Forecasting Stock Price Changes Using Natural Language Processing

Abstract

All investors attempt to predict stock market returns when they make investment decisions. However, making such predictions is not a trivial task. As a result, many strategies have been proposed by researchers as potential ways to predict stock returns. More recently, data analytics - in general - and natural language processing, in particular, have been identified as viable options. In this project, we investigate the use of natural language processing to forecast stock price changes. Specifically, we analyze firms' 10-K and 10-Q reports to identify sentiment. Using the computed sentiment scores, we develop models to predict the direction of stock price movements both in the short run and in the long run. Our first step in developing these models is to investigate some sentiment scoring methods and apply them to the Loughran-McDonald dictionary. Next, we use the model word2vec to extend the usage of the Loughran-McDonald dictionary and then apply the sentiment metrics. Additionally, we apply the proposed sentiment metrics to FinBERT, which learns contextual relations between words. Finally, we build supervised machine learning algorithms that use the proposed sentiments as inputs to forecast price changes. We train our algorithms on 10-K and 10-Q reports of 48 companies in the S&P 500 from 2013 to 2017. Finally, we test our models on the corresponding reports from 2018 to 2019 and conclude that predictive signals can be extracted from 10-K and 10-Q reports.

List of Acronyms

NLP Natural Language Processing ML Machine Learning
LM Loughran-McDonald
GB Gradient Boosting
RF Random Forest

Contents

List of Figures

List of Tables

Introduction

Forecasting stock price changes is an important problem that many investors and financial institutions are constantly trying to solve. The ability to forecast stock price changes can be effective for risk management, portfolio diversification, and other investment decisions. However, forecasting stock returns is not trivial. Market returns depend on stock price movements, which themselves are extremely volatile and influenced by diverse factors including microeconomic factors, international events, and human behavior events [1]. Researchers have proposed several methods to forecast price changes. These include time-series based methods, stochastic models, and other more mathematically grounded approaches. More recently, data analytics, in general, and natural language processing, in particular, have been identified as promising

options. The recent attention that this area is gaining is not a coincidence. The advances of computing power and the increased availability of both structured and unstructured data, including text, have played a significant role in investors leaning more on these techniques to determine if the data holds predictive power for stock price changes and returns.

1.1 Project Objectives

The primary objective of this project is to forecast price changes based on 10-K and 10-Q reports. Specifically, the problem statement is: “Given a 10-K and/or 10-Q, predict the direction of price movements in the next time period using sentiment scores developed from the 10-K and/or 10-Q”. Consequently, the project entails the following sub-tasks:

- Investigate some sentiment scoring metrics that are applicable to 10-K and 10-Q re ports.
- Apply the metrics to the Loughran-McDonald dictionary, which is widely used in finance.
- Explore sentiment generation strategies that are not restricted to individual words. • Investigate the possibility of applying machine learning to relevant sentiment features. • Build a portfolio based on the sentiment scores to analyze the performance.

1.2 Related works

Researchers have proposed a number of methods aimed at using language features to predict market variables. In [2], Kogan *et al* applied NLP to 10-K reports to predict the volatility of stock returns. In [3], Engelberg used typed dependency parsing, a tool from natural language processing, to demonstrate that qualitative information relating to positive fundamentals and future performance is the most difficult information to process. In [4], Xie *et al* showed that features derived from semantic frames parsing have significantly better performance across years on polarity tasks than a change of price task. The authors of [5] forecasted companies’ stock price in response to financial events in 8-K documents and reported a prediction accuracy in excess of 10% over a strong baseline that incorporates many financially-rooted features.

Methodology

In this chapter, we present the background and theory of the methods implemented in this project. First, we discuss the sentiment scoring metrics that were used. Second, we introduce the Loughran-McDonald dictionary (LM), a popular dictionary for financial sentiment analysis. Third, we explain word2vec as a way to expand on the LM dictionary for sentiment analysis. Fourth, we present the FinBERT model to classify sentences and account for context. Lastly, we introduce three machine learning algorithms that we employ to incorporate all of the above approaches.

2.1 Sentiment Scoring

The key aspect of sentiment analysis is to explore the input text to understand the expressed tone. The overall sentiment of a word, sentence or text can be classified into different categories. For example, the LM dictionary classifies words into positive, negative, neutral, uncertain, litigious, strong modal, weak modal, and constraining. Once the sentiment classification for each word has been obtained, the next task is to quantify the sentiment of the entire text.

In the literature, there are several methods for analyzing sentiment. These include:

1. Polarity: this measure aims to quantify the precision or intensity of the tone. In this project we want to quantify how positive the tone is, therefore we define polarity for words (and later for sentences) as:

$$\text{polarity} = \frac{(\#positive - \#negative)}{(\#positive + \#negative)} \quad (2.1)$$

2. Percentage Positivity: this measure also aims to quantify the intensity of the tone, but instead of only considering the positive and negative words (or sentences), it takes into account all words (or sentences) present in the corpus. Mathematically we define the percentage positivity for words (and later for sentences) as:

$$\text{percentage positivity} = \frac{(\#positive - \#negative)}{(\#total)} \quad (2.2)$$

2.2 Loughran-McDonald Dictionary

The LM dictionary is a sentiment dictionary developed by Tim Loughran and Bill McDonald specifically for financial texts. It classifies words into one of six of the following categories: positive, negative, uncertain, litigious, strong modal, weak modal, and constraining.

The goal of the LM dictionary is to classify words that have a specific sentiment (for example, negative) in financial filings. More specifically, in their research [6] Loughran and McDonald show that the Harvard “Psycho-sociological Dictionary”, which is commonly used in NLP applications, classifies words that are typically not negative in finance as negative - for example, capital, vice or board. The LM dictionary reclassifies these words and also finds other frequent words that are not classified by the Harvard dictionary. It was initially created by analyzing 10-K and 10-K405 reports from 1994-2008. 50,115 reports of 8,341 firms were analyzed. The LM dictionary is updated almost every two years.

2.3 Extended LM Dictionary and word2vec

One of the challenges of using the LM dictionary is that the number of classified words is very low compared to the number of words in the reports. This challenge can be addressed in several ways. We next discuss two commonly used methods. The first one is word embeddings and the second one is word2vec models.

2.3.1 Word Embeddings

The objective of word embeddings is to transform words into vectors, with the idea that similar words will end up close to each other in the feature space. One way of embedding is to use one hot encoding. For example, consider the sentence: “The results of the last trial are promising.” The one-hot encoding matrix of this sentence is as follows:

Word	1 2 3 4 5 6 7
The results of the last trial are promisin g	1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

One problem of this approach is that by matching each unique word into a real continuous m-dimensional space, the dimensionality may end up being very large, which requires a lot of computer memory and space. According to [7] even after using dimensionality reduction and smoothing techniques, the resulting matrix can be sparse, which may make the task of extracting meaning computationally expensive.

Another approach that can be used is word2vec models (CBOW and Skip-gram), which use local information only. Rong [8] offers a detailed explanation on how word embedding models, such as word2vec, are trained. We summarize the high-level idea in the following section.

2.3.2 Word2vec skip-gram model

Word2vec

Word2vec models are neural network models. Neural networks are models that mimic the workings of the human brain. They recognize underlying relationships in the data set and adapt to changing inputs. A neural network is made of layers, called nodes. A node combines input from the data with a set of weights that amplify or dampen the input. The input weights are summed and passed to an activation function that determines whether the signal should progress further through the network. If the signal is activated we get an output [9].

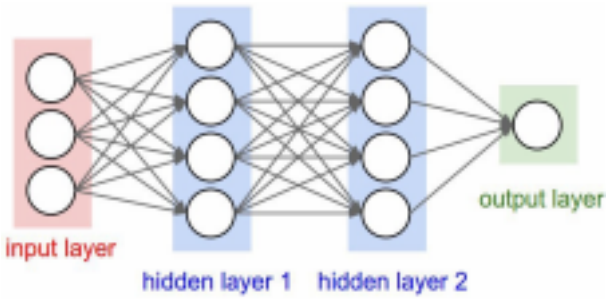

Figure 2.1: Neural Network, Image obtained from [10]

Word2vec models are two-layer neural networks, whose goal is to reconstruct the linguistic context of words [11]. A word2vec model trains a neural network with a hidden layer to reduce a loss function by adjusting the weights of the word vectors. In other words, it takes a large input vector (the one-hot encoding matrix introduced in Section 2.3.1) and compresses it to a smaller vector. What makes these models unique is that they are not used for the task they were trained on, instead they are used to obtain the weights from the hidden layer, which are then used as the word embeddings (word vectors). What this means is that instead of decompressing the vector back to the original input form, we output the probabilities of the target words. In short, word2vec takes a text corpus as input and produces word vectors as outputs. To find similar words we look for words that have similar vector representations. For a detailed mathematical description on how these hidden layers and loss functions are defined, the reader can refer to [8]. The following graph summarizes the steps to train a word2vec model. The process starts with the input corpus, followed by the vocabulary and context builder. Then the right path refers to the Skip-gram model and the left path refers to the Continuous Bag of Words model.

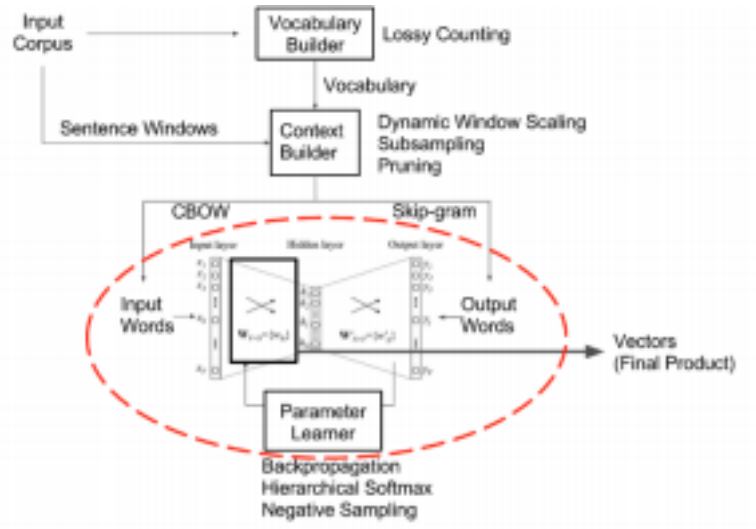

Figure 2.2: word2vec model diagram. Image obtained from [12]

Skip-gram

A skip-gram is one kind of word2vec model. It trains a model so that given a specific word in the middle of a sentence, it examines and picks a nearby word at random. More specifically, the target word acts as the input layer and the context words are the output layer. Given a set of sentences, in our case from a 10-K or 10-Q report, a skip-gram model loops over all the words and tries to use the current word to predict its “neighbors” (the words that surround it within a given window size). The main idea of a skip-gram model is that given a word, $w(t)$, the model can calculate the probability of each word of the vocabulary being near the chosen word, $w(t-2)$, $w(t-1)$, $w(t+1)$, $w(t+2)$. Figure 2.3 illustrates the input and outputs of the model.

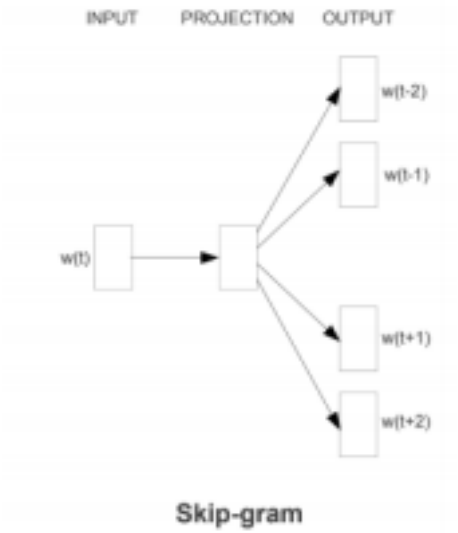

Figure 2.3: Skipgram model, image obtained from [13]

2.4 FinBERT

Another drawback of the LM dictionary is that it does not take the context surrounding the words into account. One way to address this is to use a model called FinBERT. It learns contextual relations between words and is one of the most recent models researched in the field of NLP. FinBERT is a BERT model trained on a corpus of financial texts. The BERT language model is based on a machine learning architecture known as the transformer, which has an encoder-decoder architecture. In this section, we first explain the building blocks of the FinBERT model, followed by the model itself.

2.4.1 Transformer: Encoder and Decoder

A transformer is the most important part of the BERT model on which the FinBERT model is built. It consists of two architectures:

- The encoder has layers that process input data iteratively, one layer at a time. Each encoding layer processes its input to encode information about which parts of the input data are relevant to each other (for example, the subject, verb, and object of a sentence). After this, the output, which is made up of encodings from the previous layer, is passed on to the next encoding layer as input data, and the same sequence of processing repeats for each layer in the encoder.
- The decoder has layers that process the data iteratively, one layer at a time. Each decoder layer processes the encodings, from the encoder, by using the incorporated contextual information and generating an output sequence.

2.4.2 BERT

BERT is a model introduced by a team at Google, and was published in the paper “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. BERT is an acronym for “Bidirectional Encoder Representations from Transformers” [14].

The key technological innovation of BERT is to apply the two-way training of the transformer to language modeling. Previous models are trained to view text sequences from left to right or a combination of left to right and right to left. The Google paper shows that the two-way trained language model has a deeper language background than the one-way language model.

The pre-training tasks of the BERT model are the masked language model and the next sentence prediction model. The masked language model “masks” a word in it and predicts possible words based off of the words surrounding it. The next sentence prediction model determines whether a particular sentence B follows sentence A in the original text corpus.

The advantage of BERT is that it considers the context of a particular word.

2.4.3 FinBERT model

FinBERT is a BERT model pre-trained on financial communication text. One of the goals of FinBERT is to analyze the sentiment of financial text, which as mentioned before, may use language differently than in other domains. FinBERT is built by further training the BERT model, introduced in Section 2.4.2, on financial text corpora size of 4.9B tokens [15].

Training Corpus

The pre-training corpus used by FinBERT mainly includes three types of financial corpus, as follows:

- Corporate Reports, namely 10-K and 10-Q reports: 2.5B tokens
- Earnings Call Transcripts: 1.3B tokens
- Analyst Reports: 1.1B tokens

2.5 Machine Learning

Processing text as data for statistical analysis can be a high-dimensional problem. Since machine learning excels in handling high-dimensional data, we investigate three supervised machine learning models. More specifically, we employ the following machine learning classifiers in this project: logistic regression, random forest and XGBoost. A brief description of each classifier is outlined below:

- Logistic regression uses the logistic function $((1+e^{\beta \cdot x})^{-1})$ to model a binary

dependent variable and therefore can be used as a classifier. The assumptions of logistic regression include a binary dependent label, independent observations, little to no multicollinearity, and a large sample size. The regression coefficients in logistic regression are estimated using maximum likelihood estimation (MLE) rather than ordinary least squares (OLS). In addition, logistic regression does not require a linear relationship between features and labels, and can therefore be used to model non-linear relationships between features and labels.

- Random forest classifier is a meta estimator which fits a number of decision tree classifiers on sub-samples of the dataset. One of the reasons for using random forest is because decision trees are known to overfit, and therefore using random forest allows for ensemble forecasts to be produced with lower variance. Random forest also shares similarities with another machine learning technique known as bagging, in the sense that it independently trains individual estimators over bootstrapped subsets of the data. In addition, when optimizing each node split within a tree, only a random subsample (without replacement) of the features is evaluated, with the purpose of de-correlating the estimators. Hence, random forest classifiers are able to lower forecasts' variance, calculate feature importance on-the-fly, and provide out-of-bag accuracy estimates [16].

- XGBoost is an advanced implementation of the gradient boosting algorithm, which is an ensemble machine learning model where the ensembles consist of weak prediction models, such as decision trees. It provides parallel tree boosting, which is more efficient, flexible, and faster for large data intensive computations.

Experiments and Results

We present our approach to the project, show the results of our experiments, and discuss our results. Our implementation is as follows:

i. Scrape 10-K and 10-Q reports for 48 chosen companies from EDGAR ii.

Generate sentiment features from the reports using LM, word2vec, and FinBERT

iii. Generate returns associated with each report of the selected companies iv.

Evaluate the performance of the sentiment metrics

v. Build an investment portfolio to test the predictive power of the models

In the following sections, we discuss each of the steps in this approach.

3.1 Scraping of data

The data we use in this project comprises 10-K and 10-Q reports. The 10-K is an annual report filed by a publicly-traded company and includes its history, organizational structure, financial statements, earnings per share, subsidiaries, executive compensation, and any other relevant data [17]. The 10-Q on the other hand is a quarterly report that shows the company's financial statements, management discussion and analysis, disclosures, and internal controls.

Obtaining this data consists of three steps, which are:

- Web scraping: In this step, we retrieve the 10-K and 10-Q reports from the SEC EDGAR (Electronic Data Gathering, Analysis, and Retrieval system) website. This involves using URL requests to connect to the SEC website to download the 10-K and 10-Q reports for a specific company as represented by its CIK number.
- Text cleaning: Distill the required text from the scraped HTML files by removing tables and HTML tags using Beautiful Soup and regular expressions.
- Management section extraction: From the cleaned text files, extract the management section using string slicing. This management section corresponds to:
  - "Item 7. Management's discussion and analysis of financial condition and results of operations" for 10-K reports
  - "Item 2. Management's discussion and analysis of financial condition and results of operations" for 10-Q reports

It should be noted that we focus on the management section of the reports because it includes the company's updated business results from the previous term, and it is where the company gets to tell its story in its own words.

We choose between 8-10 companies from each of the following sectors: Financials, Communications, Information Technology, Consumer Discretionary, and Health Care. The chosen companies were:

- Financials
  - JPMorgan Chase (JPM)
  - SVB Financial Group (SIVB)
  - Citizens Financial Group (CFG)
  - Citigroup (C)
  - Allstate (ALL)
- Communications
  - Creative Technology (CTL)
  - Interpublic Group of Companies (IPG)
  - ViacomCBS (VIAC)

– Netflix (NFLX)

• Information Technology

– AMD (Advanced Micro Devices)
– Intel (INTC)
– Apple (AAPL)
– Lam Research (LRCX)
– Microsoft (MSFT)

• Consumer Discretionary

– Principal (PFG)
– Cboe Global Markets (CBOE)

– Charter Communications (CHTR)
– Facebook (FB)
– Twitter (TWTR)
– News Corp (NWSA)
– Fox Corporation (FOXA)

– NortonLifeLock (NLOK)
– Cognizant (CTSH)
– Alliance Data (ADS)
– Western Union (WU)
– Paypal (PAYC)
– Target Corporation (TGT) – Ford Motor Company (F)
– McDonald's (MCD)
– Whirlpool Corporation (WHR) – The Home Depot (HD)

• Health Care

– Abbott Laboratories (ABT) – CVS Health (CVS)
– Pfizer (PFE)
– Johnson and Johnson (JNJ)
– Biogen (BIIB)
– Best Buy (BBY)
– Amazon (AMZN)
– Ebay (EBAY)
– Booking Holdings Inc. (BKNG)

– Invesco (IVZ)
– E*TRADE (ETFC)
– MetLife (MET)

– Waters Corp (WAT)
– Align Technology (ALGN) –
Edwards Lifesciences (EW)

– Incyte Corporation (INCY) –
Henry Schein (HSIC)

We collect all 10-K and 10-Q reports from these companies between the years of 2013 and 2019. We decide to store words and sentences from the cleaned text reports separately, since sentences are important inputs for the word2vec and FinBERT analyses.

After scraping the data we note some unique characteristics in each sector. Figure 3.1 and Figure 3.2 show the average number of sentences for different sectors and average length of sentences, respectively. It can be seen that the communication and financial sectors tend to have longer management sections, on average. However, the consumer discretionary and health care sectors have longer sentences. Longer sentences can be viewed as proxies of language complexity.

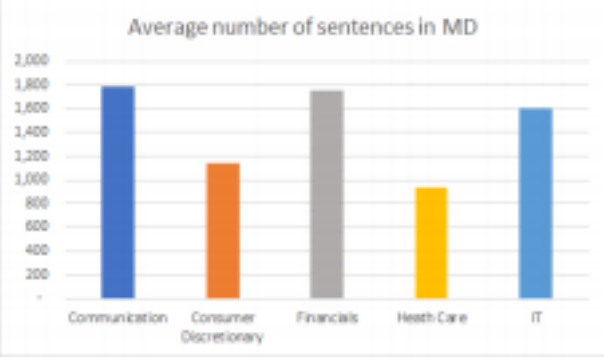

Figure 3.1: Average number of Sentences for different sectors

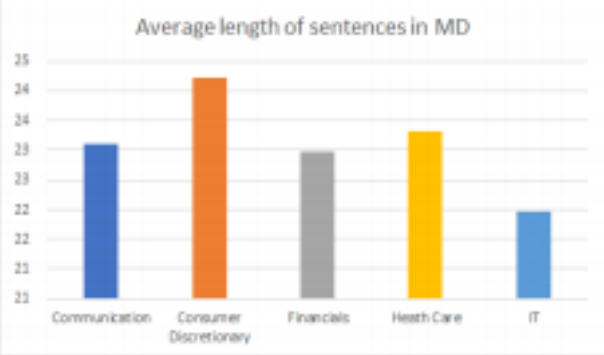

Figure 3.2: Average length of Sentences for different sectors

3.2 Response Variable

Since the objective of this project is to predict whether the stock price of a company will increase or decrease, the response variable is the stock return. We calculate the returns between 1 day before the release date of report and 3, 5, 30, and 60 days after. Returns are adjusted by the S&P 500 returns for the corresponding dates. We calculate the adjusted returns for each one of the 1,095 collected reports. We then label the returns in the following way:

- +1 when excess return > 1%
- −1 when excess return < −1%

3.3 Forecasting Returns Using Sentiment Features

In this section, we apply the sentiment scoring metrics discussed in Section 2.1 to the LM dictionary, Extended LM and FinBERT. To adapt the discussion in Section 2.1 to our specific application, we binarize the computed sentiment score. The binarization process proceeds as follows. First, we split the 1095 observations into train and test data. The train data is from 2013 to 2017 (a total of 853 reports) while the test data is from 2018 to 2019 (242 reports). For each sentiment feature in the train set, we compute the mean, μ, and standard deviation, σ. Consequently, for any sentiment score, x, the binary equivalent is given by

$$f(x) = \begin{cases} -1, \text{ if } x \leq \mu - \sigma \\ 1, \text{ if } x \geq \mu + \sigma \end{cases}$$

By performing this transformation, we eliminate observations that may lead to weak sentiments. Figure 3.3 illustrates this scenario. In this figure, only values that are below the lower bound and above the upper bound are candidates for signal generation.

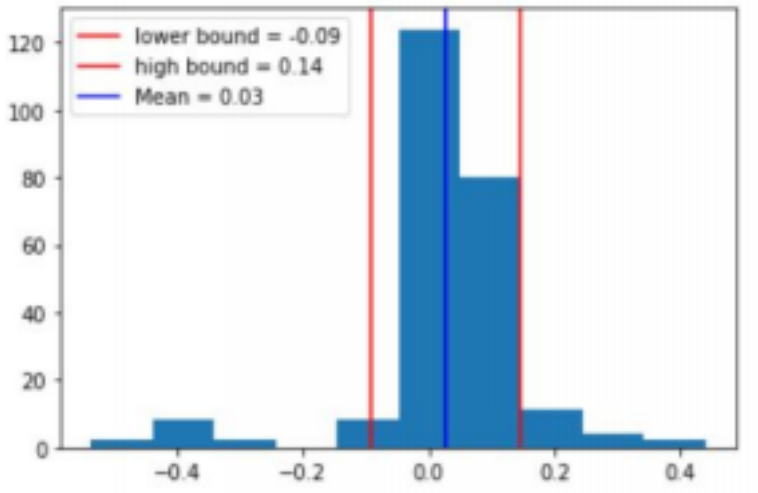

Figure 3.3: Sentiment Histogram

3.3.1 LM Dictionary

For each report we remove stop words, punctuation marks, and we lemmatize all nouns and verbs, for example, the word 'changes' is lemmatized to 'change'. In addition, we also lemmatize the words of the LM dictionary to ensure that the words collected from the 10-K and 10-Q reports and the words in the LM dictionary are comparable.

Figure 3.4 shows an example of the dataset we produce after matching words in Abbott's 10-K and 10-Q reports with the words in the LM dictionary for the negative and positive categories.

Year	Ticker	Sector	File	Date	Positive Words	Negative Words	Total Words	3-Day adj	5-Day adj	30-Day adj	60-Day adj
2013	ABT	Health	10-K	2013-02-15	122	144	9558	-0.283	0.101	-5.593	1.800
2014	ABT	Health	10-K	2014-02-21	109	139	8533	-0.349	0.837	-2.295	-2.472
2015	ABT	Health	10-K	2015-02-27	100	148	10080	-1.134	-0.682	0.048	0.279
2016	ABT	Health	10-K	2016-02-19	110	125	10301	0.776	1.699	-0.580	5.620
2017	ABT	Health	10-K	2017-02-17	112	155	11666	1.059	1.168	0.646	-0.927
2018	ABT	Health	10-K	2018-02-16	126	153	12558	-0.092	0.237	5.308	1.896
2019	ABT	Health	10-K	2019-02-22	123	125	11592	1.524	2.151	3.266	-4.031
2014	ABT	Health	10-Q	2014-05-07	24	30	1952	1.208	0.240	-0.565	1.709
2014	ABT	Health	10-Q	2014-08-05	30	29	2491	-1.270	-0.695	-1.822	-1.993
2014	ABT	Health	10-Q	2014-11-06	34	38	3066	0.259	0.555	2.286	3.011
2015	ABT	Health	10-Q	2015-05-06	25	36	3213	0.135	0.265	3.254	7.159
2015	ABT	Health	10-Q	2015-08-04	24	44	3834	0.203	0.024	-5.592	-15.380
2015	ABT	Health	10-Q	2015-11-05	24	45	3909	0.841	2.263	2.661	-0.716
2016	ABT	Health	10-Q	2016-05-04	37	47	3527	-2.186	-1.252	-0.099	4.152
2016	ABT	Health	10-Q	2016-11-03	41	64	4280	1.397	0.936	-6.171	-6.717
2017	ABT	Health	10-Q	2017-05-03	41	42	3824	2.214	2.134	4.648	10.138
2017	ABT	Health	10-Q	2017-08-02	45	43	4540	-0.492	0.121	4.122	7.086
2017	ABT	Health	10-Q	2017-11-02	47	48	4842	1.414	1.711	-0.484	4.356
2018	ABT	Health	10-Q	2018-05-02	46	36	3622	-0.832	0.178	3.097	0.998
2018	ABT	Health	10-Q	2018-08-01	44	37	3905	-2.718	-1.669	-1.043	9.246
2018	ABT	Health	10-Q	2018-10-31	45	41	4292	1.638	1.439	6.007	12.920
2019	ABT	Health	10-Q	2019-05-01	19	36	2571	-1.234	-0.162	2.267	5.569

Figure 3.4: LM classification for Abott

Table 3.1 shows the accuracy scores obtained when the sentiment features (percentage positivity and polarity) generated from the LM dictionary are used to forecast price changes. These accuracy scores are obtained by counting the number of times the label for the return matches the binarized score for the sentiment feature and dividing by the total number of reports. We notice an average accuracy score of slightly above 50% in all time periods except for the 60 days returns.

Approach	3 days return(%)	5 days return(%)	30 days return(%)	60 days return(%)
Polarity	56.25	54.10	54.24	46.88
Positivity	55.17	52.73	55.77	47.37

Table 3.1: Accuracy of LM dictionary

One problem we observe while experimenting with the LM dictionary is that the percentage of total words classified as positive or negative by the LM dictionary is very low. Table 3.2 shows the average percentage of words classified by the LM dictionary for the 1,095 reports. It can be seen that less than 1% of total words are classified as positive. This suggests potential loss of important information that may generate signal that could potentially lead to higher accuracy scores.

Approach	Average % positive words	Average % negative words
LM dictionary	0.96%	4.88%

Table 3.2: Average percentage of words classified as positive and negative by LM Dictionary

3.3.2 Extended LM

One possible way to address the drawbacks of the LM dictionary mentioned in Section 3.3.1 is to extend the dictionary using the models discussed in Section 2.3, i.e. the word2vec skip-gram model. For each report, the input data is cleaned by removing the stop words and numbers, and lemmatizing the verbs and nouns, while maintaining the order of the words. We then train a skip-gram model on the corpus of the specific report, and find all the positive and negative words present in the report as classified by the LM dictionary. Following this, we search for the most similar words to the current word (positive or negative) using the following conditions:

i. The probability that the new word (positive or negative) is within a window size of 2 of the classified word is at least 95%.

ii. The new positive words are different from the original and newly found negative words. iii. The new negative words are different from the original and newly found positive words.

Figure 3.5 shows a sample result obtained when the positive and negative words for a Twitter 10-Q from 2016-06-29 is extended.

Figure 3.5: Example of extended positive and negative words for Twitter

Table 3.3 shows a comparison of the average percentage of positive and average percentage of negative words classified by the LM dictionary and the extended LM.

Approach	% Positive words	Average % Negative words
LM dictionary	0.96	4.88
LM dictionary extended	8.14	6.79

Table 3.3: Percentage of Classified words using LM and extended LM

It can be seen from Table 3.3 that the average percentages have increased as a result of the expansion of the LM dictionary. Additionally, due to this increase, we observe some shift in sentiment from negative to positive as shown in Figures 3.6 and 3.7.

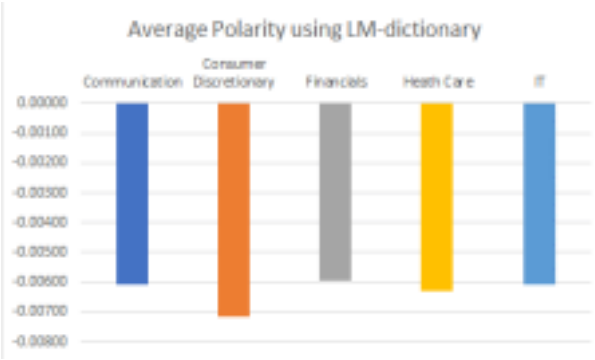

Figure 3.6: Average Polarity using LM-Dictionary

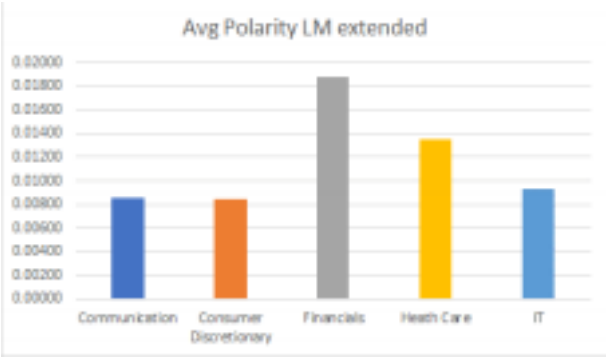

Figure 3.7: Average Polarity using LM-extended

Table 3.4 shows the accuracy scores obtained when the sentiment features generated from the extended LM dictionary are used to forecast price changes. These accuracy scores are obtained similarly to the method described in Section 3.3.1. In this case, we notice a comparable performance to the LM dictionary.

Approach	3 days return(%)	5 days return(%)	30 days return(%)	60 days return(%)
Polarity	51.67	49.18	51.61	44.44
Positivity	56.90	60.71	55.00	56.45

Table 3.4: Accuracy of LM Extended dictionary

3.3.3 FinBert

After examining the LM and the extended LM approaches, we realize that word frequency and word embedding models focus too much on individual words rather than considering the context surrounding a particular word. Therefore, we explore the FinBERT strategy discussed in Section 2.4. For each sentence in each report we categorize the sentence as either positive, negative, or neutral using FinBERT. Figure 3.8 shows an example of sentence classification for a Twitter 10-Q report.

Examples of sentences	Sentiment
Although we do not generate revenue directly from users or platform partners, we benefit from network effects where more activity on twitter results in the creation and distribution of more content, which attracts more users, platform partners and advertisers.our revenue for the three months ended march 31, 2014 was $250.5 million, which represents a 119% increase compared to the same period last year.	Positive
We generate the substantial majority of our revenue from the sale of advertising services, with the balance coming from data licensing arrangements and our mobile advertising exchange services.	Neutral
In the event that cost per ad engagement continues to decline, and we are unable to continue to offset the impact of such decreases on advertising revenue by increasing the number of ad engagements, our advertising revenue would decline a significant portion of our advertising revenue is delivered from advertisers outside of the united states.	Negative

Figure 3.8: Example Sentences FinBERT

Figure 3.9 illustrates the result of applying FinBERT on Abbott's 10-K and 10-Q reports.

Year	Ticker	Sector	File	Date	Negative Sentences	Neutral Sentences	Positive Sentences	3-Day adj	5-Day adj	30-Day adj	60-Day adj
2013	ABT	Health	10-K	2013-02-15	4	344	51	-0.283	0.101	-5.593	1.800
2014	ABT	Health	10-K	2014-02-21	9	294	50	-0.349	0.837	-2.295	-2.472
2015	ABT	Health	10-K	2015-02-27	8	351	67	-1.134	-0.682	0.048	0.279
2016	ABT	Health	10-K	2016-02-19	4	352	73	0.776	1.699	-0.580	5.620
2017	ABT	Health	10-K	2017-02-17	5	370	89	1.059	1.168	0.646	-0.927
2018	ABT	Health	10-K	2018-02-16	6	400	92	-0.092	0.237	5.308	1.896
2019	ABT	Health	10-K	2019-02-22	8	365	105	1.524	2.151	3.266	-4.031
2014	ABT	Health	10-Q	2014-05-07	7	59	4	1.208	0.240	-0.565	1.709
2014	ABT	Health	10-Q	2014-08-05	4	68	9	-1.270	-0.695	-1.822	-1.993
2014	ABT	Health	10-Q	2014-11-06	4	80	20	0.259	0.555	2.286	3.011
2015	ABT	Health	10-Q	2015-05-06	1	90	24	0.135	0.265	3.254	7.159
2015	ABT	Health	10-Q	2015-08-04	2	103	24	0.203	0.024	-5.592	-15.380
2015	ABT	Health	10-Q	2015-11-05	1	104	24	0.841	2.263	2.661	-0.716
2016	ABT	Health	10-Q	2016-05-04	0	101	24	-2.186	-1.252	-0.099	4.152
2016	ABT	Health	10-Q	2016-11-03	1	121	28	1.397	0.936	-6.171	-6.717
2017	ABT	Health	10-Q	2017-05-03	2	108	29	2.214	2.134	4.648	10.138
2017	ABT	Health	10-Q	2017-08-02	1	124	25	-0.492	0.121	4.122	7.086
2017	ABT	Health	10-Q	2017-11-02	1	130	29	1.414	1.711	-0.484	4.356
2018	ABT	Health	10-Q	2018-05-02	0	97	35	-0.832	0.178	3.097	0.998
2018	ABT	Health	10-Q	2018-08-01	0	100	32	-2.718	-1.669	-1.043	9.246
2018	ABT	Health	10-Q	2018-10-31	1	110	32	1.638	1.439	6.007	12.920
2019	ABT	Health	10-Q	2019-05-01	4	58	15	-1.234	-0.162	2.267	5.569

Figure 3.9: Abbott FinBERT

Table 3.5 shows the accuracy scores obtained when the sentiment features generated

from FinBERT are used to forecast price changes. These accuracy scores are obtained similarly to the methods described in Sections 3.3.1 and 3.3.2. Here, it can be seen that the accuracy scores are much higher than those of both LM and extended LM.

Approach	3 days return(%)	5 days return(%)	30 days return(%)	60 days return(%)
Polarity	67.16	53.03	56.16	51.95
Positivity	69.23	56.52	64.00	72.00

Table 3.5: Accuracy of FinBERT dictionary

3.4 Machine-Learning Based Classification Using Sentiment Features

In the process of binarizing the sentiment, we observed that a significant chunk of data is discarded. This is not ideal for a data intensive problem like the one we are trying to solve. Also, we noticed that the accuracy scores of the test data were higher than that of the train data. This could mean that our model is too simple to capture the complexity of the problem. One way to address these problems is to use machine learning to fit multiple features since machine learning models excel at handling high-dimensional data.

We begin by considering the following features derived from the LM dictionary, the extended LM dictionary (ELM) and the FinBERT model:

i. Number of positive words classified by LM dictionary

ii. Number of negative words classified by LM dictionary

iii. Total number of words in Management's discussion

iv. Number of positive words classified by ELM dictionary

v. Number of negative words classified by ELM dictionary

vi. New total number of words in Management's discussion (adding the extended ones) vii. Number of sentences in the Management's discussion

viii. Average length of sentences of the Management's discussion

ix. Polarity using LM

x. Polarity using ELM

xi. Number of negative sentences classified by FinBERT

xii. Number of positive sentences classified by FinBERT

xiii. Number of neutral sentences classified by FinBERT

From this relatively large number of features, we immediately suspect that the features may have substitution effects. In other words, before we proceed to fitting the machine learning models on these features, we must ensure that there is little or no multicollinearity between the features.

To test for multicollinearity, we perform the Variance Inflation Factor (VIF) test. VIF is calculated as:

$$VIF_i = \frac{1}{1 - R^2_i} \quad (3.1)$$

The results obtained are summarized in Table 3.6.

Feature	VIF
Positive words classified by LM	∞
Negative words classified by LM	∞
Number of words	∞
Positive words classified by extended LM	∞
Negative words classified by extended LM	∞
dictionary New number of words in	∞
Management's discussion Number of sentences	145.7
Average length of sentences	5
Polarity using LM	15.38
Polarity using LM extended	3.75
Number of negative sentences FinBERT	2.38
Number of positive sentences FinBERT	10.22
Number of neutral sentences FinBERT	142.3
	3
	2.61

Table 3.6: Features and VIF

A VIF value of ∞ implies perfect correlation. Therefore, we leave out all features whose VIF value is ∞. Thus, the original 13 features are reduced to 7 features. Following this, we do some feature engineering to further reduce the number of features. Specifically, we retain the polarity using LM, polarity using LM extended and the average length of sentences. We engineer the two other features in the following way:

$$polarity_F = \frac{(\text{positive sentences} - \text{negative sentences})}{\text{number of sentences}} \quad (3.2)$$

$$neutral\% = \frac{\text{neutral sentences}}{\text{number of sentences}} \quad (3.3)$$

Consequently, we are left with 5 features with acceptable VIF coefficients as shown in Table 3.7.

Feature	VIF
Polarity using LM	3.00
Polarity using LM extended	1.60
Polarity of sentences	1.14
Avg length of sentences in MD	9.07
Percentage neutral sentences	8.33
FinBERT	

Table 3.7: Reduced Feature Space and VIF

The 5 features seem to be equally important. Figure 3.8 shows the feature importance using the 3-day returns.

Feature	Feature Importance
Polarity using LM	20.64%
Polarity using LM extended	19.69%
Polarity of sentences	19.33%
Avg length of sentences in MD	20.17%
FinBERT Percentage neutral	20.17%

sentences FinBERT	

Table 3.8: XGBoost Feature Importance

We next train the machine learning models using the 5 features. To illustrate the effect of the multicollinearity, we also fit machine learning models on the original 13 features. Figure 3.10 shows the result of the algorithms. It can be seen from Figure 3.10 that reducing the number of features improved the accuracy of all models.

Returns	Logistic Regressioon		Random Forest		XG Boost	
	13 features	5 features	13 features	5 features	13 features	5 features
3-day	59.34%	60.58%	59.34%	60.58%	57.26%	61.00%
5-day	59.75%	61.83%	62.24%	61.83%	59.34%	61.00%
30-day	48.13%	50.62%	48.96%	51.87%	51.04%	54.77%
60-day	51.87%	53.11%	52.70%	51.87%	52.70%	53.11%

Figure 3.10: Out of sample accuracy for ML models

3.5 Investment Portfolio

To further test the performance of our models, we build a long-short trading strategy. We use the predicted direction of the price change as the trading signal and build equal-weighted portfolios of stocks. The benchmark portfolio is the equal-weighted portfolio of the 48 stocks. We evaluate the logistic regression, random forest, and XGBoost models on the test data (from 2018-01-01 to 2019-08-13). The holding periods used are 3, 5, 30, and 60 days. For example, if we predict the 3-day forward excess return of a stock as positive, we long the stock for 3 days. And if it is predicted as negative, we short the stock for 3 days. In order to compare the performance of different portfolios, we calculate the average annualized return, the information ratio and the maximum drawdown. The results of the portfolios are summarized in Table 3.9.

Annualized Return IR Max Drawdown

Logistic Regression -0.16

3-Day 5-Day 30-D ay 60-D ay	12.64% 11.27% 1.47% 12.19%

3-Day 5-Day 30-D ay 60-D ay	12.64% 11.27% -2.69% 10.35%	0.86 0.70 -0.1 0 0.92

-0.14 -0.33 -0.12

Random Forest	-0.16	-0.14						-0.30
								-0.21
XGBoost	3-Day	12.64%	0.86	-0.16	5-Day	14.09%	0.89	-0.14
							30-Day	14.47% 1.15 -0.29
							60-Day	5.01% 0.57 -0.17

Table 3.9: Long-short Trading Strategy Performance

From the results, the best model varies with the holding periods. For the holding period of 3 days, all three models have the same performance. For the holding periods of 5 days and 30 days, XGBoost tends to outperform the other models. However, for the holding period of 60 days, XGBoost underperforms logistic regression, which could be as a result of overfitting. Based on our experimental results, the best performance is obtained using XGBoost for a holding period of 30 days. It achieves an annualized return of 14.47%, an information ratio of 1.15 and a maximum drawdown of -0.29. Figures 3.11 to 3.22 show the performance, on the test data, of the long-short trading strategy based on the predictions of the machine learning models relative to the performance of the benchmark equal-weighted portfolio for different holding periods.

Logistic Regression

Figure 3.11: 3-Day Logistic Regression Figure 3.12: 5-Day Logistic Regression

Figure 3.13: 30-Day Logistic Regression Figure 3.14: 60-Day Logistic Regression

Random Forest

Figure 3.15: 3-Day Random Forest Figure 3.16: 5-Day Random Forest

Figure 3.17: 30-Day Random Forest Figure 3.18: 60-Day Random Forest

XGBoost

Figure 3.19: 3-Day XGBoost Figure 3.20: 5-Day XGBoost

Figure 3.21: 30-Day XGBoost Figure 3.22: 60-Day XGBoost

Summary, Conclusion, and Future Work

4.1 Summary

In this project, we used three methods to extract insights from companies' 10-K and 10-Q reports, namely:

- Loughran-McDonald dictionary
- LM Extended using word2vec
- FinBERT

From the information provided by those methods, we applied the following sentiment scoring metrics:

- Polarity
- Percentage positivity

In addition, we adjust and label the returns for different time horizons, which we use as the target variable of our analysis. The different time horizons we consider are 3 days, 5 days, 30 days, and 60 days.

Then, to determine the link between the sentiments and the direction of returns, we employed three machine learning models:

- Logistic regression
- Random forest
- XGBoost

Lastly, after verifying the models' accuracy on out-of-sample data, we created investment portfolios and examine their performance over time, and achieved the following results:

- All three models achieve the same trading performance metrics for the 3-day holding period.
- XGBoost performs best for the 5-day and 30-day holding period.
- Logistic regression outperforms XGBoost for the 60-day holding period, which may be as a result of overfitting.

4.2 Conclusion

The results from our study show that predictive signals can be generated from the texts

of 10-K and 10-Q reports. In particular, using machine learning models that incorporate dictionary, word embedding, and contextual models shows promising results.

4.3 Future Work

While the results obtained from this project are encouraging, a lot can still be done to improve on performance. A possible future direction would be to consider examining companies by sector since language may differ across sectors. Also, examining the individual companies may prove more effective. Another direction for future work could be the use of novel NLP methods such as XLNet and ELMo to analyze the corporate filings and compare their performances with the LM dictionary, word2vec, and FinBERT models. Finally, future work could consider extending the binary classification to multi-class classification to further categorize the price responses of companies after the release of their corporate filings.

Bibliography

[1] M. Mallikarjuna, and R.P. Rao, ”Evaluation of forecasting methods from selected stock market returns”, Finance Innov 5, 40 (2019). https://doi.org/10.1186/s40854-019-0157-x.

[2] S. Kogan, D. Levin, B.R. Routlege, J.S. Sagi, and N.A. Smith, ”Predicting risk from

financial reports with regression", *In Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, NAACL '09*, pp. 272–280, Stroudsburg, PA, USA. Association for Computational Linguistics.

[3] J. Engelberg, "Costly information processing: Evidence from earnings announcements. San Francisco", Meeting Paper, 2008.

[4] B. Xie, R. J. Passonneau, G. Creamer, and L. Wu, "Semantic frames to predict stock price movement", *In Proceedings of the 2013 Annual Meeting of the Association for Computational Linguistics*, 2013

[5] H. Lee, M. Surdeanu, B. MacCartney,and D. Jurafsky, "On the Importance of Text Analysis for Stock Price Prediction", *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, 2014.

[6] T. Loughran, B. McDonald, "When Is a Liability Not a Liability? Textual Analysis, Dictionaries, and 10-Ks", *Proceedings of the Journal of Finance (Vol. LXVI, No.1)*, 2011.

[7] A. Tixier, "Introduction to word embeddings", *Data Science and Mining Team Ecole Polytechnique*, 2015

[8] X. Rong, "word2vec Parameter Learning Explained", *Proceedings from Computing Re search Repository (CoRR)*,2014. http://arxiv.org/abs/1411.2738

[9] K. Gurney "An introduction to neural networks", *Taylor Francis e-Library*, 2004

[10] I2 Tutorials, 2019.

https://www.i2tutorials.com/hidden-layers-in-neural-networks/

[11] A. Dua, "Introduction to Word Embeddings and its Applications",2020. https://medium.com/compassred-data-blog/introduction-to-word-embeddings-and its-applications-8749fd1eb232

[12] X. Rong "Wevi: Word Embedding Visual Inspector" https://docs.google.com/document/d/1qUH1LvNcp5msoh2FEwTQAUX8KfMq2faGpNv4s4WXhgg/p

[13] T. Mikolov, K. Chen, G. Corrado, J. Dean, "Efficient Estimation of Word Representations in Vector Space". ICLR, 2013.

[14] J. Devlin, M. Chang, K. Lee, K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", 2018.

[15] D. Araci, "FinBERT: Financial Sentiment Analysis with Pre-trained Language Models", 2019.

[16] M. Lopez de Prado, ”Advances in Financial Machine Learning: Lecture 4/10”, 2018. https://papers.ssrn.com/sol3/papers.cfm?abstract id=3257420.

[17] W. Kenton, ”10-K”,Investopedia, 2020. https://www.investopedia.com/terms/1/10-k.asp

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, I. Polosukhin ”Attention is all you need”. ICLR, 2013.

[19] J. Alammar, ”The Illustrated Transformer”, 2020. http://jalammar.github.io/illustrated-transformer/

Appendix

Code for this project can be found at the following link:
https://github.coecis.cornell.edu/sr2232/Rebellion-Research