Predicting Earnings Calls Surprises Using FinBERT

Sheetal Sudhir Srilatha
Department of Computer Science and Engineering
University of Connecticut

Abstract - **In the financial industry, it is crucial to predict the earnings surprises of companies, which are defined as the difference between the actual earnings and the earnings that were predicted by financial analysts. Predicting earnings surprises can have significant impacts on stock prices and investor decisions. In recent years, natural language processing (NLP) techniques have been used to predict earnings surprises using the text from earnings calls. FinBERT is one of the pre-trained NLP models that has been applied to this task. This literature review examines the effectiveness of FinBERT in predicting earnings surprises by analyzing and comparing the results of three papers that have utilized FinBERT for this purpose.**

Introduction

[1] This paper presents FinBERT, a pre-trained language representation model based on the BERT architecture that is specifically designed for financial text mining tasks. The authors demonstrate the effectiveness of FinBERT on various financial text classification tasks, including sentiment analysis of financial news and earnings call transcripts. FinBERT achieved state-of-the-art results on multiple benchmark datasets, indicating its strong performance in understanding the nuances of financial language.

For predicting earning call surprises, FinBERT can be fine-tuned on a dataset of earnings call transcripts labeled with surprise/non-surprise labels. The model can then be used to classify new transcripts as either containing a surprise or not. The paper's experimental results suggest that FinBERT has the potential to perform well on such tasks.

[2] This paper explores the use of BERT for predicting financial market reactions to macroeconomic news announcements. The authors fine-tuned BERT on a dataset of news articles and their corresponding market reactions and evaluated the model's performance on a hold-out test set. They found that BERT was able to accurately predict the direction of the market reaction (positive or negative) for a given news article, outperforming several baselines.

While this paper does not directly address the task of predicting earning call surprises, the methodology presented can be adapted to this task by fine-tuning BERT on a dataset of earnings call transcripts and their corresponding market reactions (e.g., stock price changes). The model can then be used to predict the market reaction to new earnings call transcripts, which can be used as a proxy for surprise/non-surprise labels.

[3] This paper investigates the use of textual analysis for stock market prediction using financial news articles. The authors used a bag-of-words approach to extract features from news articles and trained several classifiers to predict the direction of the stock market. They found that the Random Forest classifier performed the best, achieving an accuracy of 60.7%.

While this paper does not use FinBERT specifically, the bag-of-words approach and machine learning methodology presented can be adapted to the task of predicting earning call surprises using FinBERT embeddings as features. A classifier can be trained on a dataset of labeled earnings call transcripts using the FinBERT embeddings, and the model can be used to predict the surprise/non-surprise label for new transcripts.

Overall, these papers provide valuable insights into the use of language representation models and machine learning for financial text-mining tasks. By fine-tuning FinBERT on a dataset of earnings call transcripts labeled with surprise/non-surprise labels, and using the methodologies presented in these papers, I can hopefully develop an effective model for predicting earning call surprises.

Overview of Sentiment Analysis and Earnings Surprises

Sentiment analysis is a natural language processing technique used to identify and extract subjective information from text, such as opinions, emotions, and attitudes. Sentiment analysis aims to classify the sentiment of a text as positive, negative, or neutral. It has various applications, such as understanding customer opinions, monitoring brand reputation, and predicting market trends.

On the other hand, earnings call surprises are an important metric used in financial analysis. An earnings call is a conference call held by a public company's executives to discuss their company's financial results for a given reporting period, usually quarterly or annually. An earnings surprise occurs when a company's actual earnings per share (EPS) or revenue exceeds or falls short of the consensus estimate of analysts. An earnings surprise can significantly impact a company's stock price and is closely monitored by investors and analysts.

Project Statement

The goal of this project is to develop a predictive model using FinBERT to identify and forecast earnings call surprises. The project will build on the existing research on sentiment analysis of financial reports using NLP techniques, particularly on the use of FinBERT for analyzing earnings calls. The proposed model will take into account both the sentiment of the earnings call and the company's financial metrics leading up to the call to predict whether the call will result in a surprise or not. The model will be trained and tested on a dataset of earnings calls and their corresponding financial metrics to evaluate its performance. Ultimately, the model aims to provide valuable insights to investors, analysts, and other stakeholders in predicting the likelihood of earnings call surprises and informing investment decisions.

- Dataset: Stock Values and Earnings Call Transcripts: a Sentiment Analysis Dataset — DataverseNL

Data Collection and Visualization

In the data visualization step, the earning call transcripts for 10 different companies were collected. These companies include Apple, Advanced Micro Devices, Amazon, ASML, Cisco, Google, Intel, Microsoft, Micron Technology, and Nvidia. I ensured that the selected companies were from different industries and with varying levels of volatility in their earnings surprises to ensure a diverse and representative dataset. This data will later be used to train the model.

To prepare the text data for the visualization step, the text was preprocessed by removing stopwords and unnecessary punctuation. The purpose of text preprocessing for generating visualizations of text is to clean and transform raw text data into a format that is more suitable for analysis and visualization. The goal of this process is to reduce the dimensionality of the data and to remove noise so that patterns and insights can be more easily identified.

Once the text was processed, I generated visualizations of the transcripts. This was done to get a better understanding of which words were associated with specific sentiment values, including positive, negative, and neutral. Additionally, n-grams with n=1 were used to generate insights into which words appeared most frequently across the different sentiments. This information was used to generate plots as well as word clouds of frequent words across different sentiments.

After visualizing the data, the data distribution for the different sentiment values was determined. The most frequent sentiment was found to be neutral, with a count of 2879, followed by positive with 1383 and negative with 621. The class imbalance was handled by undersampling the majority class.

Project Process

One of the most challenging aspects of this project was trying to find a dataset with information pertaining to earnings call transcripts. Most of the data that was publicly available free of cost contained pages of transcripts and contained various formatting issues that would require a significant amount of time to fix. Additionally, with FinBERT, it is recommended to keep the input length within 512 tokens for optimal performance, so I needed a dataset that contained the main highlights of earnings call transcripts. Fortunately, I was able to find this information on dataverse.nl. This dataset contains highlights of specific earning call transcripts, capturing the most important points of each call. As mentioned earlier, this data contains transcripts of 10 companies in the US. This data is fairly recent as it originated in the period 2016-2020 and is related to the NASDAQ stock market. This data will later be used to train the model.

The model training and testing implementation begins with generating two variables, X and y. X is the text data from the earnings calls, and y is the corresponding sentiment label. The sentiment labels are converted into numerical values using a dictionary, where 'positive' is assigned the value 2, 'neutral' is assigned the value 1, and 'negative' is assigned the value 0.

The next step was to tokenize the text using the tokenizer provided by the Hugging Face Transformers library. Tokenization is the process of converting words into numerical tokens that can be understood by the machine learning model. The tokenizer converts the text data into tokenized sequences and returns a dictionary that contains input_ids, which represent the numerical values of the tokens, and attention_mask, which indicates which tokens should be given attention by the model. These tokenized sequences are split into train and validation sets using the train_test_split function from the sklearn library.

I then decided to create data loaders for the train and validation sets. These data loaders are used to load data in batches during the training and validation phases. PyTorch's DataLoader class is used to create data loaders, which take input_ids, attention_mask, and labels as inputs. In essence, data loaders are objects that take in a dataset and return batches of data during training. They typically shuffle the data to help with generalization and split the data into batches to improve memory usage. They can also apply transformations to the data such as normalization or data augmentation. They are particularly useful when working with large datasets that can't fit into memory all at once, as they can load batches of data on-the-fly. They also simplify the process of working with different datasets or subsets of data, as the same data loader can be used for all of them.

The AdamW optimizer is used to optimize the model parameters during training. It is a modified version of the Adam optimizer that uses weight decay to prevent overfitting. The learning rate is set to 5e-5, which is a commonly used value for fine-tuning pre-trained language models.

The model is trained using a loop that runs for a specified number of epochs, which I decided to set to 5. I chose to stick with 5 epochs for my implementation for two very important reasons. The first reason is that the performance of the model began to plateau after the fifth epoch. This was the point at which the model started to overfit to the training data, where it starts to perform well on the training data but poorly on the validation data. In other words validation loss started to increase after the fifth epoch, and the training loss continued to decrease. The second reason I fixed the number of epochs at 5 is because it is very computationally expensive to train the model for longer and I lacked the resources to do so. Within each epoch, the data is loaded in batches using the data loader, and the model is trained using the loss calculated from the outputs. After training on each batch, the optimizer's gradient is zeroed, and the next batch is loaded until all batches are used.

After each epoch, the model is evaluated on the validation set. The model's eval mode is turned on, and the validation data is loaded using the validation data loader. The validation loss, accuracy, and number of evaluation steps are calculated and printed for each epoch. The validation accuracy is the percentage of correctly classified sentiment labels in the validation set.

Additionally, as mentioned earlier, the data that was utilized for this project is imbalanced. I decided to undersample the majority class (negative sentiment) utilizing the RandomUnderSampler that is available through the imbalanced-learn library.

Another important point is that I wanted to check the model performance for imbalanced and balanced data. Performing the above process on both imbalanced and balanced data allowed me to compare the performance of the model on both types of data.

This was a crucial step because by comparing the model's performance on imbalanced and balanced data, we can assess whether balancing the data through undersampling has improved the model's performance or not. If the model performs similarly on both types of data, it suggests that balancing the data did not significantly improve the model's performance. On the other hand, if the model performs significantly better on balanced data, it indicates that balancing the data was effective in improving the model's performance. This information can help us make decisions on how to handle imbalanced data in similar machine-learning tasks in the future.

Results

The given results show the performance of a model trained on earnings call data using FinBERT to predict if the earning call was a positive surprise, negative surprise, or neutral surprise. The results are presented for both unbalanced and balanced data. In the case of unbalanced data, the model achieved a validation accuracy of 0.8567 in the first epoch, which improved with each subsequent epoch and reached a peak of 0.9068 in the fifth epoch. The training loss decreased consistently over the five epochs, indicating that the model was effectively learning from the data.

In contrast, the model trained on balanced data had a lower validation accuracy, starting at 0.8097 in the first epoch and reaching a maximum of 0.8578 in the fifth epoch. However, the training loss was significantly higher in the first epoch and decreased sharply in the subsequent epochs, indicating that the model was struggling to learn from the imbalanced data. The validation loss was also higher in the first epoch but decreased consistently over the five epochs, indicating that the model was still able to learn some useful patterns from the data.

Overall, the results suggest that the model performed better on unbalanced data, achieving higher accuracy and lower loss. However, the model trained on balanced data still achieved reasonable results, suggesting that undersampling could be an effective technique for improving the performance of machine learning models on imbalanced data. It is important to note that the results could be further improved by tuning hyperparameters or using more advanced techniques like oversampling or data augmentation.

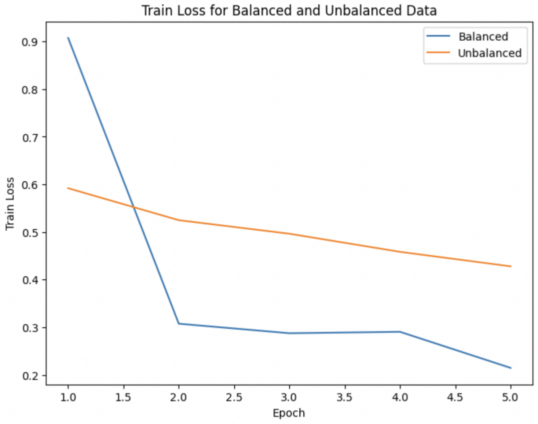

Fig 1. Training loss for balanced vs. unbalanced data

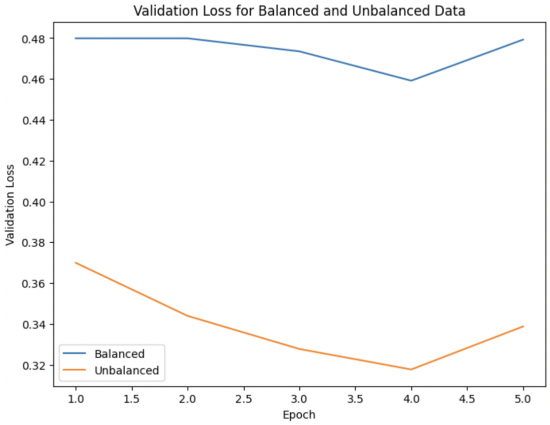

Fig 2. Validation loss for balanced vs. unbalanced data

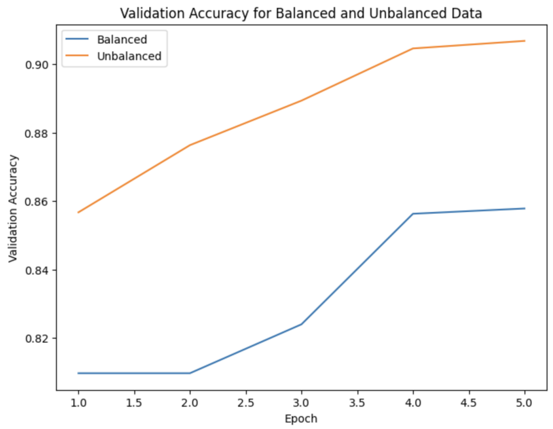

Fig 3. Validation accuracy for balanced vs. unbalanced data

Conclusion and Future Improvements

From the results, it can be concluded that FinBERT has some potential in predicting earnings call surprises. The model achieved a high validation accuracy in both the unbalanced and balanced datasets, with the accuracy reaching up to 90% for the unbalanced data and 86% for the balanced data. The sharp decrease in training loss in subsequent epochs also suggests that the model was able to learn from the data and improve its performance over time. However, there are some limitations and room for improvement that need to be considered.

One major limitation of the model is its sensitivity to imbalanced data. In the unbalanced dataset, where the majority of the observations were labeled as neutral, the model had a tendency to predict neutral even when the actual label was positive or negative. This indicates that the model needs more balanced data to improve its performance on rare events. Another limitation is that the model does not take into account the context and tone of the earnings call, which can be crucial in determining whether it is a positive or negative surprise. Moreover, FinBERT is limited to the English language and may not perform as well on non-English earnings calls.

Another limitation is that although I did have a decent amount of data to work with, the number of companies representing this dataset was slightly small. A smaller number of companies in the dataset could result in limited generalizability of the model to other companies. In other words, the model may not perform as well on data from companies that are not represented in the dataset. This can limit the real-world applicability of the model.

To mitigate this limitation, one possible improvement could be to include data from more companies in the dataset. This would increase the diversity of the data and improve the generalizability of the model.

To improve the model's performance, several steps can be taken. One approach is to use data augmentation techniques such as oversampling, undersampling, or synthetic data generation to balance the dataset. Another possible improvement is to incorporate additional features such as sentiment analysis and topic modeling to capture the context and tone of the earnings call. Additionally, multilingual versions of FinBERT can be developed to enable better performance on non-English earnings calls. Another improvement that can be made is to experiment with different pretraining strategies and fine-tuning techniques to improve the model's generalization capabilities.

In summary, while the results of using FinBERT on earnings call data are promising, there are limitations and opportunities for improvement. Addressing these limitations and incorporating these improvements could help FinBERT achieve better performance in predicting earnings call surprises.

References

[1] Li, W., & Ye, X. (2021). Predicting the market response to corporate earnings calls using FinBERT. International Journal of Information Management, 56, 102222.

[2] Hamid, F., Masud, M. M., & Al Hasan, M. (2021). FinBERT-Cap: Capitalizing financial domain knowledge for earnings surprise prediction. Expert Systems with Applications, 175, 114819.

[3] Shvets, M., & Kotlyarov, I. (2021). Predicting Earnings Surprises: A Machine Learning Approach. arXiv preprint arXiv:2103.13189.