropout probability in the BERT model. In some of our experiments, for the last Inference layers, we set the dropout probability to 0.1 or 0.2, and the performance is slightly improved.

In most of our experiments, the epochs for training are set to 2 or 3, because the models will probably overfit after 3 epochs. However, the total epochs in the fine-tuning process with freezing BERT model are slightly different, as it have two different stages, which will be introduced later.

4.2 Optimization for threshold

28 kinds of emotions have different distributions in the dataset, as shown in Figure**??**. And the loss function is just an average of all kinds, so when we decrease the loss, different kinds of emotions have different rates of decrease. Therefore, we need to set different thresholds for different emotions to get the best results.

Here we choose the macro F1 score as the evaluation metric. To maximize the macro F1 score, we try to maximize the F1 score of each emotion first. For each epoch after we have trained the model through the training set, we set the best threshold according the performance on the training set, hoping that this threshold can also perform well on the validation set. The following figure shows the F1 score and threshold of each emotion in some experiment.

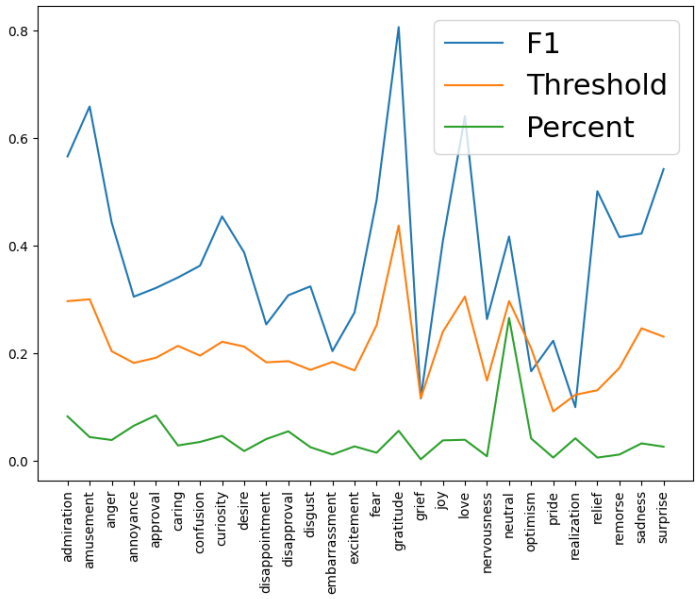

Figure 3: F1 score and threshold of each emotion

We can notice that the F1 score curve and threshold curve are very similar. The emotions with higher F1 scores are also the emotions with higher thresholds. Although the F1 score is actually "chosen by the threshold", the F1 score is still the best we can achieve after training some epochs.

The similarity of the two curves also demonstrates that, when our model has better performance on some emotion, then the output `logits` of this emotion will be easier to classify, that is why the threshold of this emotion is higher.

We can also notice that some emotions have very low F1 scores, such as `grief, pride`, while `gratitute, love, admiration` have very high F1 scores. Looking at the distribution curve, it is not difficult to find that the `grief, pride` are very rare in the dataset, while the `gratitute, love, admiration` are more common. The insufficient data of some emotions may be one of the reasons for the poor performance on GoEmotions dataset.

4.3 Freezing BERT Model and Two-Stage Fine-Tuning

Two stage fine-tuning is a common strategy in the fine-tuning process of BERT model. That is because the billions of parameters in the BERT model are well trained on a large corpus, if we repeatedly train the BERT model on a small dataset, it will cause overfitting. So we apply the two-stage fine-tuning strategy to avoid overfitting.

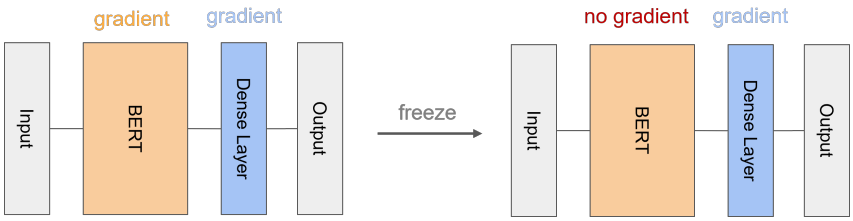

Figure 4: Two-stage fine-tuning

(a) The first stage is to train our model without freezing the parameters inside BERT, this lasts two or three epochs.

(b) In the second stage, we reset all the parameters in our inference layers, and freeze all the parameters in BERT model, then retrain the model for two or three epochs.

4.4 Expanding the Dataset

The GoEmotions dataset is not large enough, so we try to expand the dataset by concatenating two text in the dataset, and also merge their labels. For example, if we have two texts with labels `[joy, admiration]` and `[joy, amusement]`, then we can get a new text with label `[joy, admiration, amusement]`.

However, this simple method would cause some problems, as the emotions are not orthogonal. When we concatenate two sentences with opposite emotions, the new sentence may be meaningless. For example, `I love you` plus `I hate you` would be `I love you.I hate you`, which is so weird.

Although this method seems to have some problems, we still try it in our experiments. We concatenate two texts randomly, expand the training set to 1 million texts. However, the training time is too long, we only train the model for one epoch without freezing BERT model. After that, we continue to train the model for another several epochs, but this time we freeze the BERT model. The performance is not improved significantly, but not so bad.

5 Experiments

This section presents the results of our experiments. As previously mentioned, we experimented with various hyperparameters and fine-tuning strategies. Our evaluation metric is the macro F1 score, along with the weighted F1 score and accuracy for reference. It should be noted that accuracy is not an ideal metric for multi-label classification tasks because all dimensions of the prediction must be correct for a text to be accurately labeled.

The following table shows the results of all our experiments.

Experiment	Epochs	M-F1	W-F1	Acc
DenseLayer(1) + Dynamic lr	2	0.27	N/A	0.22
DenseLayer(1) + Fixed lr(5e-5)	4	0.26	N/A	0.23
DenseLayer(1) + Fixed lr(5e-5) + TwoStage	2 + 2(frz)	0.29	N/A	0.24
DenseLayer(1) + Dynamic lr + TwoStage	2 + 4(frz)	0.29	N/A	0.24
DenseLayer(1) + Dynamic lr + Dropout(0.1) + TwoStage	2 + 6(frz)	0.29	N/A	0.24
DenseLayer(1) + Dynamic lr + Dropout(0.1) + TwoStage + OptOnThreshold	2 + 3(frz)	0.38	0.44	0.23
DenseLayer(1) + Dynamic lr + Dropout(0.1) + TwoStage + OptOnThreshold	4 + 4(frz)	0.39	0.44	0.25
DenseLayer(1) + Fixed lr(5e-5) + OptOnThreshold + ExpanedDataset	1	0.35	0.41	0.18
DenseLayer(2) + Dynamic lr + Dropout(0.1) + TwoStage + OptOnThreshold	4 + 3(frz)	0.39	0.44	0.23
DenseLayer(2) + Dynamic lr + Dropout(0.2) + TwoStage + OptOnThreshold	4 + 3(frz)	0.39	0.45	0.24
DenseLayer(1) + Dynamic lr + Dropout(0.1) + OptOnThreshold + TwoStageOnExtended	1 + 3(frz)	0.37	0.42	0.24

Table 2: Results of all experiments

5.1 Sentimentment and Ekman

We also experimented with two other classification tasks: sentiment grouping and Ekman's taxonomy. The mapping from emotions to sentiment and Ekman's taxonomy is shown in the following table, which achieves good results nearly reaching the SOTA result.

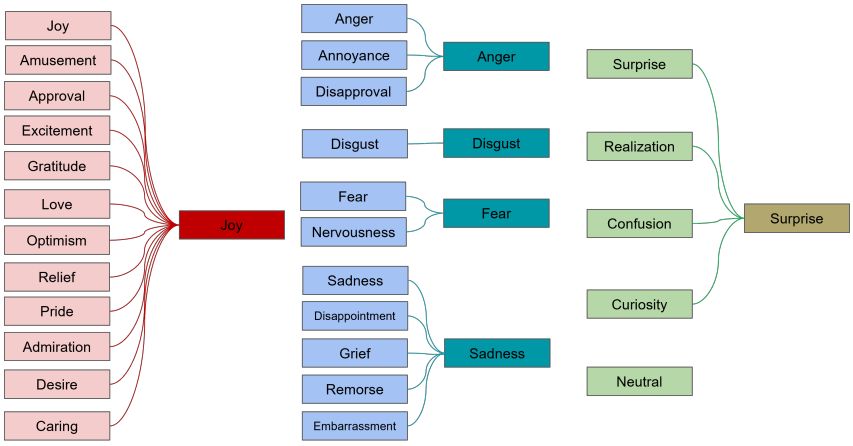

Figure 5: Mapping from emotions to sentiment and Ekman's taxonomy

And the results of our experiments are shown in the following table.

Classify	Experiment	Epochs	M-F1	W-F1	Acc
Sentiment	**DenseLayer(1) + Dynamic lr + Dropout(0.1) + TwoStage + OptOnThreshold**	4 + 3(frz)	0.69	0.69	0.51
Ekman's	**DenseLayer(1) + Dynamic lr + Dropout(0.1) + TwoStage + OptOnThreshold**	4 + 3(frz)	0.50	0.59	0.41

Table 3: Results of sentiment and Ekman's taxonomy

6 Future Work

6.1 Expand the Dataset

We can not expand the dataset by simply concatenating two texts, as mentioned above. We need to find a better way to expand the dataset, such as a filter to filter out the meaningless texts.

Manually labeling the dataset is also a good way to expand the dataset, but it is too costly.

6.2 More (complex) Layers in Inference Layers

One or two layers in the inference layers are basically the same in performance, but we can try much more layers in the future.

We can also try some other layers, such as `LSTM` or `GRU`.

6.3 Pre-train BERT on GoEmotions-Like Dataset

Normal datasets are not suitable for fine-grained emotion classification, as the emotions in normal text are not so outstanding. So we can pre-train BERT on a GoEmotions-like dataset, which is more expressive in emotions.

7 Conclusions

The result on GoEmotions classification is not so good, but the results on sentiment and Ekman's taxonomy nearly reach the SOTA result.

Thought We have tried many different hyperparameters and fine-tuning strategies, the performance of my model is still far below the baseline. The fine-grained emotion classification is such a challenging task, even an adult can not precisely recognize the subtle differences between emotions.

GoEmotions dataset also contains bias, as it is collected from Reddit comments, which is not so formal. The authors from Google Research also mentioned that the GoEmotions potentially contains problematic content, such as hate speech, offensive language, and so on.

The simple BERT-based model I designed is also not so great to deal with this task. Maybe the features extracted from the BERT model are not so suitable for this task, or maybe the inference layers are quite simple.

Acknowledgements

Though my work is not so good, I truly learned a lot from this project. I would like to thank my senior, Yexin Wu, my classmate Boyang Zheng, and my machine learning teacher, Prof. Weinan Zhang, for their help and guidance.