NLP for Overall Market Sentiment Prediction of Entities

In this project our goal was to bring insight to investors on the general sentiment of the market for a given company. To do this we created a model that looked at different financial news datasets and performed NLP and sentiment analysis. The purpose of this project was to get a sense of how well a company is doing in order to take this into consideration when deciding whether or not to invest.

Our methodology consists of several steps the first of which was data consideration and collection. For our data, we used a Kaggle database with 4837 unique financial news articles. Each article had an associated label 'positive', 'negative', or 'neutral' sentiment depending on the content of each article. The dataframe we created contained these 'labels' and we named the content of each article as the 'text' parameter.

After deciding the database we analyzed the overall data to understand it before pre-processing. We checked for null values and found that there were none. We also checked the distribution of data between each label. This was a key analysis as it helped us figure out that the data was imbalanced and hence it would affect the predictions. With this, the model was biased to guess 'neutral', which was the majority class. Therefore, to counteract this and make sure the model made accurate predictions, we oversampled the minority classes, which we later cover in this paper.

Once explored, we used the Bert Tokenizer to pre-process the data. This is a key step in order to prepare the input data for the BERT model by tokenizing sentences, adding special tokens, and creating attention masks. More specifically, the tokenizer transformed text to lowercase letters, for the model to understand the sequence structure. Then, it transforms the sentences into word-level tokens and converts these into embeddings (vectors). Those vectors represent the meaning of each word given the context.

In addition, as part of the pre-processing and for the feature creation, we named the two columns in the dataframe 'text' and 'labels' and converted the labels into categorical numerical values for easier processing. Neutral is 2, Negative is 0, and Positive is 1. The data was then split as follows, 80% training set and 20% testing set which turned out to be very efficient for this model.

We also added a new feature called 'named_entities' to our model. In order to conduct this we used NLTK named entity recognition in python to process each article to find the principal entity it is discussing. Then, looking at the named_entities column we determine the 100 most frequent companies across the articles. We do this in order to determine the current most relevant/talked about entities in the market. So later we can show the majority sentiment of the market on those specific companies. It is presented at the end as a chart with all the companies names and their respective sentiment.

Regarding the model itself, we decided to run it on GPU due to time efficiency when training the machine learning model. Then we decided to use the pre-trained Bert Model 'BertForSequenceClassification' because it is great for sequence classification tasks since it involves training on both word and sentence representations. First, it pre-trains on word embeddings, learning contextualized representations for individual words in a specific context. Then, it fine-tunes the model to understand the connection and context between sequences of words. The model takes the embeddings as input and operates on tensors during the forward and backward passes and adjusts its weights to minimize the difference between the predicted and true labels. To minimize the differences between the predicted and actual output, we chose AdamW as our optimizer, which allowed for the model's parameter to be updated and hence have a lower loss function.

Finally, in order to train the model we ran 4 epochs, meaning 4 complete passes through the entire training dataset to make sure the model learnt well. We analyzed the average training loss, the training time, the model's accuracy, validation loss and validation time.

In our results, initially we found that we had only 85% accuracy in predicting the sentiment. However, this was with no oversampling and 4 epochs. To improve the accuracy we decided to collect more data, from another source. This would help the model with generalization (seeing different styles and quality of text). To implement this strategy, we wrote a python script that fetches data from the marketaux API which is a page with different financial articles. The articles taken from the API however, did not have a sentiment classification without the premium subscription, so we had to read each of the documents and correctly classify them ourselves. We formatted the API data to have the same 2 columns and format as the database data and merged them together. With this change we saw that after running the more complete data on the model, the accuracy only increased from 85% to 86%.

Therefore, as previously mentioned before, we turned to oversampling. Since we noticed that there was an imbalance in the type of articles, an excess of 'neutral' articles, we used Random Oversampling to oversample the minority label text. This way the data would have balanced categories and the model wouldn't be biased. The effect of the oversampling in our model was an increase in accuracy from 86% to 94%.

The last thing we also decided to consider was the epochs. Looking at the training and validation graph, we saw that after the third epoch even though the validation accuracy kept increasing, the validation loss started increasing as well. That could be due to the model overfitting to the training data. Therefore, we reduced the number of epochs to 3.

After taking the three strategies said above, we were able to then completely analyze our findings and be more confident in them. Firstly, the results from the confusion matrix show that the model achieved a 92% accuracy level with a mean squared error of 11%. This shows that the model accurately predicts the sentiment of a given financial news article. Therefore we could easily input articles from a specific company to the model and see what is the general sentiment of the market for that particular company.

Moreover, we see that the model learns from the training data as in epoch 1 we have a training loss of 51% which then decreases to 9% in epoch 3.
The validation loss also decreases across epochs which shows that the model generalizes the data well to the validation set, from 29% to 26%. This shows as well that since we have a low validation loss, the model is not overfitting to the training data

Regarding the validation accuracy, the model reaches 92% in the final epoch which reflects that the model performs well on unseen data and learns well from the training set. Also, the mean validation accuracy is 91% which again reflects the aforementioned across epochs.

Overall, the training time and validation time are consistent and reflect efficient training and quick validation predictions. In conclusion, the results show that the BERT model is effectively learning from the training data, generalizing well to the validation set. Therefore we can conclude that it achieves good accuracy. Moreover, both training and validation losses decrease showing that the model generalizes well to unseen data. We also conclude that the high validation accuracy suggests that the model is capable of making accurate predictions on new data.

During this project we both learnt a lot about the BERT model and especially about natural language processing. Having never used nltk in python it was fun to discover its different uses. We also explored other ways in which we could have implemented the 'by company' feature such as using BERT's sentiment vectors and finding where a sentence with x company tends to. Even though we did not take this approach we conducted research for potential future implementation.

Bibliography of data and tokenization part of model:
https://www.kaggle.com/datasets/ankurzing/sentiment-analysis-for-financial-news
https://www.kaggle.com/code/arashnic/bertvsfinbert-v2
https://www.marketaux.com/documentation