CMPT 732: Programming for Big Data I
Project report (Fall 2019)

Analyzing News for Stock Market Prediction

StockAnalysisAmigos:

Olawumni Kolawole,
Juan Ospina,
Mohammad J. Eslamibidgoli

(okolawol,jospina,meslamib)@sfu.ca

1 Problem Statement

To better understand the impact of global news on stock market, in this project we applied *Sentiment Analysis* (Turney, 2002) on a large collection of world news headlines, along with the decomposition of seasonal time-series stock data, in order to find possible correlations between the news sentiment and the stock noise. The raw data for news headlines were collected from *Reddit* (World News subreddit). Stock market information and trends were collected from *Yahoo! Finance* for arbitrary number of available companies in various sections including technology, services, financial, consumer goods, healthcare and industrial goods.

We explore possible stock price reaction to news. The rational approach is to find such reactions from the financial news of a specific company. This, however, requires access to the financial news as well as intra-day stock data of the specific company, which is usually not feasible. On the other hand, world news and historical stock data are accessible for analytics. Therefore, our objective is to find the correlation between overall world news sentiment and the noise in the stock price of an arbitrary company. This approach is based on our hypothesis that the local fluctuations in the stock price may be correlated to the overall sentiment of the world news. This correlation can, in general, be low or high depending on the section of the company, its size, location, etc.

For this purpose, a big data and analytics solution is provided that comprises batch and real-time ETL (extract, transform, load) modules to process news sentiment and stocks data sources. We first present our approach to collect news and stock data, followed by developing a scalable machine learning pipeline for text data preprocessing, feature extraction, training and finally real-time implementation of the models for the sentiment prediction of news headlines. Likewise, we show our approach to perform scalable exploratory data analysis (EDA) on the stock data from a large number of selected companies. We then use time-series decomposition to obtain trend, seasonality, and noise components of the stock prices of a specific company, integrate this information with the average sentiment of the news headlines per day and explore the possible correlation between the headline sentiments and stock residuals.

Figure 1 shows the architecture diagram of our application. Central panel shows blocks for stock data and news data collection, preprocessing, exploratory analysis, data integration and ML model training. Green boxes shows the main outputs of the application, which are sentiment prediction on real-time news headlines and exploring correlations between news and stock noise. Side panels show the corresponding name of Python programs developed for this project. In the following sections, we first present the methods that we used to approach this problem. This involves our choices for technology and tool selection and steps taken to create a pipeline for our application. Finally, we discuss results, challenges and possible future directions for this project.

2 Methodology

We used Pandas Datareader for collecting stock data and python http requests library for collecting news data from an API endpoint. We employed Spark Python API (PySpark), Spark ML, Spark Streaming and Dask technologies to scale up our analysis. Pandas and statmodels were also used for time-series decomposition and integration of aggregated data. Sklearn preprocessing was used to standardize the residual of stock data. TextBlob and NLTK were used for processing text data. Word2Vec and Term frequency-inverse document frequency (TF-IDF) in MLlib was used for feature vectorization of news headlines. Matplotlib, Seaborn, WordCloud and statmodels plot function were used to generate various plots. In the following, we briefly explain how each tool or technology were used in this project and why we chose to implement them that way.

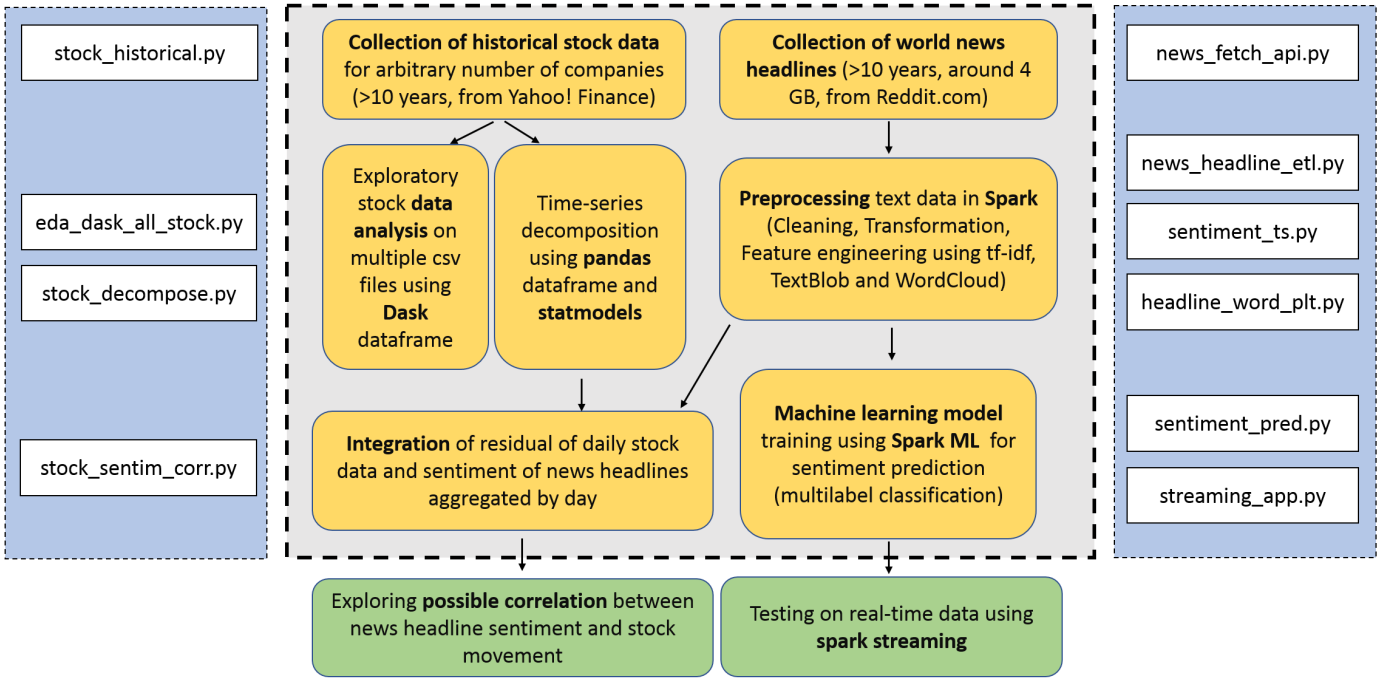

Figure 1: Application architecture diagram

2.1 Dataset creation

The full dataset was built from two sources: Yahoo! Finance and Reddit World News. From these data sources, it was possible to build a unique dataset that was comprised of stock market and news information. For news headlines, Reddit was our preferred choice because it aggregated and gathered news from a variety of different credible sources like the Washington Post, BBC and CNN. Along with this variety, Reddit also provided a robust API with metadata that proved useful in our implementation and analysis. Engagement metrics like the number of comments, creation dates, scores, upvotes and downvotes made it possible to determine news headlines with high impact. In the span of 12 hours, we gathered over three million news headline records in total from 2008 to 2019, making the dataset 4GB in size. In addition, we also setup a streaming application that fetched headlines from different channels. These were used as test data for our sentiment prediction model.

On the stock side, given the tickers for arbitrary number of companies, historical stock market data were collected for over the past 10 years using Pandas Datareader. The data frame have the index set to dates and the columns were:

1. **Open:** the price of the first trade on the given trading day.
2. **High:** the highest price at which a stock traded for the given trading day.
3. **Low:** the lowest price at which a stock traded for the given trading day.
4. **Close:** the price of the final trade before the end of the trading day.
5. **Adj. Close:** according to Yahoo! Finance, this is the "the closing price after adjustments for all applicable splits and dividends distributions".
6. **Volume:** The number of shares traded for the given trading day.

Using stock_historical.py code it took 10 seconds to collect data for 22 companies.

2.2 ETL process

For news headlines, we developed a Python program that incrementally fetched news headlines via the Reddit API from January 2008 to October 2019. On this raw data, the following processes were performed:

1. Remove null / empty entries. Especially in columns needed for future analysis like score, headline title and the created UTC timestamp.

2. Clean the headline string: Remove stopwords present in the NLTK stopword corpus, special characters and punctuations. This will prepare the string for sentiment analysis.

3. Use the LangDetect library to filter out non-English headlines.

4. Convert POSIX epoch timestamp integer into an ISO format date string (YYYY:MM:DD) using the Datetime library. This will make it easier to query specific time periods.

5. Extract metrics for polarity (sentiment orientation: positive, neutral, or negative) and subjectivity (determine if a sentence expresses an opinion or not, and if so, if the opinion is positive or negative) from the cleaned headline string in step 2 with the TextBlob sentiment analysis library. This will be used in our ML pipeline.

For exploratory stock data analysis we used Dask, a Python library to scale up the computations. Dask Dataframe is a delayed version of Pandas Dataframe. It performs lazy evaluation and does not read a file immediately until the *compute()* method is invoked. It can read many files at once using the asterisk wildcard (a glob pattern) and concatenate them into a single Dask Dataframe. For this purpose, the choice of Dask Dataframe makes it very convenient to read and analyze our separated CSV files of stock data for different companies. Moreover, a Dask Dataframe can contain more data that fits into available memory and it has many methods in common with Pandas Dataframe.

As an example of using Dask Dataframes for scalable exploratory stock data analysis, we built a delayed pipeline to determine the total volume of company NKE. We first make a Boolean Dask series comparing the company's name column to the string 'NKE'. Then .loc accessor was used to filter the rows corresponding to NKE and the Volume column from the Dask Dataframe. The result is a Dask series that remains unevaluated until *compute()* is invoked. We then sum up the NKE Volume, using the *sum()* method. As we call *compute()* the result will be a single value.

Dask object has a method called *visualize()* that displays the execution sequence and flow of data in a graph. The plot in Figure 2 shows how Dask uses a variety of heuristic schedulers for a complicated execution sequence. The scheduler automatically assigns tasks in parallel to extra threads or processors. As a result, we do not have to decompose the computations ourselves. We invoke the *compute()* method of the delayed object. The result yields a Pandas Dataframe with a single row. Specifically, Figure 2 shows the directed acyclic task graph summarizing the entire pipeline with the visualize method. It describes flow of data (the rectangles) through functions (the circles) from left to right. All left circles correspond to reading each of the 5 CSV files. The filtering operations are independent, so all work across the 5 main channels can be executed in parallel by the Dask scheduler. The channel coalesce into the rightmost circle where the sum is aggregated to return the final computed value.

In another task, seasonal decompose in statmodels was employed to generate trend, seasonality and residual of the stock price of a given company. We then used StandardScaler from *sklearn.preprocessing* to standardize the residual values by removing the mean and scaling to unit variance; recalling that our objective is to compare the computed residual with the sentiment of the news headlines. Moreover, we calculated the lag of stock price using *shift()* method from which the difference of price with

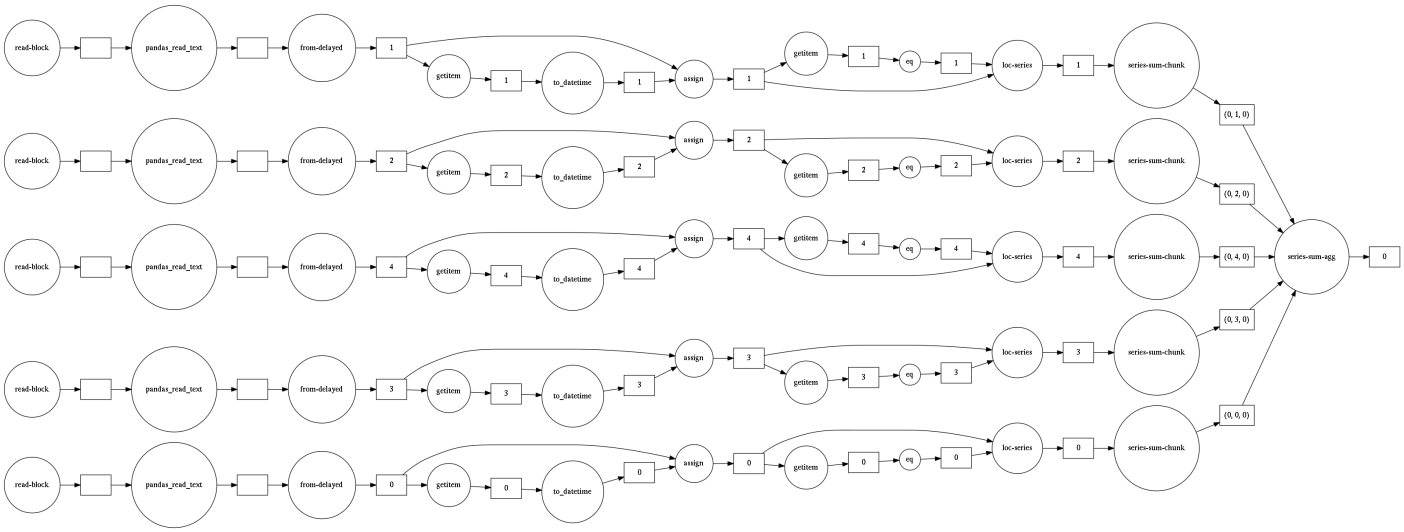

Figure 2: Directed acyclic Dask task graph

previous day was calculated. We consider this difference value as another metric to compare with the sentiment. Finally we merge difference, standardize residual and computed sentiment and calculate the corresponding Pearson product-moment correlation coefficients.

2.3 Machine Learning model

Regarding sentiment prediction, a model was developed based on the following feature engineering process:

- Feature 1: headline as a vector. A tokenizer was used to transform the cleaned headline sentences into a dense vector of individual words in the sentence, using Word2Vec, each tokenized headline was converted into a unique vector of floating point values with size 3. A hyperparameter was chosen due to performance.
- Features 2 and 3: Score and Number of Comments. The headline score and comments are metadata provided by the Reddit API. Headlines with the most engagement tend to have drastic changes in polarity, i.e. very bad or very good news.
- Feature 4: Subjectivity. The subjectivity score was provided by the TextBlob library, higher subjectivity also affects how drastic the polarities become.
- Feature 5: Word Frequency. Word frequency of all headlines is extracted using TF-IDF (Term Frequency - Inverse Document Frequency). We used this process to determine the top 100 important words in the headline corpus A tokenized headline word in this ranking gets weighted higher than the rest and is marked as a substantial influence on polarity. Words like 'trump', 'killed', 'death' or 'attack' for example.

Classifiers: Logistic regression, Random Forest and Decision Tree Our dataset was split into two parts, 75% training set, 25% validation set. Each headline was assigned a target value (i.e. the sentiment polarity we wished to predict) using the TextBlob library. A float between -1 and 1 indicated a polarity spectrum from -1,negative to 1, positive but to further aid our candidate classifiers we decided to remap these values into 3 classes via a chosen threshold. We used a threshold of 0.05 to ensure the labels are balanced within the dataset: 0=negative, 1=neutral, 2=positive.

TextBlob polarity	remapping
polarity greater than 0.05	2
polarity less than -0.05	0
else	1

In order to test our sentiment prediction model, we utilized Spark Streaming to fetch new headlines in real time before applying our ML pipeline to it.

3 Results and Discussion

Figure 3 shows the calculated average polarity (aggregated by day) of news headlines using TextBlob, along with the generated WordClouds for various years. Interestingly, we can observe some trends and seasonality in the polarity time-series and it does not look stationary. Moreover, from WordClouds we can identify some trends. For example, in 2013 the presence of North Korea and Russia related headlines dragged the sentiment on a downward trend. Mentions of countries in news headlines used to be very dominant in the past, but in 2019, one male name eclipsed them all. Between 2016 and 2019, the polarity of headlines were more spread out into the negative zone.

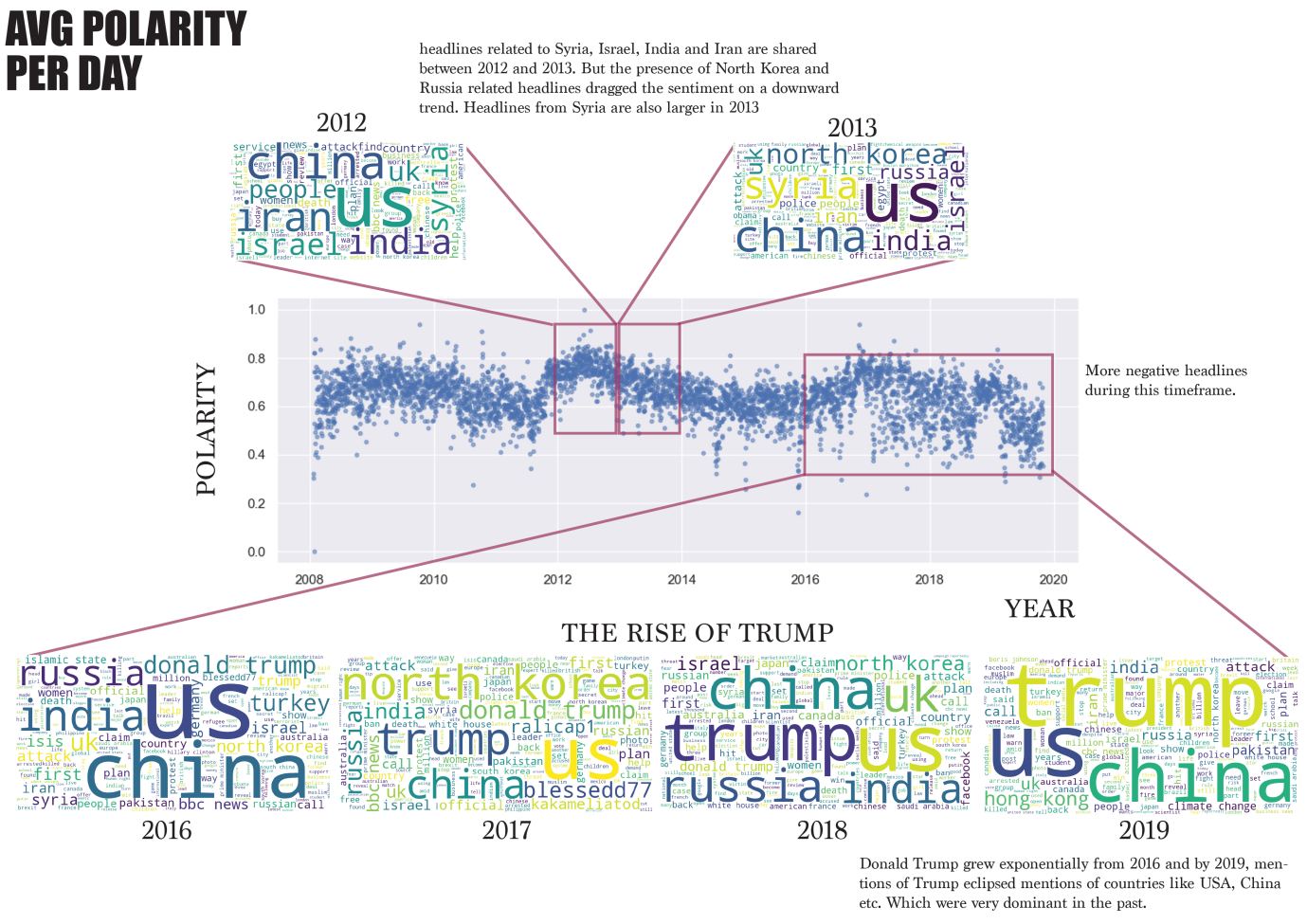

Figure 3: Calculated average polarity (aggregated by day) of news headlines using TextBlob, along with the generated WordClouds for various years. (Matplotlib and WordCloud were used for visualization.)

Our best performing classifier The mapping made it possible for ML classifier training. The following table shows the performance of each classifiers on the validation set in predicting the news sentiment.

The hyper parameter tuning on max_depth for decision tree and random forest has paramount impact on the performance. We evaluated the model based on f1 metric as well as accuracy metric. As can be seen Decision tree results in the best prediction.

Classifier	**f1 metric**	**accuracy metric**
Decision Tree	0.79824	0.801855
Logistic Regression	0.743459	0.760898
Random Forest	0.543525	0.653316

For stock data, in another case of using Dask for the EDA process, we determined the average open price of each 22 companies per month of the year. For this purpose, we created a column that extracts the month-and-year from the Date column, then group by company and month-and-year and compute the mean of open price. The result is shown in Figure 4 (a). Moreover, we created some bar plots from min, max and mean values of open stock of each company by year, which are included in our code *eda_dask_al_stock.py*

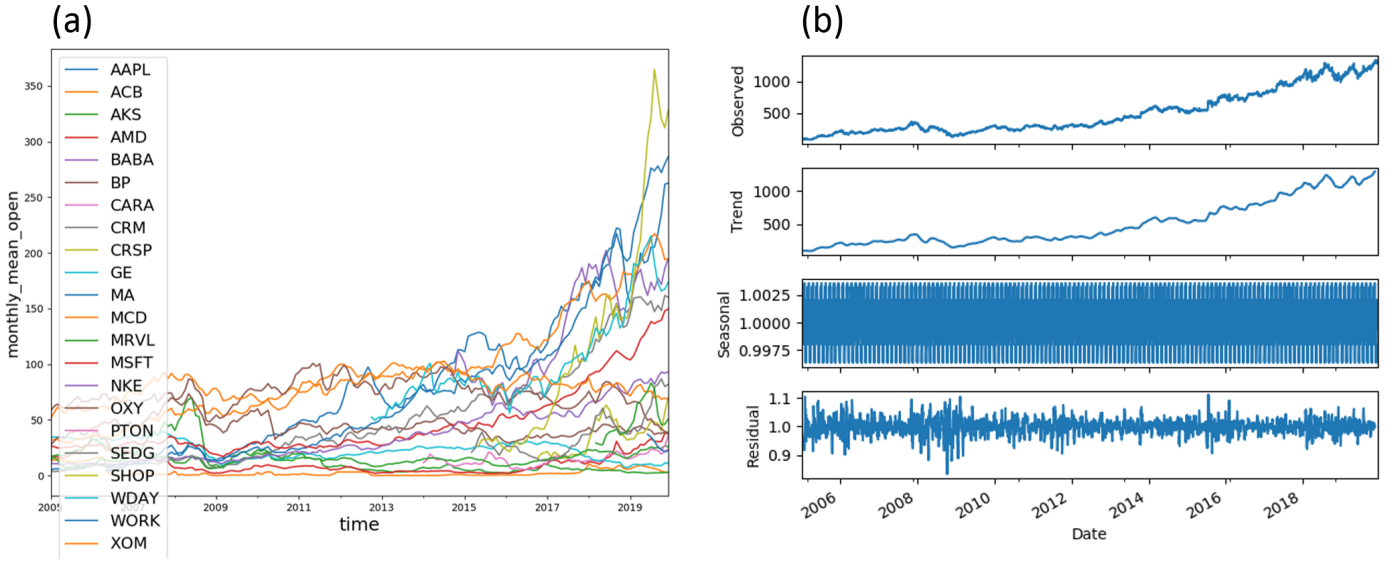

Figure 4: (a) Demonstration of scalable capabilities of our model for exploratory stock analysis using Dask dataframe. This example shows monthly average open price of 22 selected companies. (b) Decomposition of stock to trend, seasonality and residual using statmodel Python module. Matplotlib and decomposition.plot in statmodels were used to create (a) and (b), respectively.

Figure 4 (b) shows the decomposition of observed data for NKE company into trend, seasonality and residual. It can be seen that the residual time-series is stationary and it is centered around 1 for this company. In order to compare with the standardized sentiment, we standardized the residual value. In addition to standardized residual, we also calculated the difference between the price of each day with previous day in order to compare with the average sentiment. Figure 5 shows the joint plots of calculated difference and residual of stock prices of OXY, MSFT and NKE companies vs. standardized average sentiment of all news headlines. Calculated Pearson correlation coefficients are included for each plot. As can be seen, the correlation coefficients are very small suggesting that the current version of our model does not capture the relation between the news headlines and stock noise. In the final section of this report we discuss possible directions to improve the predictability of our model.

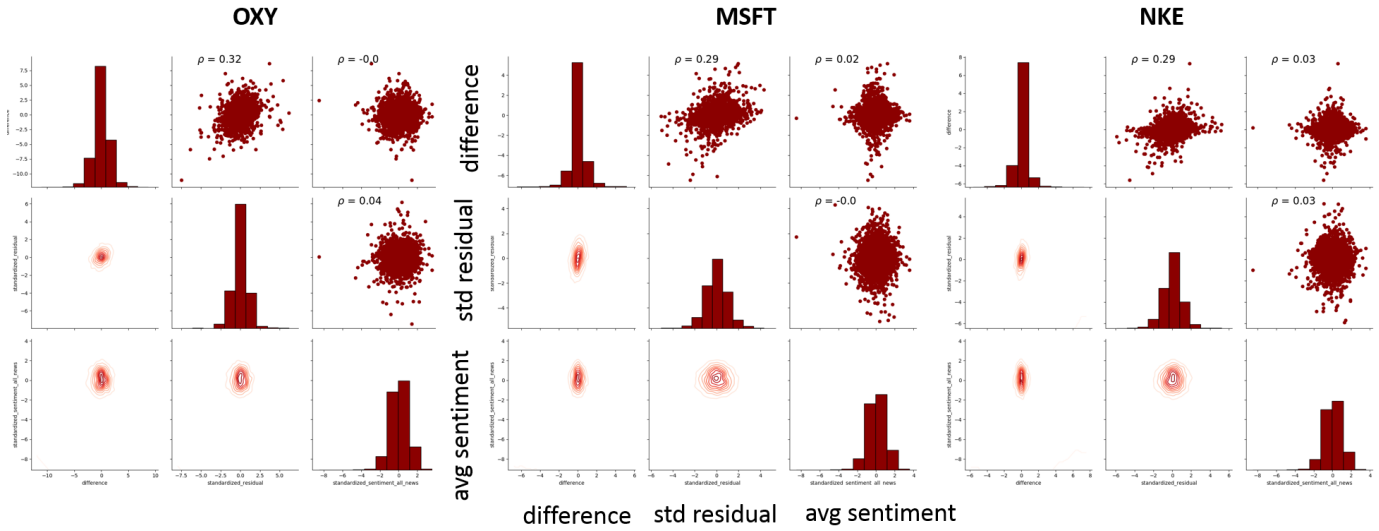

Figure 5: Joint plots of calculated lag difference and residual of stock prices of OXY, MSFT and NKE companies vs. standardized average sentiment of all news headlines. Calculated Pearson correlation coefficients are included for each plot (Seaborn was used to create this plot)

4 Challenges

In a first approximation, sentiment analysis for news headlines was performed using VADER. We performed two attempts. First attempt: (a) The sentiment value was established using a float value instead of polarity. (b) Residual from stock price information is replaced by the news headlines sentiments. (c) Standardization was applied. (d) Detrending procedure performed over data. (e) A regression model was used to predict stock price based on sentiment and seasonality. Result: No correlation.

Second attempt: Processing news headlines with high scores only: top 25 per day. Result: No strong correlation, however, less outliers were detected.

Next, we attempted a second approximation: (1) extract as many features available in the dataset. (2) Train the ML model. (3) Apply feature engineering to predict stock price accurately. Sentiment analysis was just one of the methods applied to extract features. Features: news score, news source. (4) Use Spark Streaming to validate the model. (5) Generate word plots of popular words in headlines.

During the ETL process of our news headlines. A large number of native Python libraries were used for extracting datetime, cleaning sentences with stopwords, getting sentiment scores and detecting English language for filtering.

These dataset specific tasks could not be achieved with Spark SQL functions or vectorized UDFs and as a result, our extensive use of Python UDFs on 3 million records increased extraction time to about 6 hours. We alleviated this issue by utilizing a high level of parallelization which cut extraction runtime down to 2 hours. We also trained another ML model for sentiment prediction so that we would not have to use UDFs to get sentiment scores on higher scaled data.

Another problem we encountered was related to Spark Streaming and realtime model prediction. Due to the nature of streaming and lazy evaluation, empty dataframes were being sent into our ML pipeline.

This phenomenon resulted in generated errors because our feature assembler tried to access non existent

columns. It was solved by adding vector size hints to our pipeline. This metadata informed our feature assembler about what feature vector sizes to expect without having to query an empty dataframe for them.

5 Summary and Path Ahead

We presented our scalable application to analyze news headlines for finding correlations between world news sentiment and the noise term in the stock price. For this purpose, we employed big data processing frameworks such as Spark and Dask, natural language processing tools such as NLTK, Word2Vec and Textblob, statmodels - the statistical python module - for time-series decomposition as well as sklearn, Pandas, NumPy libraries. For visualization purposes, we employed matplotlib, seaborn, WordCloud, Dask visualize, and statmodels plot function. Three machine learning models were trained in order to predict the sentiment of news headlines namely Decision Tree, Logistic Regression and Random Forest. Among them the best accuracy of around 80.19% was obtained for the Decision Tree classifier on the validation set. For testing the classifier, we developed a streaming application in Spark that received fresh headlines from recent times and applied our finalized ml pipeline to it. In the future, this model will move us away from using UDFs and all sentiment score assignments will be done purely within spark which will help with scalability.

Based on our hypothesis that the noise in stock price should be correlated with the news sentiment, we calculated the correlation coefficient between predicted sentiment and the standardized noise of the stock price. However, current version of the model does not capture this correlation. Therefore, there is room for improvement in this model. One way is to include more information about the company through feature engineering and filter the news headlines that are related to those features. For example, the news headlines can be filtered based on the country where the company is located in and only the sentiment of those headlines can be accounted for in taking the average sentiment.

In addition, we can observe seasonality and trend in the sentiment over the years. These long term dependencies in sentiment were not captured in our model that can have an impact on the trend and seasonality of the stock prices of companies. Moreover, as an intermediate step we can check the correlation between the financial news headlines of the company with the world news and develop and intermediate model to capture these correlations.

6 Acknowledgement

This project was enabled by accessing computational resources at gateway.sfucloud.ca as well as Cedar cluster of Compute Canada - Calcul Canada (www.computecanada.ca).

7 Project Summary Evaluation

Item	Points
Getting the data	3
ETL	3.5
Problem	2.5
Algorithmic work	3.5
Bigness/parallelization	2.5
UI	0
Visualization	2.5
Technologies	2.5
Total	**20**