DEPARTMENT OF COMPUTER SCIENCE

Semantic Analysis of Financial Headlines Based on Realised Stock Returns

Research

Joshua Felmeden

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Master of Engineering in the Faculty of Engineering.

Tuesday 26th April, 2022

Abstract

A compulsory section, of at most 300 words

This section should précis the project context, aims and objectives, and main contributions (e.g., deliverables) and achievements; the same section may be called an abstract elsewhere. The goal is to ensure the reader is clear about what the topic is, what you have done within this topic, *and* what your view of the outcome is.

The former aspects should be guided by your specification: essentially this section is a (very) short version of what is typically the first chapter. If your project is experimental in nature, this should include a clear research hypothesis. This will obviously differ significantly for each project, but an example might be as follows:

> My research hypothesis is that a suitable genetic algorithm will yield more accurate results (when applied to the standard ACME data set) than the algorithm proposed by Jones and Smith, while also executing in less time.

The latter aspects should (ideally) be presented as a concise, factual bullet point list. Again the points will differ for each project, but an might be as follows:

- I spent 120 hours collecting material on and learning about the Java garbage-collection sub-system.
- I wrote a total of 5000 lines of source code, comprising a Linux device driver for a robot (in C) and a GUI (in Java) that is used to control it.
- I designed a new algorithm for computing the non-linear mapping from A-space to B-space using a genetic algorithm, see page 17.
- I implemented a version of the algorithm proposed by Jones and Smith in [6], see page 12, corrected a mistake in it, and compared the results with several alternatives.

Dedication and Acknowledgements

I am very fortunate to work with my superviser Rami Chehab, who assisted in my research greatly and inspired the vast quantity of this project. He has been a great mentor.

I would also like to thank my friends and family for always being there for me, especially when the going gets rough. You guys really helped make this university experience what it was.

Declaration

Joshua Felmeden, Tuesday 26th April, 2022

Contents

List of Figures

List of Tables

Ethics Statement

A compulsory section

In almost every project, this will be one of the following statements:

- “This project did not require ethical review, as determined by my supervisor, [fill in name]”; or
- “This project fits within the scope of ethics application 0026, as reviewed by my supervisor, [fill in name]”; or
- “An ethics application for this project was reviewed and approved by the faculty research ethics committee as application [fill in number]”.

See Section 3.2 of the unit Handbook for more information. If something went wrong and none of those three statements apply, then you should instead explain what happened.

Supporting Technologies

- I used a sample of headlines from Kaggle as training and validation data (https://www.kaggle.com/datasets/miguelaenlle/massive-stock-news-analysis-db-for-nlpbacktests)
- I used the Natural Language Toolkit to assist the preprocessing of data, using their English words and stop words data (https://www.nltk.org/)

Notation and Acronyms

NLP	:	Natural Language Processing
SESTM	:	Semantic Extraction via Screening and Topic Modelling
SESTM Specific Notation		
m	:	Number of words in sample
n	:	Number of articles in sample
$d_{i,j}$	:	Number of times word j appears in text i
$d_{[S],i}$	:	Subset of columns where the only indices are those with sentiment
$D = [d_1, \ldots, d_n]$	:	$m \times n$ Document term matrix
$sgn(y)$	:	Sign of returns of article y
$\hat{x}$	:	Expected value of variable x

Chapter 1

Introduction

1.1 Motivation

Financial news is a widely available resource that offers crucial information on the health and status of the stock market and operating businesses. A huge volume of content is created each day, to such an extent that it is virtually impossible for manual methods to keep up with, leading to developments in automated analysis. With the growth of natural language processing techniques, it has become possible to observe a news article and extract the sentiment of an article; exploiting this information as an indicator of potential future movement of the stock market.

An article is naturally formed of two parts, the headline and the article body. The headline itself is a unique text type, in that it is often not written by the author of the article, and serves as an attractor to a reader that encapsulates the story as a whole [Reah, 2002]. In attempting to fulfil this role, headlines have come to have a vocabulary that is alien from other vocabularies, and can sometimes be ambiguous or misleading in the attempt to draw a reader in to read the full article. The words chosen to be included in the headline therefore carry significant meaning, since they intend to convey a lot of information with far fewer words than other text types.

When analysing the sentiment of an article, the body is usually used as input, since the bulk of the information conveyed by a given article is in the body. However, since headlines are a summarisation of the body, it is possible to mine information from the headline itself with as much success as the information collected from the body [Kirange et al., 2016]. The lower word counts of headlines mean processing the data is much quicker, as well as being easier to collect large quantites of headlines.

With financial news outlets portraying the state of markets in such detail, the relationship between the words used in a news article and the effect on the stock market is one that can be utilised to predict future retunrs. In 2019, Ke et Al. [Ke et al., 2019] proposed a novel text mining algorithm that observed previous articles, and their subsequent effect on associated firm share prices, and reteroactively assigned sentiment to words in each article, creating a probabilistic model that can be used to effectively predict future movement of stocks based on news articles on a given day. They state that the algorithm could be extended to use any text based data with some teaching signal, in their case news articles as the text data, and the realised return of the associated stock. Analysing headlines in a similar manner would be beneficial as the headline of an article are both more readily available and shorter, meaning shorter computation times.

1.2 Aims and Objectives

The aims and objectives for this project are the following:

1. Implement and optimise the algorithm proposed by [Ke et al., 2019]
2. Train a model based using a dataset of financial news headlines, with the associated stock returns as a teaching signal.
3. Evaluate the predictive success of the trained model by creating portfolios according to predicted sentiment
4. Compare the performance of the algorithm against other sentiment analysis techniques

Chapter 2

Background

We begin by discussing the technical background that relates to the work in the dissertation, to provide a foundation from which to build the rest of the project.

2.1 Semantic Analysis

Semantic analysis (also known as opinion mining) is the task of identifying opinions or sentiment of authors from an input of text. The implications of being able to programmatically extract the intended meaning of an input are extremely useful in a variety of fields and is particularly prudent in a financial context. The nuances of natural language can sometimes make this difficult to extract, and therefore significant research has been conducted on the topic over the course of the last decades.

2.1.1 Lexicon Based Methods

The fundamental concept of sentiment analysis revolves around investigating a piece of text and deciding on a binary classification: positive or negative. The simplest method involves compiling a weighted word list, known as a lexicon. The weight of a word corresponds to the associated positivity or negativity of the word (for example 'great' would have a high positivity, while 'terrible' would have a very high negativity score). The overall sentiment of a piece of text could then be estimated by summing the individual positive sentiment scores of each word while subtracting the negative sentiment scores to yield an overall score for a piece of text. This lexicon is not limited to single words, and can be expanded to include n-grams (phrases of n words), as the context in which a word is used can dramatically change its sentiment. This may increase the accuracy of the dictionary at the cost of increased dimensionality. The lexicon itself must be compiled before it is possible to utilise this method, and is normally constructed via in depth analysis of many texts. One of the most widely used for English text is the Harvard-IV-4 psychosociological lexicon (H4) which is a general usage model that can be used to estimate the sentiment of a piece of text.

There are many difficulties faced with creating a dictionary of this sort, as language is often a subjective entity, leading to conflicting opinions in assigning tone to a specific word. Furthermore, the context within which a word is used can drastically change the intended sentiment of a word, for example, the word 'great' is naturally a very positive word, however, if used in a sarcastic manner (e.g. 'It's so great that my flight is delayed!') the seniment conveyed can be inverted entirely. For this reason, simply summing the sentiment of a piece of text on a word by word level can give an incorrect estimate.

Certain words that may have no meaning at all in one context, may have significant sentiment in another, particularly in the field of finance, where jargon dominates text. Loughran and McDonald [Loughran and McDonald, 2011] conducted an investigation into the use of standard lexicons for use in analysing 10-K filing reports, which are comprehensive reports filed by companies about their financial performance. They discovered that almost 75% of negative words found in the filings based on the H4N file were not typically negative in a financial context. For example, 'liability' is a negatively charged word in a standard context, while it carries no tone at all in the context of the filings, or have strong links to certain industry specific sectors, rather than general links. Loughran and McDonald created their own lexicon based on these results called the Loughran McDonald dictionary that is created for the purpose of classifying 10-k filings. They find that there is a high misclassification rate when using these generic lexicons, proving lexicon based methods work far better if it is bespoke for the topic.

2.1.2 Usage

Usage of these lexicons can vary greatly, with some simply summing the count of positive words and subtracting the count of negative words to end up with an overall word count leaning either positively or negatively. However, this is a very rudimentary method and can be augmented with more complex models. Loughran and McDonald themselves suggest using Term Frequency Inverse Document Frequency (TF-IDF) as a method to weight the word counts. TF-IDF is one of the most popular term weighting schemes, and it therefore makes sense to augment these strategies with this weighting.

TF-IDF has two distinct parts, term frequency and inverse document frequency. Term frequency (TF) is the relative frequency of a term t in a document d:

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d, t' \neq t} f_{t',d}} \tag{2.1}$$

where $f_{t,d}$ is the raw count of term t in document d. This is one method of calculating the term frequency, however there are many variations, such as logarithmically scaled frequency ($\log(1 + f_{t,d})$), boolean frequency (1 if t appears in d at all, and 0 otherwise), etc.

Inverse document frequency (IDF) measures how much information a word carries by its rarity. The generic equation is as follows:

$$idf(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|} \tag{2.2}$$

where $N = |D|$ and $D = [d_1, d_2, \ldots, d_n]$. Using these two components in conjuction, the final TF-IDF of a term is:

$$tfidf(t,d,D) = tf(t,d) \times idf(t,D) \tag{2.3}$$

This method is used to compensate for words appearing frequently. The TF controls the impact of high frequency words, ensuring that if a word has significantly higher frequency than other words, the impact does not grow out of control. Similarly, the IDF controls for the commonality of a word, in that if a word appears in a lot of the documents, the impact will be reduced as it is not rare.

2.2 Semantic Extraction via Screening and Topic Modelling

Semantic Extraction via Screening and topic model (SESTM) is a novel text mining algorithm that makes use of a teaching signal developed by Ke et Al. in 2019 [Ke et al., 2019]. It utilises realised stock returns as a teaching signal to develop a model of sentiment words in maximally positive or negative arguments in order to predict the sentiment of out of sample articles.

To begin, we assume that each headline has some sentiment score $p_i \in [0,1]$, with $p_i = 1$ being a headline with maximum positive sentiment, and $p_i = 0$ maximally negative. Our problem is ot model the distribution of returns of a headline y_i given the sentiment of the headline p_i. Thus we have assumed:

$$\mathbb{P}(\text{sgn}(y_i) = 1) = g(p_i), \text{ for a monotone increasing function } g(\cdot) \tag{2.4}$$

where $\text{sgn}(x)$ returns 1 if $x > 0$ and -1 otherwise. This assumption simply states that the higher the sentiment score, the higher the probability the headline has of returning positive returns.

Next, we observe the distribution of word counts in an article. We assume a vocabulary has partition

$$\{1, 2, \ldots, m\} = S \cup N \tag{2.5}$$

where S is the set of sentiment charged words and N is the set of sentiment neutral words. Thus, the distribution of sentiment charged words are the vector $d_{[S],i}$, similarly for sentiment neutral words $d_{[N],i}$, although sentiment neutral words serve as useless noise and remain unmodelled.

Utilising the work of Hoffman, we adapt latent semantic analysis (LSA) which is a technique that maps high-dimensional word count vectors to a lower dimensional representation (in our case, the realised returns) [Hofmann, 2013]. We then assume that the vector of sentiment charged word counts are generated by a mixture multinomial model of form

$$d_{[S],i} \sim \text{Multinomial}(s_i, p_i O_+ + (1-p_i)O_-) \tag{2.6}$$

where s_i is the total count of sentiment charged words in headline i. O_+ is the probability distribution and simply describes expected word counts in a maximally positive headline (namely $p_i = 1$). Similarly, O_- describes expected word counts in maximally negative headlines ($p_i = 0$). Fundamentally, this model is the probability of counts for each sentiment charged word in our vocabulary, with s_i trials and the probability that a word is in a document is modelled by $p_i O_+ + (1-p_i)O_-$, which takes into account the positive and negative values of O.

Using these two probability distributions, we can also gain insight into vectors of tone T (total leaning of a word) and frequency F (how often a word appears), since $O_\pm$ captures both frequency and sentiment:

$$F = \frac{1}{2}(O_+ + O_-), \quad T = \frac{1}{2}(O_+ - O_-) \tag{2.7}$$

2.2.1 Screening for sentiment charged words

The first step in this algorithm is to screen for sentiment words in a collection of articles, since sentiment neutral words are a nuidance and contribute to noise. This strategy isolates the sentiment charged words and uses these alone to calculate sentiment. This is achieved by using a supervised approach with the realised returns of an associated stock, since if a word appears in headlines that result in positive returns, it is reasonable to assume that the word carries positive sentiment. Some notation will be introduced here to facilitate the discussion and explanation of the algorithm.

- The sample is defined as n articles producing a dictionary of m words.
- The word count of article i is recorded in vector d_i
- $D = [d_1, d_2, \ldots, d_n]$ is an $m \times n$ document term matrix
- The count of sentiment charged words in article i is defined as the submatrix $d_{[S],i}$.

For a word j, we define f_j as the fractional representation of the frequency with which a word appears in a positively tagged article versus the frequency with which a word appears in any article:

$$f_j = \frac{\text{count of word } j \text{ in article with } sgn(y) = +1}{\text{count of word } j \text{ in all articles}} \tag{2.8}$$

Here, sgn is simply the sign of the difference in tagged returns of an article. If an article is released on day t (specifically, between 4pm of day $t-1$ and 4pm of day t), the article is tagged with the returns of the associated firm from day $t-1$ to $t+1$ (specifically market close on day $t-2$ to close on day $t+1$). Naturally, higher values of f_j indicate that word j appears in a higher proportion of positively tagged articles.

Next, f_j is compared against positive and negative threshold. We define $\hat{\pi}$ to be the fraction of articles that have $sgn(y) = +1$, or articles that are tagged with positive returns. In practice, as the sample size increases, $\hat{\pi}$ tends towards 0.5. For sentiment neutral words, it is expected that $f_j \approx \hat{\pi}$ with some variance either side. Therefore, two thresholds are set: α_+ and α_-. These thresholds are set such that any word with $f_j \geq \hat{\pi} + \alpha_+$ is determined to be sentiment positive, and the inverse for words such that $f_j \leq \hat{\pi} - \alpha_-$. This filtering technique accepts words that are further away from the expected average sentiment, meaning only those words with extreme sentiment are included. To ensure that words appearing infrequently do not skew sentiment values significantly (for example a word appearing in exactly one article that is maximally positive ending up with $f_j = 1$), we introduce a third parameter

κ, restricting words that appear in fewer than κ headlines. The number of articles in which word j appears is defined as k_j The set of sentiment charged words is therefore defined by:

$$S = \{j : f_j \geq \hat{\pi} + \alpha_+ \cup f_j \leq \hat{\pi} - \alpha_-\} \cap \{j : k_j \geq \kappa\} \tag{2.9}$$

The three parameters introduced here $(\alpha_+, \alpha_-, \kappa)$ are hyperparameters and are tuned via cross-validation (see section 3.3 for usage). The best choice for each parameter is selected based on minimising a cost function.

2.2.2 Learning Sentiment Topics

With the wordlist S calculated, we can now fit a two-topic model to each of the sentiment-charged counts. We introduce matrix $O = [O_+, O_-]$ that determines the expected probability of sentiment charged words in each article using a supervised learning approach. The teaching signal in this case is the realised three day returns of the associated firm for each headline.

In the model, p_i is the headline's sentiment score, described by:

$$p_i = \frac{\text{rank of } y_i \text{ in } \{y_l\}_{l=1}^n}{n} \tag{2.10}$$

where the return of a headline is represented by y, and headlines are ranked in ascending order (meaning an article with the highest returns would have p_i of 1). More concretely, the returns are calculated as discussed above, and represented as a percentage. Let the price of the associated firm on day t be y_t, the three day returns would be calculated as $y_{t+1}/y_{t-1} - 1$. As before, day $t-1$ refers to market close on day $t-2$ and day $t+1$ market close on day $t+1$.

Once this value is calculated, let $h_i = d_{[S],i}/s_i$ denote the $|S| \times 1$ vector of (sentiment charged) word frequencies for article i. Using model 2.6, we know that the expected distribution of these sentiment word frequencies can be modelled as:

$$\mathbb{E}h_i = \mathbb{E}\frac{d_{[S],i}}{s_i} = p_i O_+ + (1-p_i)O_- \tag{2.11}$$

or in matrix form:

$$\mathbb{E}H = OW, \quad \text{where } W = \begin{bmatrix} p_1 & \cdots & p_n \\ 1-p_1 & \cdots & 1-p_n \end{bmatrix}, \text{ and } H = [h_1, h_2, \dots, h_n] \tag{2.12}$$

We are now able to estimate O via regression of H on W. H is not directly observed due to S being unobserved, so we estimate H by plugging in S:

$$h_i = \frac{d_{[S],i}}{s_i} \quad \text{where } s_i = \sum_{j \in S} d_{j,i} \tag{2.13}$$

W is estimated using the ranks of returns described in 2.10, leading to the final representation of:

$$O = [h_1, h_2, \dots, h_n]W'(WW')^{-1}, \quad \text{where } W = \begin{bmatrix} p_1 & p_2 & \cdots & p_n \\ 1-p_1 & 1-p_2 & \cdots & 1-p_n \end{bmatrix} \tag{2.14}$$

Finally, O may have negative entries, so we set these to zero and normalise each column to have ℓ^1-norm. The resulting matrix is referred to as O to simplify notation.

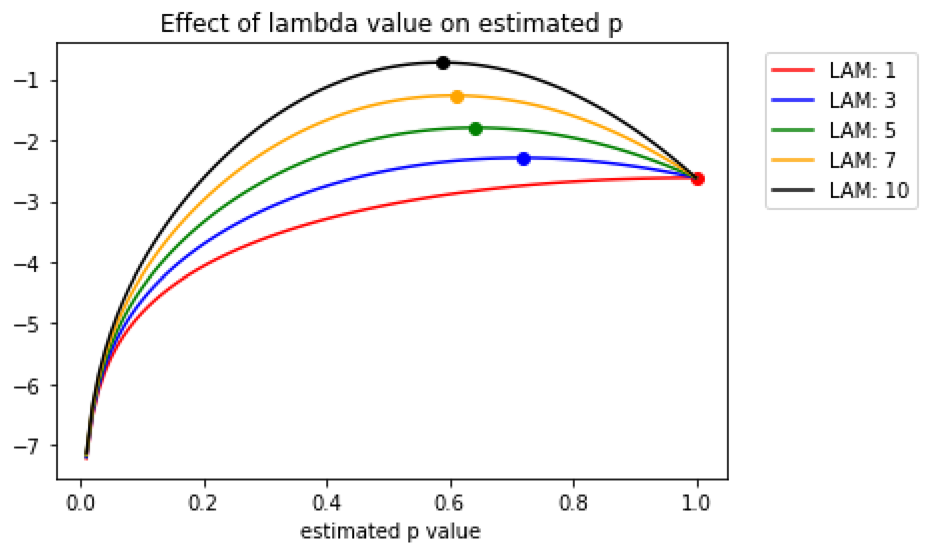


Figure 2.1: Effect of different values of λ on estimated p value. The large point on each line is the maximum value, and reflects the returned estimated p of a headline

2.2.3 Scoring New Headlines

The previous steps give estimators S and O. Using these, we are able to estimate the sentiment score p for a new article that is not in the training sample. Using model 2.6, we estimate p using Maximum Likelihood Estimation. This is simply testing values in some range and determining which value gives the maximum output. We also include penalty term $\lambda \log(p(1-p))$ and finish with the following optimisation:

$$\widehat{p} = \arg\max_{p\in[0,1]} \left\{ \widehat{s}^{-1} \sum_{j\in S} \log(pO_{+,j} + (1-p)O_{-,j}) + \lambda \log(p(1-p)) \right\} \tag{2.15}$$

where $\widehat{s}$ is the total count of word from S in the new article, $d_j, O_{+,j}, O_{-,j}$ are the entries for word j in the corresponding vectors and λ is a tuning parameter that ensures that the majority of headlines are neutral by pushing the estimate towards a neutral sentiment score of 0.5. The effect of lambda is shown in figure 2.1

2.3 Financial Markets and Stock Exchanges

A stock exchange is a platform from which buyers can connect with stock sellers. Stocks are traded in a number of exchanges including the New York Stock exchange and Nasdaq. The word stock itself means ownership or equity in a corporation, and indicates a share of ownership in a firm [Teweles and Bradley, 1998]. Each firm has its own stock ticker that uniquely identifies it in the market (e.g. Amazon's stock ticker is AMZN).

The stock exchange is only active on certain days and for specific times. The time that an exchange opens varies on the exchange, but the majority of exchanges open at 9:30 a.m. and close at 4.00 p.m, excluding half days, where the market closes at 1:00 p.m. Furthermore, the markets are shut on weekends and 9 trading holidays during which no trading is able to take place [Hours, 2022]. Of course, the opening and closing times depend are in the local time for each market, and for American based exchanges, this is in Eastern Standard Time (EST).

2.4 Portfolios and Financial Market Analysis

Evaluating word lists generated by any method of sentiment analysis can be done by creating portfolios based on news with the highest counts of positive or negative words and buying the most positive stocks while selling the most negative. This gives a good indication of the predictive power of a word list.

2.4.1 Portfolios

A portfolio is simply a list of stocks that can be invested in by an individual or firm. The returns from a portfolio is defined as the profit accrued from all stocks over a set time period, usually daily, monthly or annually. Due to the nature of stocks, simply listing the returns as a concrete value does not convey the information required. For example, if stock 'A' were to be invested in at value \$50, and it rose to \$60 the following day, the returns could be said to be \$10. However, if stock 'B' were valued at \$1000, and the following day it rose to \$1010, the monetary value would be equivalent at \$10, but the percentage return is vastly different; 20% returns for stock 'A' and 1% for 'B'. For this reason, returns from portfolios are expressed as a percentage.

Daily returns can be very marginal, as the time period is very small, often being smaller than 1%. In the interest of readability, *basis points* (also known as bps or bips) are used in lieu of a percentage, where 1 bip is equivalent to 0.01%. This makes it much easier to represent very small returns as are common in daily returns.

Creating a portfolio

A portfolio is constructed using a number of stocks and can be either bought (taking the 'long' position) or sold (taking the 'short' position). For stocks that are bought, the returns can be calculated from the difference in price at the time that the stock is sold. More concretely, if a stock has value S_t at time t, and held for n days before being sold, the long returns in percentage form can be calculated using the following formula:

$$\frac{S_{t+n}}{S_t} - 1 \tag{2.16}$$

Similarly, for short returns, as the stock is being sold, profit is acquired if the stock falls in value, therefore the returns can be calculated using the following formula:

$$\frac{S_t}{S_{t+n}} - 1 \tag{2.17}$$

Of course, these simple formulae negect transaction fees that can apply when constructing real portfolios. However, as we are creating portfolios in a theoretical sense, this suits our needs.

Once the portfolio has been constructed, the weighting for each stock must be considered. Each portfolio will have a value, which is the amount of money invested into it, and each stock will in turn get an investment that is a percentage of this overall value. There are two strategies that we will consider: equal weighted and value weighted strategies. Equal weighted is very simple: if a portfolio is comprised of n stocks and has some investment v, each stock has v/n invested into it. This strategy glosses over differences in stock size or price. On the other hand, value weighted portfolios assign much more money to stocks with higher value associated to them. This can be calculated in a number of ways, but the way we calculate this is if stock s_i in portfolio $P = [s_1, ..., s_n]$ has market value $P_{i,t}$ at time t, the weight of stock s_i would be:

$$w_{i,t} = \frac{P_{i,t}}{\sum_n^1 P_{n,t}} \tag{2.18}$$

The amount invested into stock s_i at time t would then be $w_{i,t} \times v$.

Equal weighted stocks more closely resemble hedge funds, as well as being the case that smaller companies are able to more quickly encapsulate market share and investor interest. Equal weighted investments ensure a portfolio has a higher representation of smaller stocks, at the higher risk of the stock failing. Conversely, value weighted portfolios tend to be safer, as they prioritise larger companies that are more stable. The downside to utilising this method is that the large percentage increases observed in smaller stocks will have less effect, and therefore some profit can be lost.

2.4.2 Evaluating a portfolio

To successfully determine the success of a portfolio, it is not always as simple as observing the profit alone. While this is a good indicator of the potential returns that could be gleaned from a given portfolio or investment method, it is important to consider external factors and risks that may be involved. The following methods are used to provide more insight into an investment method.

Sharpe Ratio

William Sharpe created the sharpe ratio in 1966 and is one of the most referenced comparison of risk versus return in finance [Sharpe, 1966]. The formula for this ratio is exceedingly simple — one of the key factors in its wide usage — and is as follows:

$$S(x) = \frac{r_x - R_f}{\sigma(r_x)}$$

where x is the investment, r_x is the average rate of return of x, R_f is the risk free rate of return, and $\sigma(r_x)$ is the standard deviation of r_x. The risk free rate of return is simply the theoretical rate of return on an investment with absolutely no risk. Subtracting these risk free returns from the average rate of returns of x yields the true rate of returns.

The value of an investment's Sharpe ratio measures the performance with adjustment for risk: the higher the ratio, the better the performance of the investment when adjusted for risk. As a reference, a ratio of 1 or higher is good, 2 or better is very good, and 3 or better is excellent.

Fama French 3 and 5 Factor Models

Eugene Fama and Kenneth French co-authored a 1992 paper detailing risk factors in returns on both stocks and bonds. This extends the work Sharpe completed on the Sharpe ratio and goes further in exploring risk factors in returns, along with the capital asset pricing model (CAPM) [Fama and French, 1992]. CAPM is used for describing systematic risk and expected return, especially for that in stocks. The equation for this is:

$$ER_i = R_f + \beta_t(ER_m - R_f) \tag{2.19}$$

where ER_i is the expected return of the investment, R_f is the risk free rate, β_i is the beta of the investment and $ER_m - R_f$ is the market risk premium. The beta of an investment is the volatility compared to the rest of the market. It encompases the sensitivity of a stock to changes in the market. In essence, this gives the expected returns of an asset based on systematic risk. Building on this, Fama and French observed two additional risk factors: the size premium of an asset, or small minus big (SMB), and the value premium, or high minus low (HML). SMB is used to account for companies with small value stocks that generate high returns, while HML accommodates for stocks with equity that is valued cheaply compared to its book value that generate higher returns in comparison to the rest of the market. These factors are used in conjunction to provide the following formula for the Fama French 3 factor model (FF3 model):

$$ER_i = R_f + \beta_1(ER_m - R_f) + \beta_2(SMB) + \beta_3(HML) + \alpha$$

The values for SMB and HML are available from French's website [French, 2022], and can be collected for daily, monthly, or yearly returns. Computing this model on a series of returns from a portfolio gives useful information on the nature of the returns, since the model explains part of the returns. The values for the βs detail the exposure to exposure to each of the risk factors while the α, or the intercept, refers to the amount that a portfolio outperformed the expectations of the FF3 model. This alpha is representative of the amount of private or new information that is external from the market and is utilised to construct the portfolio. This means that more simplistic methods of selecting portfolios will not have any private information and therefore the intercept will be much close to zero while a more complicated model will have a larger intercept as simple methods have profits that can be explained by these risk factors.

This model was then revisted by Fama and French in 2015 where they observed two additional factors: robust minus weak (RMW) and conservative minus aggressive (CMA) [Fama and French, 2015]. RMW corresponds to the profitability of an asset, in that it is the difference between returns of robust or high and weak or low operating profitability. CMA corresponds to the investment factor, and is the difference between returns of conservatively investing firms versus aggressively investing firms, giving the updated formula:

$$ER_i = R_f + \beta_1(ER_m - R_f) + \beta_2(SMB) + \beta_3(HML) + \beta_4(RMW) + \beta_5(CMA) + \alpha \tag{2.20}$$

Chapter 3

Project Execution

In this section, we present the execution of the project, giving an overview of the dataset used, the configurations of hyperparameters and programming completed. The main idea of the project is to gather a dataset of headlines over a given time period and implement and analyse the algorithm presented by Ke et al. in their 2019 paper [Ke et al., 2019]. Due to the difference in language used in headlines compared to that used in headlines, slightly different approaches were considered for some of the steps, but the theory behind the model remains unchanged.

3.1 Project Components

For this project, I opted to use Jupyter notebook as a base on which to implement the data processing, model training and validation, and portfolio formation. The modular nature of Jupyter, combined with Python's wide range of useful libraries facilitated the creation of this model. Creating each section as an independent cell allows for rigorous debugging and testing of each individual part, which would otherwise be extremely time consuming to test. The full notebook along with accompanying datasets can be found on GitHub [Felmeden, 2022].

3.2 Dataset and Pre-Processing

The dataset used for training and validation is available from Kaggle[1] and is a collection of around 1.4 million headlines for 6000 stocks from 2009 to 2020. Each headline has the date published, and the ticker that the headline concerns. Some headlines have multiple tickers associated with them, but each ticker-headline combination is a unique entry in the dataset.

The first challenge is to align these headlines with the relevant three day returns, and this was achieved using the Yahoo Finance python library.[2] Once again, this relates to market close on day $t-2$ to close on day $t+1$ for a headline released on day t (between 4pm on day $t-1$ to 4pm on day t). Some headlines are released on non-market days (such as weekends), and for these edge cases, the next available market day is selected as day t, and then the previous market day from this new day t is defined as day $t-1$. Similarly, for market days where $t+1$ would fall on a non-market day, the next available market day is defined as day $t+1$. As many headline are released each day, computing the returns for each unique headline in this way would create significant redundancy, and therefore for each unique stock in the dataset, market data for the entire 11 year timespan is pulled in order to create a lookup table. Then, as opposed to calculating the stock values for each headline, if a headline is released on day t relating to stock s, the open and close values can be retrieved via the key (s,t). Finally, each headline is iterated through, assigning the appropriate market close values from the lookup table, and stored in json format for future usage. An example json entry is shown below in listing 3.1 where 'open' refers to market value of a ticker day $t-1$ and 'close' refers to market value on day $t+1$.

Note that some tickers do not have publicly available stock market information for the entire span of

[1] https://www.kaggle.com/datasets/miguelaenlle/massive-stock-news-analysis-db-for-nlpbacktests
[2] https://pypi.org/project/yfinance/

```
        {
                "headline": Barclays Maintains Equal-Weight on Agilent
                Technologies, Lowers Price Target to $76
                "date": 2020-03-26
                "ticker": A
                "mrkt_info": {
                        "open": 67.0
                        "close": 70.9100036621
                }
        }
```

Listing 3.1: Example JSON headline entry

```
day_t_data = {}
TOTAL_ARTS = len(stock_data)
for t in stock_data:
        # iterate through each ticker, and gather day t-1 (from_stock),
        # day t and day t+1 (to_stock) data for ease of lookup
        #
        # to get day t: day_t_data[t][date]['day_t']
        ticker_date_data = {}
        start = min(stock_data[t].index)
        end = max(stock_data[t].index)
        curr_date = start
        while curr_date < end:
                day_t = curr_date
                while (not (day_t in stock_data[t].index) and day_t <= end):
                        day_t += dt.timedelta(days=1)
                from_stock = day_t - dt.timedelta(days=2)
                to_stock = day_t + dt.timedelta(days=1)
                while (not (from_stock in stock_data[t].index) and from_stock >= start):
                        from_stock -= dt.timedelta(days=1)
                while (not (to_stock in stock_data[t].index) and (to_stock <= end):
                        to_stock += dt.timedelta(days=1)
                if(from_stock > start and to_stock < end):
                        ticker_date_data[curr_date] = {
                                'day_t': day_t,
                                'from_stock': stock_data[t]['Close'][from_stock],
                                'to_stock': stock_data[t]['Close'][to_stock]
                        }
                curr_date += dt.timedelta(days=1)
        day_t_data[t] = ticker_date_data
```

Listing 3.2: Creating lookup table

the sample. This is due to some companies being private for the duration, or turning private, meaning that their information is not accessible through standard means. Headlines aligned with these private tickers are removed from the sample, leaving around 1 million articles in the sample.

With the headlines aligned to the appropriate returns, the text data must be preprocessed to allow for successful and efficient semantic analysis. Taking the text content of each headline, the following transformations are applied, also detailed by figure 3.1:

- Convert the headline to lower case. This is to ensure that different cases do not lead to multiple entries of the same word, differing only by letter captialisation.
- Remove non alphabetic characters
  - Spaces are retained to allow for tokenisation
- Tokenise the headline (convert to list of words)
- Remove non-English words [3]
- Remove stop words. [4] Stop words are a term used in NLP to describe very commonly used words that are unimportant (such as 'and' or 'the'). They serve only as noise and are removed to allow

[3]The list of English words is available from item 106 from https://www.nltk.org/nltk_data/
[4]The list of stopwords used is from item 86 from https://www.nltk.org/nltk_data/

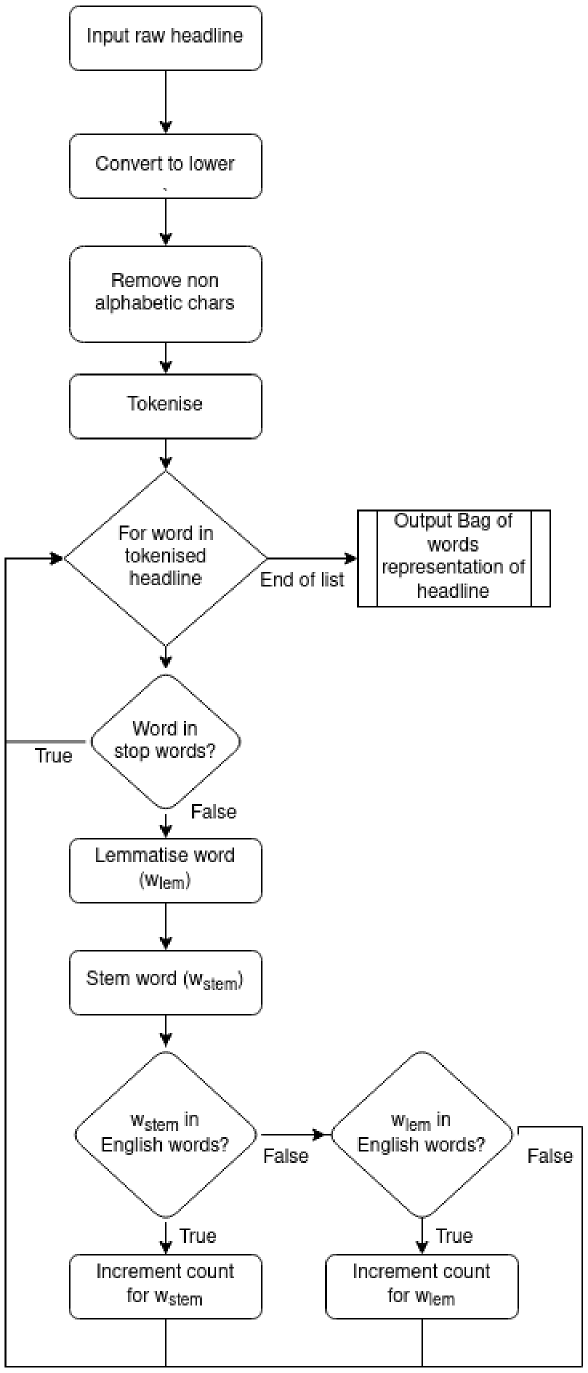


Figure 3.1: Flowchart showing conversion of raw headline into bag of words format

focus on more important news.

- Lemmatise each word (for example converting 'mice' to 'mouse' or 'skis' to 'ski') [5]
- Stem each word (for example, 'regional' to 'region').[6] Note that as stemming aims to create the most general stem of a word, it sometimes leaves a word that is not English (such as 'easily' to 'easili'). For this reason, if the stemmed word is not in the list of English words, but the lemmatized word is, then this word is not stemmed.
- Convert to bag of words (BOW) representation (list of unique words with associated word counts for a given headline)

[5]The lemmatisation process uses WordNet (item 106) from `https://www.nltk.org/nltk_data/`
[6]The stemming process uses Snowball stemmer from wordnet from `https://www.nltk.org/nltk_data/`

Window start date	$\lvert S\rvert/2$	α_+	α_-	κ	λ	Minimum error	Avg Min Error
2010-1-1	100	0.0529	0.0487	92	5	20402.2	0.24727
2010-5-1	100	0.0405	0.0384	94	5	20714.66	0.24926
2010-9-1	25	0.1116	0.1043	92	5	20426.81	0.24675
2011-1-1	25	0.1023	0.1064	92	1	19963.67	0.24574
2011-5-1	50	0.1002	0.0921	90	5	19075.08	0.24598
2011-9-1	100	0.0425	0.0404	94	5	19255.78	0.24653
2012-1-1	100	0.0754	0.0744	88	5	20474.7	0.24744
2012-5-1	100	0.051	0.0489	92	10	21839.81	0.24961
2012-9-1	100	0.0536	0.0473	92	10	22243.55	0.24931
2013-1-1	50	0.0536	0.0672	94	5	21546.5	0.24702
***2013-5-1**	**100**	**0.084**	**0.0913**	**86**	**5**	**22095.56**	**0.24552**
2013-9-1	100	0.0688	0.076	88	10	23415.73	0.24885
2014-1-1	100	0.0395	0.0375	94	5	24819.11	0.24818
2014-5-1	100	0.0823	0.0954	86	5	24060.98	0.24652
2014-9-1	100	0.0392	0.0412	94	5	23536.73	0.24904
2015-1-1	100	0.0778	0.0898	86	5	22801.46	0.24805
2015-5-1	100	0.0471	0.0571	92	5	23642.32	0.24871
2015-9-1	100	0.0603	0.0734	90	5	26361.23	0.24772
2016-1-1	100	0.0478	0.0518	92	5	29950.54	0.24818

Table 3.1: Best configuration and error for each window. Smallest error window highlighted in **bold**

By utilising stemming where appropriate (i.e. when the stemmed word is included in the list of English words), we ensure that the resulting bag of words most closely resembles the original headline, while also grouping words that are similar by root. This helps to keep sentiment consistent, as the root of a word is likely to be what holds sentiment, rather than the form it is used in. Another complication is the part of speech according to which a word is lemmatised. For example, the word 'trying' is both a noun and a verb, depending on context (e.g. 'a *trying* quarter' versus 'I was *trying* really hard'). The lemmatisation of this word in the context of a noun is 'trying', in verb context is 'try' and the stem is 'tri'. By default, the Wordnet Lemmatizer assumes the input is a noun, and I decided to do the same, at the potential risk of misclassifying a word. However, the shortcomings of treating each word as a noun are often covered by the stemming of the word, therefore using the two in conjunction gives the most accurate root of a word.

3.3 Training the Model

Using the BOW representation of each of the headlines, the data is able to be processed according to the algorithm outlined by Ke et al. In the spirit of the original paper, the dataset is divided up into 17 three year training and validation windows, where two years are used for training a model, and the final year is used for validation purposes. More concretely, the training sample begins in 2010-01-01 and ends on 2017-12-31, the validation sample begins in 2012-01-01 and ends in 2018-12-31, leaving articles between 2019-01-01 and 2020-06-08 as an out of sample dataset used for testing the robustness of the model. All the computation is completed on this window, and then it is moved forward four months and repeated in a rolling window method.

The training section employs the screening (section 2.2.1) and learning (section 2.2.2) steps, while the validation is the application of the scoring new headlines (section 2.2.3) step. The validation section is used to consider the hyperparameters ($\alpha_+, \alpha_-, \kappa, \lambda$), and these are evaluated according to a fixed number of possibilities. $\alpha_\pm$ is calculated such that S has either 25, 50, or 100 words of each sentiment (i.e. for the selection of $\alpha = 25$, $|S| = 50$). This is calculated via binary search, assuming $\alpha_\pm = 0.25$ at first. If the resulting set of words is larger than the selected configuration, then α must be larger than the current value, as the larger α is, the fewer words satisfy the condition. Similarly, the inverse for the case where the set of words contains fewer words than desired. κ is selected to be the 86, 88, 90, 92 or 94th percentiles of word counts. Note that the κ restraint is applied first such that a word is not selected via the α constraint that must then be removed due to the κ constraint, leaving S with fewer words than desired. Combining both the calculated $\alpha_\pm$ values and the calculated κ value, the list of sentiment words

```
kappa_configs   = [86, 88, 90, 92, 94]
alpha_configs   = [25,50,100]
# fraction of positively tagged training articles
train_pi = sum(sgn_i > 0 for sgn_i in train_sgn)/len(train_sgn)
for alpha in alpha_configs:
      for KAPPA in kappa_configs:
      #TRAINING
      kappa_percentile = np.percentile(np.array(list(total_j.values())),KAPPA)
      # return the nth percentile of all appearances for KAPPA

      #calculate alpha vals
      ALPHA_PLUS  = train_pi/2
      ALPHA_MINUS = train_pi/2
      delta_plus  = train_pi/4
      delta_minus  = train_pi/4
      # set limit on max iterations
      delta_limit = 0.0001

      while(delta_plus > delta_limit):
            no_pos_words = len([w for w in total_j if f[w] >= train_pi + ALPHA_PLUS and
    total_j[w] >= kappa_percentile])
            if no_pos_words == alpha:
                  # alpha plus found
                  delta_plus = 0
            elif (no_pos_words > alpha):
                  ALPHA_PLUS += delta_plus
                  delta_plus /= 2
            else:
                  ALPHA_PLUS -= delta_plus
                  delta_plus /= 2
      while(delta_minus > delta_limit):
            no_neg_words = len([w for w in total_j if f[w] <= train_pi - ALPHA_MINUS and
    total_j[w] >= kappa_percentile])
            if no_neg_words == alpha:
                  # alpha minus found
                  delta_minus = 0
            elif (no_neg_words > alpha):
                  ALPHA_MINUS += delta_minus
                  delta_minus /= 2
            else:
                  ALPHA_MINUS -= delta_minus
                  delta_minus /= 2
      sentiment_words = [w for w in total_j if ((f[w] >= train_pi + ALPHA_PLUS or f[w] <=
     train_pi - ALPHA_MINUS) and total_j[w] >= kappa_percentile)]
```

Listing 3.3: Calculating list of sentiment words

for the training window is compiled, illustrated in listing 3.3. Finally, λ is selected to be either 1, 5, or 10, for a total of 45 configurations.

Each of these 45 configurations is iterated through for each window, and the ℓ^1 error is calculated for each, before selecting the setup with minimum error (as this is our loss function). ℓ^1 error in this case is simply:

$$\sum_{i=1}^{n} |\widehat{p_i} - p_i| \tag{3.1}$$

where $\widehat{p_i}$ is the estimated sentiment and p_i is the standardised return rank of article i in the validation set. The loss function of ℓ^1-norm error was selected for its robustness. The entire process of training and validation takes a considerable length of time and therefore some time was spent optimising the code. A complete list of optimisations can be found in appendix (REFERENCE)

Table 3.1 details the results of the completed rolling window training. Due to the nature of news, some validation sets are larger than others, leading to skewed summed ℓ^1-norm error. To accommodate for this variation is sample size, the error is taken as an average over all headlines in the sample. Window 2011-1-1 has the smallest minimum error, but also has the smallest validation sample size and after controlling for this factor, window 2013-5-1 is has slightly lower error, meaning this is the optimum window.

3.3.1 Bigram training

Naturally, the order in which words appear in a headline can have a profound impact on the sentiment of a word. This can be captured by training the model on *bigrams*, which are sequences of two words, rather than single words alone. This can then be combined with the lexicon of unigrams to provide a clearer insight into the true sentiment of a headline based on the words used.

```
KAPPA_BIGRAM = 90
kappa_percentile_bigram = np.percentile(np.array(list(total_j.values())),KAPPA_BIGRAM)
# return the nth percentile of all appearances for KAPPA_BIGRAM
bigrams_to_remove = []
mutual_info = {}
for w in total_j_bigram:
      component_words = w.split()
      if not (total_j[component_words[0]] >= kappa_percentile_bigram and total_j[
    component_words[1]] >= kappa_percentile_bigram):
      bigrams_to_remove.append(w)
      else:
      mutual_info[w] = total_j_bigram[w] / (total_j[component_words[0]] * total_j[
    component_words[1]])
mutual_info_percentile = np.percentile(np.array(list(mutual_info.values())),95)
bigrams_to_remove.extend([w for w in mutual_info if mutual_info[w] <=
    mutual_info_percentile])
for b in bigrams_to_remove:
      pos_j_bigram.pop(b)
      total_j_bigram.pop(b)
      f_bigram.pop(b)
```

Listing 3.4: Calculating list of sentiment words

The algorithm naturally lends itself to training with bigrams and little is needed in the way of adjustment, as the bigrams are treated the same way as words. The only slight difference is the way that bigrams are filtered out. Due to the nature of bigrams, they will be, on average, far less frequent than unigrams, simply because of the increased number of possibilities. For this reason, centain measures are taken to ensure that only meaningful bigrams are considered. For each window, all articles are divided into bigrams in the same way that the BOW representations are created. Stopwords are not considered for potential bigrams, and as such if two words are separated by a stop word, the resulting bigram would be as if the stop word were absent (e.g. 'chalk and cheese' would create the bigram 'chalk cheese'). Then, bigrams are removed if either component word is not common (below 85th quantile of word counts). Among the remaining phrases, only those bigrams with mutual information ranking in the top 5% are retained. Mutual information score is calculated as the ratio of the frequency of the bigram divided by the product of the frequency of the component words. The percentile chosen here is slightly higher than the 1% used by the original paper, as the dataset contained articles, meaning there are far more bigrams than in the dataset used here. For this reason, I chose to consider a higher number of bigrams to ensure completeness. After this filtering step, the rest of the training continues as before. The preparation for bigrams is shown in listing 3.4.

3.4 Out of Sample Testing

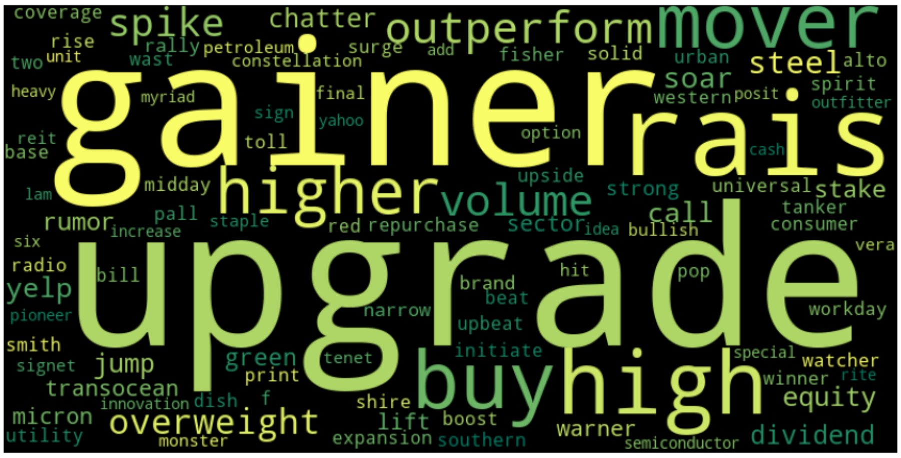
Using the optimally trained model (shown in table 3.1), the articles not used in either training or validation samples are then used to determine the strength of the model. Each market day t, the headlines released from 9 a.m. on day $t-1$ to 9 a.m. on day t are selected and ranked p value calculated from the scoring step (2.2.3). Each ticker in the sample is then ranked according to sentiment of related headlines for that day. If a ticker has multiple headlines, the average sentiment from all related headlines is taken for the firm. From this, a portfolio is created, where the top 50 sentiment stocks are bought, and the lowest 50 sentiment stocks are shorted.

Some constraints are placed on the stocks that can be chosen, to ensure that stocks are not bought when they have negative sentiment. For a stock to be bought, it must have $\widehat{p}_i < 0.5$, and the inverse for a stock to be sold. On the occasion where there are not 50 stocks with positive sentiment, this is to avoid the portfolio purchasing slightly negative stocks, and therefore less than 50 stocks are used in this instance.

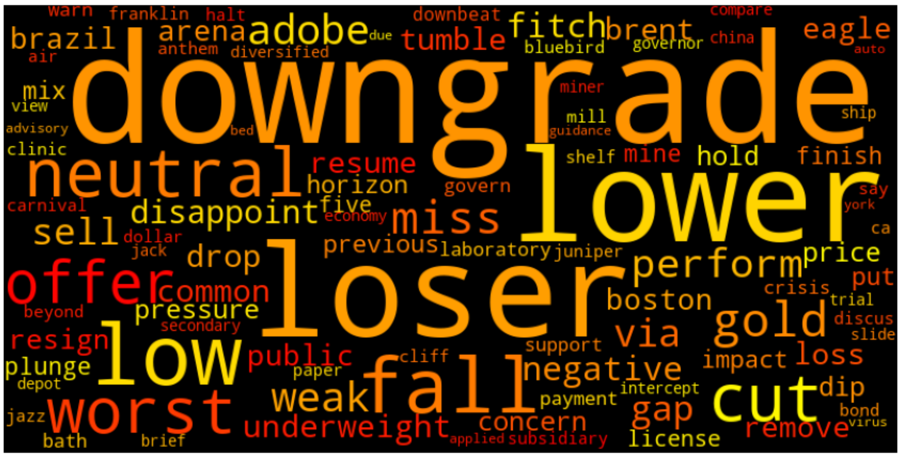
Chapter 4

Critical Evaluation

4.1 Analysis of Word Lists



(a) Positive words



(b) Negative words

Figure 4.1: Word clouds demonstrating sentiment charged words. Font size corresponds to average tone across all training samples

Following the construction of matrix O, figure 4.1 demonstrates the list of sentiment charged words on average over all training 19 windows. At each training and validation window, the sentiment lists are generated completely from scratch, and while there is some overlap, each list can vary significantly. The font size corresponds to the average tone (calculated by $\frac{1}{2}(O_+ - O_-)$) of the words across all windows.

Of the top 50 positive sentiment words, the following appeared in at least 15 of the 19 windows, with words highlighted in **bold** appearing in all windows:

author, rumor, volume, repurchase, rais(e) **gainer, mover, high, upgrade**

The following words appear in at least 15 of the 19 windows with respect to top 50 negative sentiment words with words highlighted in **bold** appearing in all windows:

offer, negative, neutral, public, low **miss, lower, loser, cut, fall, weak downgrade, underweight**

Simply by inspection, each group appears reasonable, in the sense that many of the words with high values in either sentiment could be assumed. However, some words are somewhat surprising and this may offer an insight into subconsious bias that exists in writing headlines as opposed to article bodies. For example, the word *volume* is, under normal circumstances, a sentiment neutral word, but according to the model generated by SESTM, is a highly positive word. Examples of headlines including this word include:

- *Agilent spikes to high of $60.40 on Volume*
- *Markets gather some momentum as volume remains light, geopolitical tension improving*
- *Tuesday's Mid-day options for volume leaders*

Observing these headlines, it is clear that the words are being used in a positive context, and this could be due to subconscious usage of the word when constructing such headlines. However, another explanation could be overfitting. Included in the sample are headlines from 'Benzinga', which is a company that offers realtime news articles, and has a significant quantity of headlines of the form *Benzinga's top upgrades* (around 16,000 headlines from the entire dataset) and *Benzinga's volume* (with around 2,000). This could be seen as an issue of overfitting and may skew these words' sentiment value meaning that it is not reflective of the true sentiment of the word when not used in the context of Benzinga. However, both 'upgrade' and 'volume' appear multiple times in the word lists for bigrams [1] with different combinations of words, meaning that there is positive sentiment attached to these words without the context of Benzinga.

When compared to the Harvard IV and Loughran McDonald dictionaries, we find that the majority of words labelled with sentiment according to our model are not in either dictionary. The negative sentiment words have much higher overlap, with 13 of the top 50 words appearing in the LM dictionary, while only 3 appear in the H4. Conversely, only 6 words overlap LM in the positive tone, while 5 words overlap the H4 dictionary. Furthermore, many words that are included in either dictionary are determined to be sentiment neutral by the model. This is because the model is trained on a sample of headlines and the vocabulary used in headlines is vastly different to that in everyday use or 10-k filings in the case of LM. Headline vocabulary often contains much more impactful words, as it is intended to be a punchy, attention grabbing piece of text. Often, words that are typically used in headlines are rarely found outside of the context of headlines [Reah, 2002]. For this reason, the lexicons of the model, and that of H4 and LM differ.

4.1.1 Bigrams

The order in which words appear in can have a profound effect on the sentiment of a word. This order sentiment can be captured by exploring the dictionary when constructed from *bigrams* as opposed to unigrams alone. By combining both of these dictionaries, it is possible to gain a clearer understanding of the true sentiment of a headline.

4.2 Daily returns

Using the headlines that were saved for out of sample testing, a portfolio is constructed for each day. On average, 353 firms have articles linked to them on a given day, and of these, almost half of these headlines contain one or more sentiment words (are not marked neutral by the model). According to the constraints ($\widehat{p}_i < 0.5$ for a stock to be bought, and vice versa for a stock to be sold), there are some days where less than 100 stocks form the portfolio, in which case we trade with the largest value possible. On average, the long side of the portfolio has 40 stocks, while the short side has 48, therefore the average number of stocks in the portfolio is 88.

[1]Information on bigram computation further on

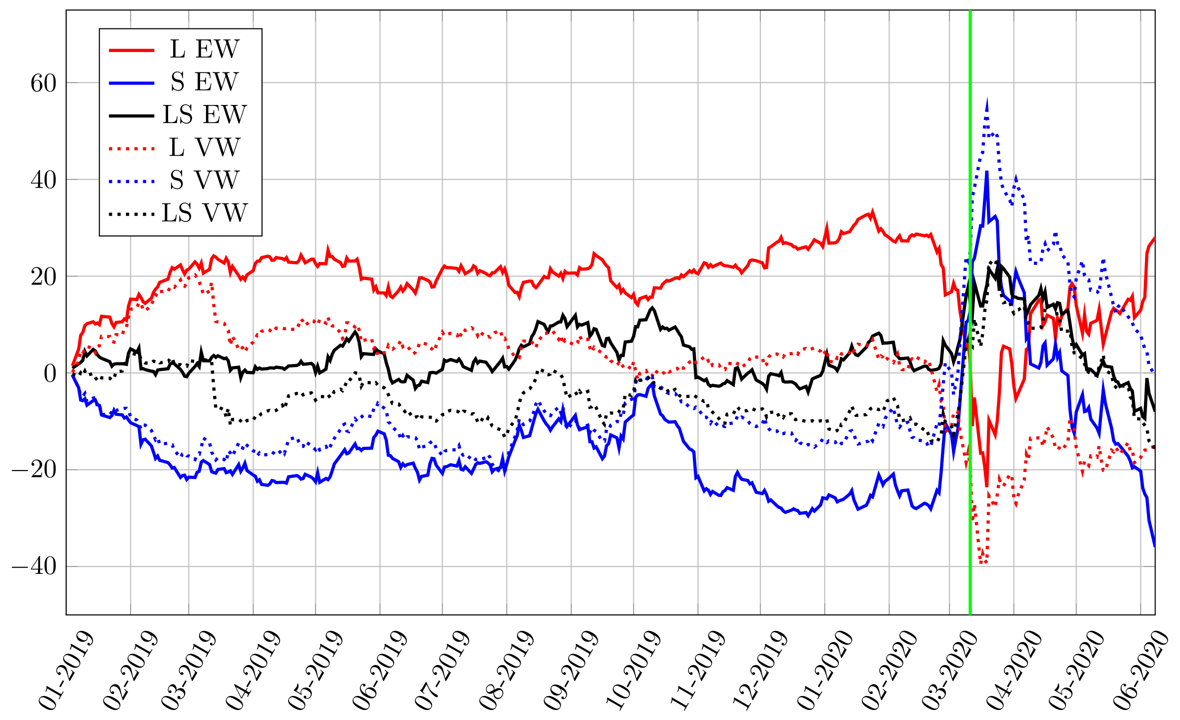


Figure 4.2: Cumulative log returns for each formation over the out of sample headlines

	Sharpe	Average	Daily	FF3		FF5	
Formation	Ratio	Return	Turnover	α	R^2	α	R^2
EW LS	0.19	2.61		0.73	1.46%	1.23	1.62%
EW L	0.7	10.09		8.39	26.6%	8.55	27.32%
EW S	-0.57	-7.47		-8.36	24.29%	-8.02	24.87%
VW LS	-0.14	-0.6		-2.71	4.45%	-2.87	4.47%
VW L	-0.28	-2.51		-5.29	22.04%	-5.89	23.52%
VW S	0.09	1.91		1.88	29.57%	2.32	30.65%

Table 4.1: Performance of Daily News Sentiment Portfolios one day ahead

Table 4.1 describes the performance of the constructed portfolios. The two investment methods (equal and value weighted) are split up into the Long-Short combined portfolio (L-S), and the long (L) and short (S) legs are also displayed separately for comparison purposes. The daily turnover section displays the average turnover each day, which would be 100% as the profit is liquidated at the end of each day, but some stocks are retained the following day. A turnover of 90% (as in VW L) implies that on average 1 in 10 stocks are retained the following day. This could be due to headlines or news articles that are concerning the same events (stale news), or repeat sentiment headlines as a story unfolds over a number of days.

The FF3 and FF5 sections refer to Fama French 3 and 5 factor regression respectively, while the α concerns the intercept. The higher percentage of the average returns that the α value is refers to the amount of private information held in the investment. In other words, if the α is a very small percentage of the generated returns, then the returns that an investment has generated can be explained by regular movement in the markets, and there is no private information that is being used to generate profit.

This figure clarifies identifies three key facts: firstly, none of these formations are profitable, with the only formation that is profitable being the equal weighted long strategy with a Sharpe ratio of 0.7, indicating that the profit versus risk ratio is beneficial. The second fact that the equal weighted formation outperforms the value weighted formation significantly in the long leg, while the short leg favours the value weighted formation. Fundamentally, this means that the trained model is better at detecting positive sentiment about smaller stocks than that of larger stocks, while also being better at detecting negative sentiment in larger stocks than smaller stocks. This is due to the nature of headlines, as, while they are intended to summarise the contents of a news article, the language used favours information that is likely to generate clicks and views. Small stocks performing well and large, supposedly stable stocks performing poorly are more likely to incentivise a user to click on the full article than their respective counterparts, meaning these are more likely to be included in the headline itself. Furthermore, the risk of the market lies with the party who shorts a stoc

Figure 4.2 details the cumulative log returns over the entire out of sample dataset. Here, The performance of each of the legs are relatively steady in their respective directions until March 2020. At this point, the profits of the short legs skyrocket, while that of the long section plummet — especially for the value weighted portfolios. This is due to the Coronavirus outbreak, as it was officially declared a pandemic on March 11 2020, indicted by the vertical green line on the graph. This, of course, caused stocks to crash worldwide, and was felt particularly by large stocks, before recovering a short while later. This also clearly highlights a limitation with all lexicon based sentiment analysis methods, whether they use supervised learning signals, or use a manually labelled dictionary: they are unable to adapt to rapid changes. Since the COVID-19 virus did not exist during the training and validation samples, the model has no information on the sentiment of headlines that would discuss this, and would ignore it. Naturally, if the model was retrained using data from this time period, it would be able to detect such headlines in the future, but the crux of the issue is that it is impossible to obtain enough information to allow the model to react to such drastic changes in the market.

4.3 Speed of information Assimilation

In the previous sections, we explore the relationship between the sentiment score of a headline calculated by the model and the changes in price the following day. Here, we explore the relationship between the changes in price after differing delays to investigate timing responses.

	Sharpe	Average	Daily	FF3		FF5	
Formation	Ratio	Return	Turnover	α	R^2	α	R^2
			Day $t-1$				
EW LS	15.53	258.76		259.78	4.92%	259.21	7.6%
EW L	7.23	138.28		140.4	7.78%	140.33	9.12%
EW S	7.84	120.48		118.68	29.62%	118.18	29.78%
VW LS	12.72	164.11		164.44	3.4%	163.49	6.23%
VW L	5.26	76.57		76.11	14.2%	76.35	15.76%
VW S	6.58	87.55		87.63	28.88%	86.44	29.39%
			Day $t+0$				
EW LS	10.0	113.75		110.15	4.11%	110.42	4.25%
EW L	2.5	34.18		31.59	23.47%	32.09	24.37%
EW S	4.83	79.57		77.87	23.08%	77.64	23.36%
VW LS	9.61	95.39		93.13	4.79%	93.42	4.93%
VW L	2.78	30.1		27.12	30.44%	27.53	31.03%
VW S	4.77	65.29		65.31	24.58%	65.2	25.13%
			Day $t+1$				
EW LS	0.19	2.61		0.73	1.46%	1.23	1.62%
EW L	0.7	10.09		8.39	26.6%	8.55	27.32%
EW S	-0.57	-7.47		-8.36	24.29%	-8.02	24.87%
VW LS	-0.14	-0.6		-2.71	4.45%	-2.87	4.47%
VW L	-0.28	-2.51		-5.29	22.04%	-5.89	23.52%
VW S	0.09	1.91		1.88	29.57%	2.32	30.65%

Table 4.2: Performance of Daily News Sentiment Portfolios day $t-1$ to day $t+1$

Table 4.2 portays the returns of different holding lengths on returns. For day $t-1$ and $t+0$, these are purely theoretical and serve only as an insight into the models performance.

The day $t-1$ section explores portfolios created the day before a headline is released. The portfolios are constructed in the same manner as table 4.1, however, the theoretical portfolios crafted here are using headlines that have not yet been released. By doing so, we investigate how estimated sentiment score $\widehat{p}_i$ picks up on stale news. This is reflected in the entirely infeasible Sharpe ratios of 15.53 and 12.72 for equal and value weighted portfolio formations respectively.

The day $t+0$ section explores a portfolio crafted on the same day as the headline is released, providing an insight into how the estimated sentiment score picks up on fresh news.

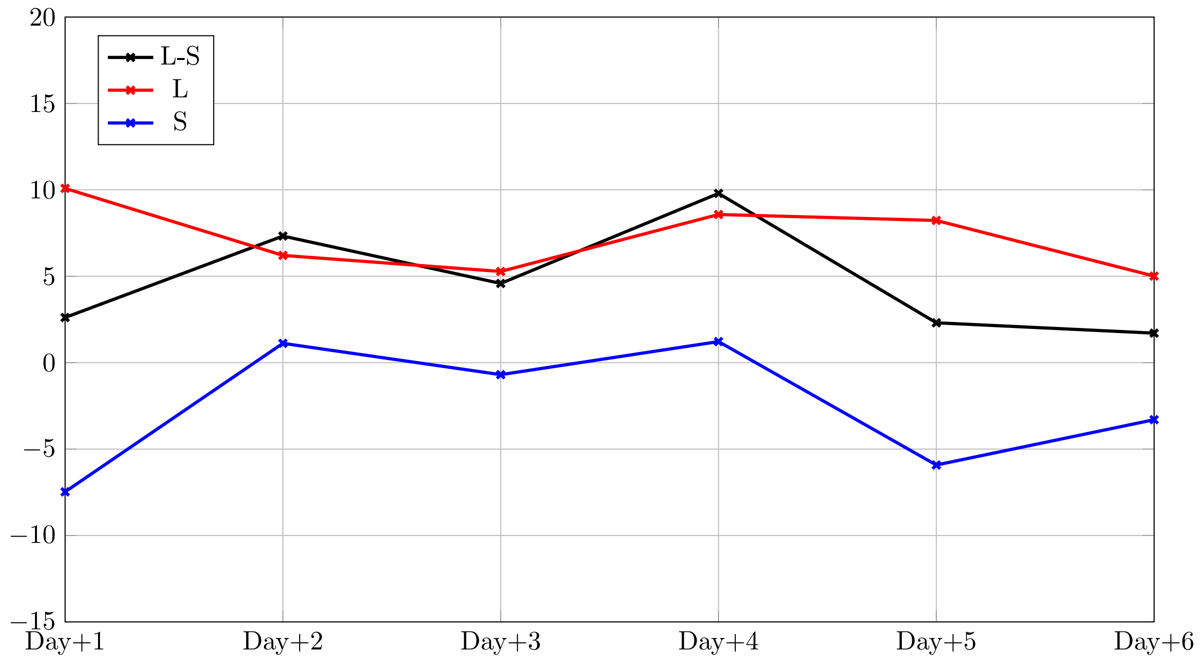


Figure 4.3: Average daily returns for different holding periods. If article is released on day t, portfolio is created on day $t + (n - 1)$ and sold on day $t + n$ where n is the value indicated by the x axis

4.4 Comparison to other methods

4.5 What to do

A topic-specific chapter

This chapter is intended to evaluate what you did. The content is highly topic-specific, but for many projects will have flavours of the following:

1. functional testing, including analysis and explanation of failure cases,
2. behavioural testing, often including analysis of any results that draw some form of conclusion wrt. the aims and objectives, and
3. evaluation of options and decisions within the project, and/or a comparison with alternatives.

This chapter often acts to differentiate project quality: even if the work completed is of a high technical quality, critical yet objective evaluation and comparison of the outcomes is crucial. In essence, the reader wants to learn something, so the worst examples amount to simple statements of fact (e.g., "graph X shows the result is Y"); the best examples are analytical and exploratory (e.g., "graph X shows the result is Y, which means Z; this contradicts [1], which may be because I use a different assumption"). As such, both positive *and* negative outcomes are valid *if* presented in a suitable manner.

Chapter 5

Conclusion

The concluding chapter of a dissertation is often underutilised because it is too often left too close to the deadline: it is important to allocate enough attention to it. Ideally, the chapter will consist of three parts:

1. (Re)summarise the main contributions and achievements, in essence summing up the content.
2. Clearly state the current project status (e.g., "X is working, Y is not") and evaluate what has been achieved with respect to the initial aims and objectives (e.g., "I completed aim X outlined previously, the evidence for this is within Chapter Y"). There is no problem including aims which were not completed, but it is important to evaluate and/or justify why this is the case.
3. Outline any open problems or future plans. Rather than treat this only as an exercise in what you *could* have done given more time, try to focus on any unexplored options or interesting outcomes (e.g., "my experiment for X gave counter-intuitive results, this could be because Y and would form an interesting area for further study" or "users found feature Z of my software difficult to use, which is obvious in hindsight but not during at design stage; to resolve this, I could clearly apply the technique of Smith [7]").

Bibliography

[Fama and French, 1992] Fama, E. F. and French, K. R. (1992). The cross-section of expected stock returns. *the Journal of Finance*, 47(2):427–465.

[Fama and French, 2015] Fama, E. F. and French, K. R. (2015). A five-factor asset pricing model. *Journal of financial economics*, 116(1):1–22.

[Felmeden, 2022] Felmeden, J. (2022). Github.

[French, 2022] French, K. (2022). Kenneth r. french - data library.

[Hofmann, 2013] Hofmann, T. (2013). Probabilistic latent semantic analysis. *arXiv preprint arXiv:1301.6705*.

[Hours, 2022] Hours, T. (2022). Is the stock market open?

[Ke et al., 2019] Ke, Z. T., Kelly, B. T., and Xiu, D. (2019). Predicting returns with text data. Technical report, National Bureau of Economic Research.

[Kirange et al., 2016] Kirange, D., Deshmukh, R. R., et al. (2016). Sentiment analysis of news headlines for stock price prediction. *Composoft, An International Journal of Advanced Computer Technology*, 5(3):2080–2084.

[Loughran and McDonald, 2011] Loughran, T. and McDonald, B. (2011). When is a liability not a liability? textual analysis, dictionaries, and 10-ks. *The Journal of Finance*, 66(1):35–65.

[Reah, 2002] Reah, D. (2002). *The language of newspapers*. Psychology Press.

[Sharpe, 1966] Sharpe, W. F. (1966). Mutual fund performance. *The Journal of business*, 39(1):119–138.

[Teweles and Bradley, 1998] Teweles, R. J. and Bradley, E. S. (1998). *The stock market*, volume 64. John Wiley & Sons.

Appendix A

Appendix

Content which is not central to, but may enhance the dissertation can be included in one or more appendices; examples include, but are not limited to

- lengthy mathematical proofs, numerical or graphical results which are summarised in the main body,
- sample or example calculations, and
- results of user studies or questionnaires.

Note that in line with most research conferences, the marking panel is not obliged to read such appendices. The point of including them is to serve as an additional reference if and only if the marker needs it in order to check something in the main text. For example, the marker might check a program listing in an appendix if they think the description in the main dissertation is ambiguous.

Appendix B

List of Optimisations

Appendix C

Word and Phrase lists

Positive					**Negative**				
Word	Sentiment	Count	LM	H4	Word	Sentiment	Count	LM	H4
upgrade	0.016795	19	0	1	downgrade	-0.023946	19	1	0
gainer	0.013524	19	0	0	loser	-0.016851	19	0	1
high	0.010034	19	0	0	lower	-0.016773	19	0	0
mover	0.007526	19	0	0	fall	-0.004345	19	0	0
rais	0.011851	18	0	0	cut	-0.003035	19	1	0
repurchase	0.000298	17	0	0	miss	-0.001481	19	1	0
volume	0.002742	16	0	0	weak	-0.001191	19	1	0
rumor	0.00078	16	0	0	underweight	-0.000751	19	0	0
author	6.8e-05	16	0	0	low	-0.005291	17	0	0
higher	0.006567	15	0	0	public	-0.000609	17	0	0
outperform	0.002857	15	1	0	neutral	-0.003327	16	0	0
spike	0.002189	15	0	0	offer	-0.002486	16	0	0
solid	0.000432	15	0	0	negative	-0.000794	16	1	1
buy	0.008344	14	0	0	disappoint	-0.000759	15	1	0
green	0.000711	14	0	0	common	-0.000731	15	0	0
overweight	0.001675	13	0	0	concern	-0.000485	15	1	0
soar	0.001171	13	0	0	loss	-0.000699	14	1	1
strong	0.000545	13	1	0	remove	-0.000593	14	0	0
lift	0.000659	12	0	0	impact	-0.000458	14	0	0
jump	0.000856	11	0	0	tumble	-0.000678	13	0	0
special	0.000154	11	0	1	resign	-0.000534	13	1	0
strength	0.000147	11	1	0	dip	-0.000478	13	0	0
mention	9.7e-05	11	0	0	drop	-0.00074	12	0	0
chatter	0.000873	10	0	0	resume	-0.000661	12	0	0
stake	0.000815	10	0	0	pressure	-0.000469	12	0	0
narrow	0.000471	10	0	0	shelf	-0.00027	12	0	0
pop	0.000359	10	0	0	worst	-0.002948	11	1	1
boost	0.000324	10	1	0	sell	-0.000961	11	0	0
dynamic	5.2e-05	10	0	1	secondary	-0.000194	11	0	0
expansion	0.000243	9	0	0	adobe	-0.001185	10	0	0
steel	0.001328	8	0	0	perform	-0.001148	10	0	0
dividend	0.000862	8	0	0	fitch	-0.000813	10	0	0
micron	0.000665	8	0	0	plunge	-0.000421	10	0	0
rally	0.00044	8	0	1	valuation	-0.000143	10	0	0
base	0.000407	8	0	0	lose	-7.9e-05	10	1	0
beat	0.000404	8	0	0	laboratory	-0.000361	9	0	0
f	0.000321	8	0	0	warn	-0.000352	9	1	0
southern	0.000314	8	0	0	downbeat	-0.00029	9	0	0
upside	0.000271	8	0	1	beyond	-0.00023	9	0	0
final	0.000216	8	0	0	halt	-0.000225	9	1	0
add	0.000191	8	0	0	prelim	-0.00012	9	0	0
outfitter	0.000167	8	0	0	four	-3.1e-05	9	0	0
unconfirm	0.00014	8	0	0	gap	-0.000914	8	0	0
test	2.1e-05	8	0	0	price	-0.000484	8	0	0
proceed	0.0	8	0	0	mix	-0.000447	8	0	0
yelp	0.001107	7	0	0	bath	-0.000343	8	0	0
call	0.001105	7	0	0	paper	-0.000232	8	0	0
warner	0.000664	7	0	0	downside	-0.000129	8	0	0
transocean	0.000651	7	0	0	delay	-8.2e-05	8	1	0
surge	0.000343	7	0	1	loan	-5.1e-05	8	0	0

Table C.1: Top sentiment words for each polarity, along with appearance in either Loughran McDonald dictionary (LM) or Harvard IV psychological dictionary (H4). Note sentiment in this case refers to average *tone* over all 20 training windows. Words are first sorted via count of training windows appeared in, and then by sentiment