**RMS Project – Final Interim Report**

|  |  |
| --- | --- |
| Student name / Student no.: | ZHOU Yukun |

|  |  |
| --- | --- |
| Project Title: | < Risk Prediction by Sentiment Analysis in NLP > |

|  |  |  |
| --- | --- | --- |
| Supervisor: | **<SONG Linqi>** |  |

(1)    Introduction and Topic of Research   
This project provides a demo tool to help company make the risk prediction with Natural Language Processing skills. The risk manager can't handle all the latest news and people's trend in a short time, also for the investor. Sometimes, that's mortal for a company. Hence, it's necessary for us to develop a tool to handle the sentiment trend in the real-time for manager to deicide and avoid the potential risks. We choose twitter, a famous social media with the large users' base, as our data source. We also attempt to grasp the news title and analysis the sentiment to get more convincing result. Moreover, combine them with knowledge graph to get more real effect.

(2)  Implementation, Experiments, Analysis

I use Google CoLab to build my simple demo. The Tweet Data with sentiment label and other information we used is collected by Vivek Rathi in Kaggle.

**Model Building Based on Fine-tuning BERT**

Text

Description automatically generated

The demo aims to get a sentiment score of a company and therefore we need the model to do the classification work. The BERT can used for classification through fine-tuning, which adds some layers to transform for other task before the last layer output. After the output of the origin BERT model, add two feed-forward layer to compress the output to 3 dimension, representing 3 sentiment label, negative, natural, positive. Also, there is a softmax function to transform the result to the probability style.

**Data Processing**

Graphical user interface, application

Description automatically generated

This part is for data loading. Data\_with\_name stores the tweet data with corresponding company information and data is connected to the sentiment label.

Graphical user interface, text, application, email

Description automatically generated

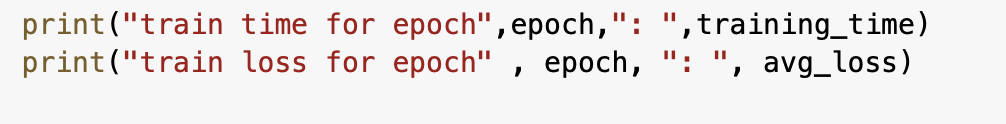
First, load the pre-trained BERT model from Hugging Face. For the tweets, some of them are too long to be fed to the BERT. We have different deleting method but according to the research, combining the start and the end is the best method to get the most accurate result. Then tokenize the sentence with numerical style.Graphical user interface, text, application

Description automatically generated

Padding is for generating tensor, which is a data collection format like matrix in Pytorch. Use a loop to add 0 to fill the gap. Attention mask is a must for BERT. BERT basic task will cover some tokens in the sentence and predict them to train the BERT. We add an attention mask to avoid the modification of those 0s after padding.

To utilize our data better in the following train and test stage, setting data loader can help to split data to many batches as input to avoid the heavy computation. With GPU features, batch size should be times of 32 to maximize the efficiency. By testing, 64 batch size occupies too much resource, and 16 batch size is not quite accurate for our project.

**Train and Test**Graphical user interface, text, application

Description automatically generatedGraphical user interface, text, application

Description automatically generated

In training stage, different epoch may affect the result. With small data testing, we get epoch=4, final eval loss=0.312, MCC score = 0.833; epoch=3, final eval loss=0.403, MCC score = 0.785; epoch =5, final eval loss = 0.394, MCC score = 0.890. In total, we find 3 is not enough for training and 5 is a little overfitting because the eval loss is higher when the size is 4, even the MCC score seems better. The training uses AdamW as updated version of normal Adam optimizer to decide how the model updates. And the scheduler here are used to update the learning rate. The last part is validation. Validation is used for testing the epoch training result and adjusting the corresponding parameters like epoch, batch size and so on.

Graphical user interface, text

Description automatically generated



In each epoch, it uses the same data to train. Here first move padded input, attention\_mask, and the sentiment label to the GPU for train use and feed them to the model constructed before to get a (32,3) matrix, representing 32 samples in a batch and 3 labels. The model returns the loss between true label and prediction, and calls backward() to calculate the hyperparameter. The clip line is for vanishing gradient problem solving. Set a value 1.0 to fill the vanishing part to the 1. Finally, it generalizes the loss of an epoch.

Graphical user interface, text, application, email

Description automatically generated

The validation function utilizes the similar structure of the train. The main difference is set the model to the evaluation mode and use "with torch.no\_grad()" to exclude the gradient backward process from the loss calculation to make sure the model doesn't change when doing validation.

Text

Description automatically generatedGraphical user interface, text, application

Description automatically generated

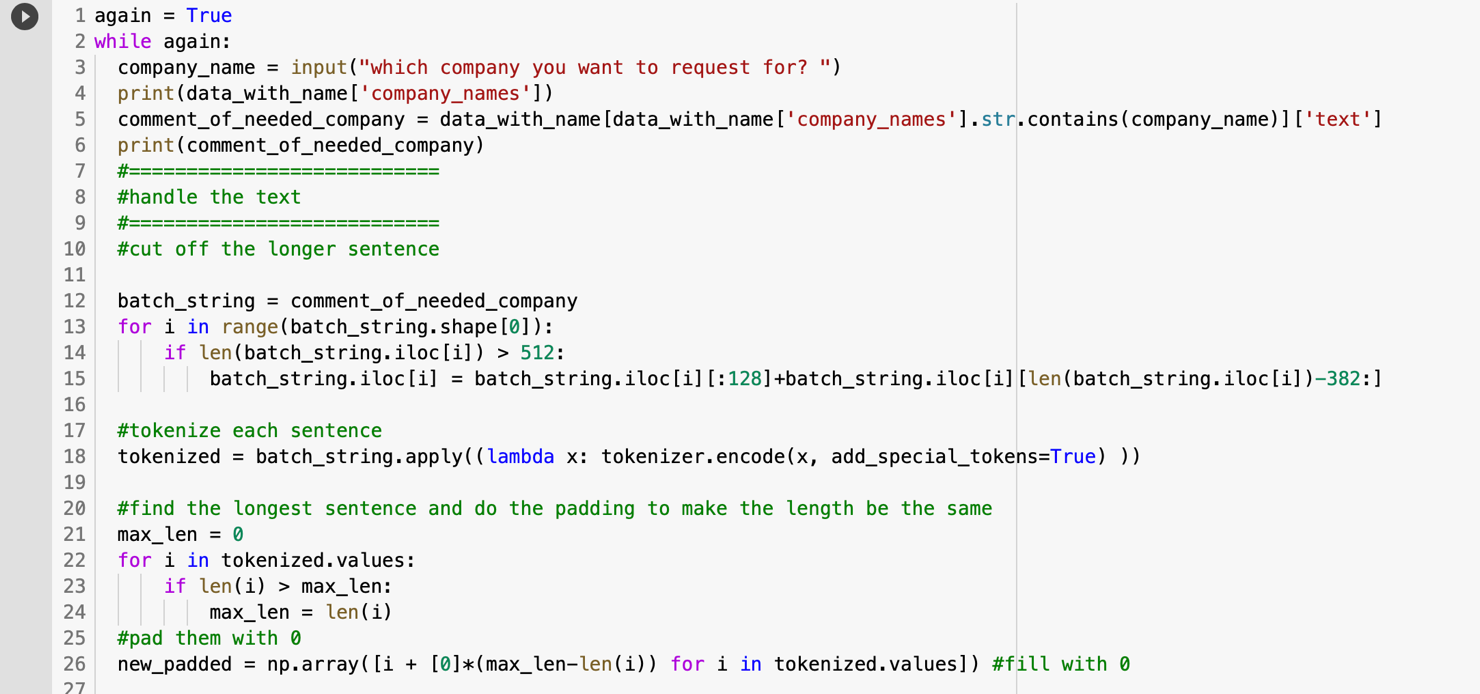
Next, after all modification finished, take the test dataset for testing the model. In line 24, take the index of the maximum prediction value as its sentiment label result and use Matthews correlation coefficient method to calculate the score of our multiple classification model. It returns the correlation coefficient with +1 for prefect prediction, 0 for random prediction, -1 for complete opposite result.

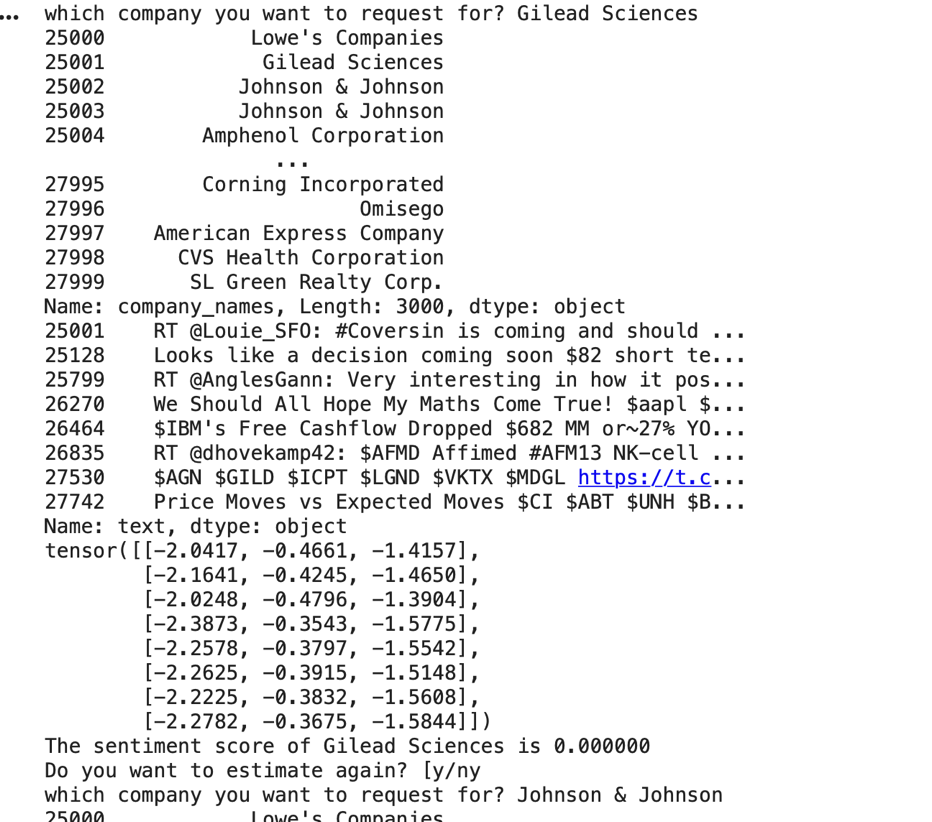
The result below shows the test set reach 0.971. To avoid the fake high score caused by random split set, I completely re-train the model and it also gets the score higher than 0.8. Therefore, the model has high prediction accuracy.

Text

Description automatically generated with medium confidence

Finally, I design the part for company searching and returning the sentiment scores. It depends on the company name given in the data and calculated by averaging the prediction score.

**Application Scenario**For the codes, it has similar structure with the test part. Process the needed company tweet data with tokenizing, padding operation and so on. Feed them to the model and see the prediction result.



(5)    Conclusion and reflection on experience gained through participation in RMS  
The whole RMS project is not an easy task. The most important part for me is self-study ability and self-control. The study of CS224n offered by Stanford University is very important for the project and even for my academic and career life. I learned how to persist on study and how to solve uncountable difficulties including confusion about the knowledge when studying, infinite bugs when building the project.

For knowledge itself, I have some understandings about Deep Learning especially for NLP field. The data processing skill and real train and test procedure design for model are also a rich harvest. Finally, thanks for Dr. Song's help.

(6)    References

*1) BERT Fine-Tuning Tutorial with PyTorch*. BERT Fine-Tuning Tutorial with PyTorch · Chris McCormick. (2019, July 22). https://mccormickml.com/2019/07/22/BERT-fine-tuning/#31-bert-tokenizer.

2) Elijha.css-1cd9gw4{margin-left:.3em;}忙于人工智能的人类智障(n.d.). *Pytorch数据读取(Dataset, DataLoader, DataLoaderIter)*. https://zhuanlan.zhihu.com/p/30934236.

3) Prateek JoshiData Scientist at Analytics Vidhya with multidisciplinary academic background. Experienced in machine learning. (2020, July 26). *Transfer Learning NLP: Fine Tune Bert For Text Classification*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/07/transfer-learning-for-nlp-fine-tuning-bert-for-text-classification/.

*4) Pretrained models*. Pretrained models - transformers 4.7.0 documentation. (n.d.). https://huggingface.co/transformers/pretrained\_models.html?highlight=based+bert.

5) Rathi, V. (2019, September 5). *Sentiment Analysis on Financial Tweets*. Kaggle. https://www.kaggle.com/vivekrathi055/sentiment-analysis-on-financial-tweets?select=tweet\_sentiment.csv.

*6) sklearn.metrics.matthews\_corrcoef¶*. scikit. (n.d.). https://scikit-learn.org/stable/modules/generated/sklearn.metrics.matthews\_corrcoef.html.

*7) Tutorial: Fine-tuning BERT for Sentiment Analysis - by Skim AI*. Skim AI. (2020, December 31). https://skimai.com/fine-tuning-bert-for-sentiment-analysis/.

8) 深度学习调参技巧 - Jing's blog. (n.d.). https://tsinghua-gongjing.github.io/posts/DL-tricks.html.