FINA4350 Group Project Report

Group Textonomy
Zhang Wenkai (Leader) (3035637981)
Tang Kai Chi, Zepa (3035806520)
Zhang Wanqian (3036173726)
LORENZ, Marcel (3036174106)
Wu Tsz Kiu (3035801037)

1. Introduction

We are in a new AI era, marked by rapid product launches like GPT-4, Sora, and Suno. This has sparked new interest in AI, as seen by the stock market surge. Nvidia's stock, for example, rose 96.5% from USD 492.44 on January 2nd, 2024, to a record USD 967.66 by March 25th, 2024. However, negative news like copyright issues, reduced model capabilities, and environmental impacts may weaken this enthusiasm. Therefore, it is important to closely monitor the sentiments toward AI conveyed by media. In our group project, we analyze news sentiments using NLP models with the aim of exploring news' impact on AI-related stock prices.

2. Methodology

2.1 Data Collection

2.1.1 Training Set

As the target is sentiment analysis, sentences with sentiment labels are desired training data. Through careful research and selection, three data sources were selected: Financial phrasebank, which mainly contains financial news headings and sentiment scores, Sander, which is a Twitter sentiment corpus, and Taborda, which contains stock market tweets data with sentiment scores.

2.1.2 Web Scraping

We decided to get our sentiment data from the Yahoo search site: https://news.search.yahoo.com. We used a set of AI-related keywords as search parameters, running the scraper separately for each keyword. This way we received a broad set of AI-related news information. We scheduled the scraping to occur daily, so as to always have up-to-date data. The process begins with constructing a dynamic search URL and setting custom HTTP headers, mimicking a browser request stemming from Google. We used Python with the Requests library to send the request to Yahoo, retrieving the HTML content and parsing with Beautiful Soup to locate the news article elements: headline, source, date posted, description, and clean_link (a direct link to the news article). We extracted these using a function that uses regular expressions to clean up redirect URLs.

def get_article(): #Extracts article details such as headline, source, posting time, description, and link from raw HTML. It performs link unquoting and pattern matching to clean the extracted URL.	def get_the_news() #Scrapes AI-related news articles using a set URL template and saves them to a CSV file. It collects articles without duplication, handling pagination and network requests with custom headers.

To ensure there are no duplicates we collected the data with their links in a set. We implemented a brief pause between requests to reduce server load so as to not be categorized as spam. The collected data was stored in csv files with columns for each of the article attributes.

With this setup we facilitate a continuous, up-to-date data collection which we feed into our sentiment model.

2.2 Data Preprocessing

2.2.1 Training set

After selecting the three data sources, our group found there are different file types (txt and csv files), different data structures, different labeling sequences, and different labeling methods, so we did some data preprocessing to tackle these problems: we unified the label format, 0 for negative, 1 for neutral, and 2 for positive. Then we merged all three datasets into one large dataset with [sentence, label] pairs.

The final result "combined_result.txt" consists of 9111 lines of code with the desired format.

2.2.2 Preprocessing during Web Scraping

Our scraped data's 'posted' field had to be converted from a text value ("1 week ago", "34 minutes ago", …) to a Time value. We built a custom Python function to catch all of the possible values where we filtered for the column containing one of the following words: minute, hour, day, week, month, year and applied an appropriate Timedate conversion.
Additionally, we combined all the entries in the csv files of different keywords into a common combined.csv file. We also made sure that after each scraping iteration, duplicate data would not be added to our files.

2.3 Model Training

To achieve our goals, we trained a BERT model to perform text classification and used the trained model to explore the relationship between investors' sentiment and AI stock price. BERT, which stands for Bidirectional Encoder Representations from Transformers, is built based on transformers, a deep learning model that relates each output to every input element by a weighting dynamically computed based on their associations. One advantage of using the BERT is that it is pre-trained using text from BookCorpus and English Wikipedia and can be fine-tuned with question-and-answer data sets, such that our workload is greatly reduced and the model can be customized for the context of AI news in our project.

TensorFlow Hub provides various kinds of BERT which can tackle different needs. For example:

- Small BERT: Fewer Transformer blocks, which allow users to strike a balance between speed, size, and quality.
- ALBERT: Reduces model size by sharing parameters between layers.
- Electra: Shares the identical structure as BERT, but undergoes pre-training as a discriminator within a configuration that mimics a Generative Adversarial Network.

We have selected Small BERT due to its compact size and efficiency. As we only have limited time, Small BERT offers a good balance between model performance and computational resources. It allows for faster training and inference while still capturing essential language representations.

For the fine-tuning process, we utilized the AdamW optimizer, which was originally employed during the training of BERT. This optimizer is responsible for minimizing the loss associated with predictions, while also performing weight decay for regularization purposes. Regarding the learning rate, we will adopt the same schedule employed during BERT pre-training. This schedule involves a linear decay of an initial learning rate, preceded by a linear warm-up phase that spans the first 10% of the training steps. According

to the BERT paper, a smaller initial learning rate is preferable for fine-tuning, with potential options being 5e-5, 3e-5, or 2e-5. Hence, there will be 5 epochs with a learning rate of 3e-5, losses.SparseCategoricalCrossentropy will be used as the loss function as we have more than two classes.

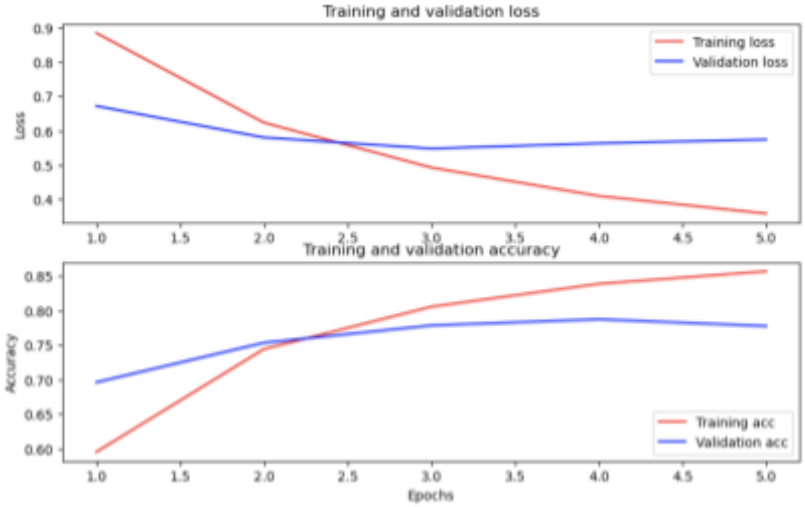


Fig. Training and Validation Loss and Accuracy

The model was optimized at the third epoch, with a validation loss of 0.548 and a validation accuracy of 77.84%. When fitting our model with the test data, the accuracy is 78.47% with a loss of 0.530.

2.4 Result Analysis

Understanding the relationship between sentiment scores and stock price changes is crucial for investors and analysts. Sentiment analysis involves quantifying the emotions and attitudes expressed in textual data, while stock price changes reflect the market's perception of a company's value. In this part, we will outline the process and steps involved in conducting a correlation analysis between AI-related news sentiment scores and AI index price changes.

2.4.1 Variable Selection

To assess the relationship between sentiment scores and stock price changes, we adopted two dependent variables given the T_n average average sentiment. One is the difference between the T_{n+1} opening price and the T_{n-1} closing price. Another is the return rate calculated from the T_{n+1} opening price and the T_{n-1} closing price. The formulas are as follows:

$$Price\ Difference\ =\ Next\ Opening\ Price\ -\ Previous\ Closing\ Price$$
$$Rate\ of\ Change\ =\ (Next\ Opening\ Price\ -\ Previous\ Closing\ Price)\ /\ Previous\ Closing\ Price$$

2.4.2 Regression Analysis

In this step, we perform a linear regression analysis to examine the relationship between the AI-related news sentiment scores (independent variable) and the two dependent variables. The purpose of this analysis is to determine whether sentiment scores have a statistically significant impact on stock price changes.

The regression model equation can be represented as follows:

$$One\ of\ the\ Dependent\ Variables\ =\ \beta 0\ +\ \beta 1\ *\ Sentiment\ Score$$

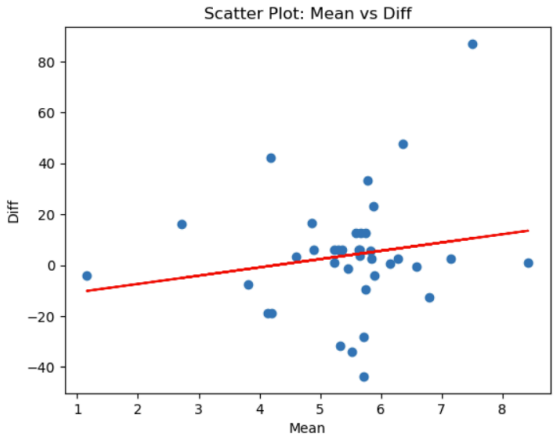
Here, β0 represents the intercept, and β1 represents the coefficient for the sentiment score. By fitting this regression model to the data, we estimate the β0 and β1 coefficients.
To evaluate the strength of the relationship, we calculate the coefficient of determination, also known as R-squared. R-squared measures the proportion of the variance in the dependent variables that can be explained by the sentiment scores. A higher R-squared value indicates a stronger relationship between sentiment scores and stock price changes.

3. Deliverables

3.1 Result Analysis

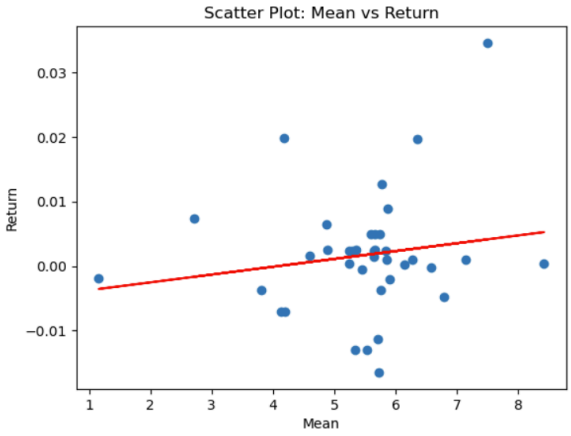
3.1.1 Mean vs Diff

Initially, we attempted to use the difference between the Next Opening Price and the Previous Closing Price, termed 'Diff', as the dependent variable. However, we found that the model did not fit well. Under the guidance of our professor, we realized that 'Mean'—the average sentiment scores of AI-related news—did not effectively reflect the impact on the 'Diff' values. This is because a large 'Diff' value does not necessarily imply a significant impact. It became apparent that using 'Return', which represents the value changes in the AI index price, as the dependent variable was more rational.



3.1.2 Mean vs Return

The scatter plot and linear regression fitting results for the average sentiment scores of AI-related news, 'Mean', and the value changes in the AI index price, 'Return', are as follows.




```
Result:
slope: 0.0012128590448452433
intercept: -0.00496125127361389
R_squared: 0.026127790096458197
```

It can be seen that the model's fitting results are still not particularly ideal. We believe this might be due to several reasons. First, the "posted" time of the data we scraped is not exact. Inaccuracies in the time of the data can lead to biases in the analysis. Secondly, sentiment tends to have a lagging effect on stock prices. This means that changes in stock prices might only become apparent after shifts in news sentiment. Lastly, news sentiment does not fully represent investor sentiment. News reports might only reflect the perspectives of certain segments of the market or specific groups, rather than the overall sentiment of all investors. This discrepancy can lead to a misrepresentation of the actual influence that sentiments have on market movements.

3.2 Live Prediction Pipeline

Although the correlation analysis results were not ideal, we developed a future-ready pipeline that operates as follows: At 7:30 P.M. HKT (7:30 A.M. EST), our data scraper begins collecting AI-related news from Yahoo Finance, deduplicating and integrating it within an hour. Next, the pipeline preprocesses this data, adjusting time zones and appending date details. By 9:00 P.M. HKT (9:00 A.M. EST), it extracts news from the last 24 hours and week, which is then analyzed by our trained Bert model for sentiment. The pipeline outputs daily mean, median, and mode sentiment values for investor reference, providing insights into the immediate and weekly news sentiment trends.

4. Conclusion

Our group project aims to analyze news sentiments related to AI and their relation to stock prices. To achieve this goal, we first researched and selected three pretraining datasets. Then we preprocessed the data and trained a BERT model for sentiment analysis. After that, we applied a model to newly scrapped news data and analyzed the correlation between the sentiment scores and AI sector stock prices. Finally, we developed a pipeline to monitor the overall sentiments of the past 24 hours and the sentiment trend in the past 7 days.

Results suggest that the correlation between news sentiment and AI sector stock prices is not significant for many reasons. However, we believe the pipeline can still offer some insights into the trend of overall sentiments toward AI, which can help investors make more informed decisions.