Individual Report of Final Project

He Huang

Introduction

Our project is about text classification task. We separated the whole task into 4 parts, EDA part, rule-based model part, transformer part and summary part. Dang mainly took charge of the EDA part, Li mainly took charge of the rule-based model part, I mainly took charge of the transformer part and Liang mainly took charge of the summary part.

Overall Description

After our discussion on this project, I wrote the code of transformer part, which contains several different transformers and different custom head. Then I helped Liang to run the summary of each text. With the help of Li and Liang, I trained most of the transformers and tested their performance. Then I helped Liang to check run just summary on those models.

For the final report, I wrote the transformer part, and helped Li to finish the result and conclusion part. Also, in making slides, I created the transformer part's slides.

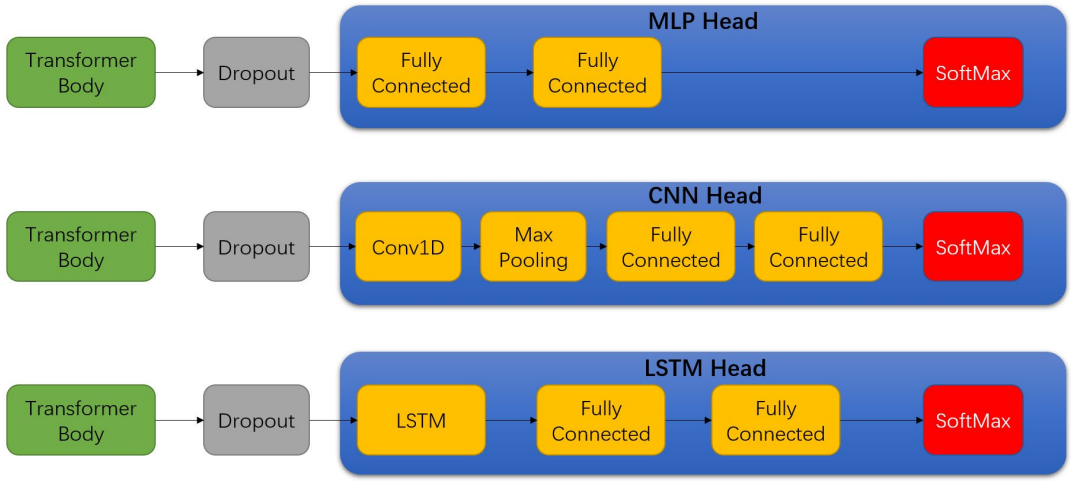
Details in coding and training

After checking those rule-based models' performances, I started to use some statistical models on this text classification task. At first, I tried to design our own MLP, CNN or LSTM models, but their performances were so pool. So, I started to implement transfer learning with the pre-trained models, in other words, transformers. Because transformers have already been trained on plenty of text, they work well than normal

simple deep neural networks that I designed. However, just using transformers is not enough, and I need to adjust them for our task. So, I used transformers as models' body, and add some special heads for our text classification task. Basically, I used 3 kinds of heads, MLP, CNN and LSTM, and their structures are shown in figure below. I added dropout layers right after the transformers body and between head layers to avoid overfitting. Also, according to Hendrycks and Gimpel(2016), GELU activation function is mathematically better than RELU activation function in nonlinearity task. So, I used GELU function in most fully connected layers. In the output layer, because this task is multiclass classification, I used SoftMax function.

Figure

Structures of Models



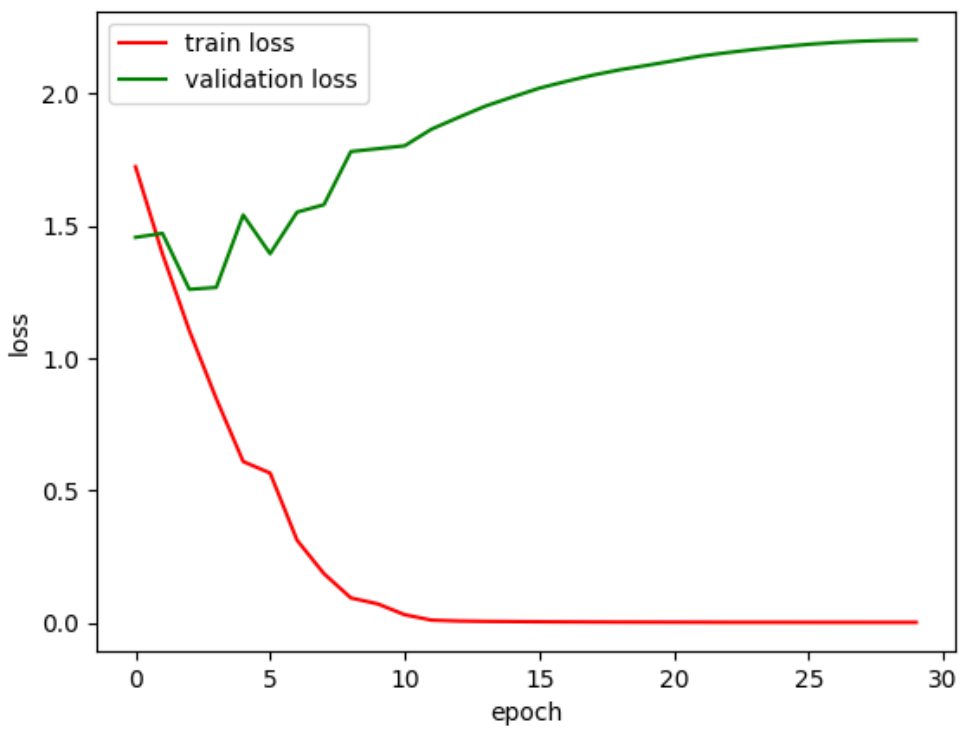
Before training, I needed to use tokenizer to tokenize the text data. In this training data, I calculated the distribution of number of tokens in each text data, which is shown in figure below. Because most of the number of tokens are less than 150, I choose 150 as the max length.

To avoid overfitting, I collected the loss changes of training data and validation

data, and one of the changes in early training attempt is shown below. In this training, the train loss kept decreasing, but the validation loss was oscillating reduced in the first 4 epochs, and then validation loss kept increasing. This change of loss showed that overfitting occurred. So, I reduced the learning rate, added more dropout layers in the head and increased the dropout rate of the transformers.

Figure

Change of Loss in One Training Attempt



First, I tried base BERT model as transformer head. After around 10 – 15 epochs, these models got their best performance. The best perdition results of each model are shown in Table below. Considering both accuracy and F1-macro score, MLP head and CNN head performed very closely, MLP had a little better F1 score, and CNN had a little better accuracy. In these models using BERT body, LSTM head performed best,

which I guess is because LSTM has memory to use previous sentences or tokens to make prediction. What's more, I tried different transformer bodies such as RoBerta, XLM-RoBerta, and BigBird, but their performances were not good. The best one XLM-RoBerta with LSTM head only has an accuracy below 0.7.

Table

Results on Test Data

Transformer body	Custom Head	Accuracy	F1-macro
bert-base-cased	MLP	0.759847522	0.733070241
bert-base-cased	CNN	0.764612452	0.729890715
bert-base-cased	LSTM	0.783989835	0.760746121
xlm-roberta-base	LSTM	0.643348561	0.620621515

In all the transformer body plus custom head models, BERT-LSTM model is the best one. The difference between different transformers body is larger than the difference between different custom head.

Also, I put replace the text with only the summary of the text in test section and tested if this change works. Actually, the models' prediction got worse. So, I thought maybe using summary as training text will be better.

Results and conclusions

To sum up, difference between different transformers is bigger than the difference between different custom head. BERT plus LSTM model has the best performance. For future improvement, with more GPU for greater calculation power, we can use all the tokens in the text and get better performance. Also, maybe other transformer is better in this text classification task.

Percentage

The percentage of code I found and copied from internet:

(120 - 15) / (120 + 180) = 38.3%

References

Hendrycks, D., & Gimpel, K. (2016). Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Sangani, R. (2022, January 26). Adding Custom Layers on Top of a Hugging Face Model. Medium. https://towardsdatascience.com/adding-custom-layers-on-top-of-a-hugging-face-model-f1ccdfc257bd