DEPARTMENT OF COMPUTER SCIENCE

# Sentiment 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.

May 5, 2022

# Abstract

In this project, I implement Zheng Ke, Bryan Kelly, and Dacheng Xiu's novel text sentiment extraction algorithm and evaluate its performance when considering a dataset of headlines. I extract information from headlines and use a supervised learning framework to construct a score, which is later used for stock movement prediction. In the study of the success of this algorithm, I use data from Yahoo Finance and compare predictive results of prediction from this algorithm against baseline lexicons. My research hypothesis is that this algorithm will be able to more accurately predict the movement of stocks based on the headlines on the previous day. I hav ealso included data obtained from hte experiments conducted, allowing for verification of the methods used.

## List of Achievements

- I wrote more than 3000 lines of source code in Python for pre-processing data, constructing the algorithm, and testing by creating portfolios
- I adapted the methods explored by Ke, Kelly, and Dacheng to accommodate for headline input, rather than article bodies.
- I designed experiments to test the predictive success the algorithm.
- I implemented the algorithm and spent considerable time optimising the code to run as efficiently as possible
- I trained a number of different models to see the effect of differing setups

# Dedication and Acknowledgements

I would first like thank my family for your continued encouragement and support for my entire university life. I would also like to thank my friends for supporting me, but also making me laugh when I needed it most.

I would also like to mention Mr. Richard Green, my secondary school computer science teacher, without who I would not be pursuing the field today.

Finally, I would like to thank Rami Chehab, who assisted in my research greatly and inspired the vast quantity of this project. I could not have done this without all of you.

# Declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Taught Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, this work is my own work. Work done in collaboration with, or with the assistance of others, is indicated as such. I have identified all material in this dissertation which is not my own work through appropriate referencing and acknowledgement. Where I have quoted or otherwise incorporated material which is the work of others, I have included the source in the references. Any views expressed in the dissertation, other than referenced material, are those of the author.

Joshua Felmeden, May 5, 2022

# Contents

# List of Figures

# List of Tables

# Ethics Statement

This project did not require ethical review, as determined by my supervisor, Dr. Rami Chehab.

# Supporting Technologies

- I used a sample of financial 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, stop words data, and stemmers and lemmatizers (https://www.nltk.org/)
- I used the yfinance library on Python to pull stock data (https://pypi.org/project/yfinance/)
- I used Refinitiv Eikon to pull outstanding stock data (https://www.refinitiv.com/en/products/eikon-trading-software#overview)
- I used Jupyter Notebook as an interactive development environment (https://docs.jupyter.org/en/latest)

# Notation and Acronyms

| | | |
|---|---|---|
| NLP | : | Natural Language Processing |
| SESTM | : | Semantic Extraction via Screening and Topic Modelling |
| $\|x\|$ | : | Size of $x$ |
| TF-IDF | : | Term frequency inverse document frequency |
| EST | : | Eastern standard time |
| UTC | : | Coordinated universal time |
| | | |
| **Fama French Factors** | | |
| CAPM | : | Capital asset pricing model |
| $R_i$ | : | Return of investment |
| $R_f$ | : | Risk free rate |
| SMB | : | Small minus big |
| HML | : | High minus low |
| RMW | : | Robus minus weak |
| CMA | : | Conservative minus aggressive |
| | | |
| **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 Financial Markets and Stock Exchanges

A stock exchange is a platform on 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).

Stock excahnges are only active on certain days and for specific times. Opening times vary on from exchange to exchange, but the majority 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 (TradingHours, 2022). Of course, the opening and closing times are in the local time for each market. For American based exchanges, this is in Eastern Standard Time (EST).

For the entire duration of these opening times, anyone is able to buy or sell stocks, usually through the medium of a stockbroker. Many investigations have been conducted into uncovering information that will help an individual make more informed investments and ultimately increase their potential profits from such investments.

## 1.2 Financial News

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. Textual analysis of this type usually consists of pre-defined dictionaries, and few conduct any statistical analysis of the text.

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, Deshmukh, et al., 2016). The lower word counts of headlines mean processing the data is substantially quicker, and collecting large quantities of headlines is easier.

## 1.3 Motivation

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 can be utilised to predict future returns. Ke, Kelly, and Xiu (2019) proposed a novel text mining algorithm that observed historic articles and their subsequent effect on associated firm share prices, and retroactively assigned sentiment to words in each article. In doing so, they 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 is both more readily available and shorter in length than full article bodies.

The analysis of headlines in particular has seen less attention than analysing entire articles and the majority of approaches have stayed rooted in utilising labelled dictionaries to glean sentiment from such sources (Kirange, Deshmukh, et al., 2016), (Nemes and Kiss, 2021), with some using slightly more complex approaches, such as regression (John and Vechtomova, 2017).

I believe that it is possible to harness sentiment portrayed in headlines and utilise the sentiment assignation method described by Ke et al. to craft a more accurate sentiment analysis model that can be used to predict stock movement based on headlines with greater success than existing methods or approaches.

## 1.4 Aims and Objectives

The aims and objectives for this project are the following:

1. Implement and optimise the algorithm proposed by Ke, Kelly, and Xiu, 2019.
2. Train a model 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

I 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 Sentiment Analysis

Sentiment 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 profound and its application could prove valuable in a variety of fields, especially 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 throughout previous decades.

The success or failure of any approach to this task can be summarised as the closeness of a prediction of sentiment to that calculated by a human. Take, for example, the following sentences:

1. *'The movie was great'*
2. *'The movie was terrible'*
3. *'The acting was terrible, but the movie was great'*

Sentences 1 and 2 have very simple sentiment: one is positive, and the other is negative. However, sentence 3 has slightly more complex sentiment. If the feature of interest is the movie, the word *'terrible'* does not influence the sentiment of this feature as it is used in reference to the acting. Simpler analysis methods may not pick up on such nuances in language.

### 2.1.1 Pre processing

Here, I introduce three terms used in the pre-processing of text data. The main goal of pre-processing in sentiment analysis is to ensure sentiment is being assigned to the correct words. Stop words are a term used in NLP to describe very commonly used words that are considered unimportant (such as 'and' or 'the'). They serve only as noise and are removed to focus on more impactful words.

Stemming and lemmatisation are two techniques whose primary is to reduce different forms of a word to its common base form. For example, 'am, are, is' all become 'be' as it is the root of the word (Manning, Raghavan, and Schütze, 2010). The two techniques share a common goal, but achieve this in different ways. Stemming refers to a more crude process that is effective at the potential cost of precision. In general, it consists of applying a series of rules sequentially, each of which reducing a word. For each reduction, there are conventions to select certain rules. There are a variety of stemmers that each perform stemming in a unique way. On the other hand, lemmatisers are a tool from NLP that does analysis to identify the *lemma*, or the language word root. Both techniques have advantages and disadvantages and as a result the two are often used in tandem, as they compliment one another.

The textual input to any sentiment analysis algorithm requires high quality preprocessing to ensure that the data is of the same format, and is one of the primary steps in any technique. Keeping noisy features such as stop words, or HTML tags serves to muddy the output. Instead, only features of interest should be considered in the ideal scenario (Haddi, Liu, and Shi, 2013). Insufficient data pre-processing can lead to underachieving analysis models, and as such the importance of this step should not be overlooked. Traditionally, this consists of online text cleaning (if necessary), removal of stop words and whitespace, and stemming and lemmatisation.

### 2.1.2 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) (Taboada et al., 2011). 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. An example of a lexicon 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.

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 (2011) conducted an investigation into the use of standard lexicons for 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. Some words were found to have strong links to certain industry specific sectors, meaning they had no relevance without the context of a specific industry sector. Loughran and McDonald created their own lexicon based on these results called the Loughran McDonald dictionary specifically 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.

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 sentiment conveyed can be inverted entirely. For this reason, simply calculating the sentiment of a piece of text on a word by word level can give an incorrect estimate. Additionally, sentiment analysis is extemely reliant on sufficiently labelled data and lack of this data is a significant obstacle in the field (Madhoushi, Hamdan, and Zainudin, 2015).

#### 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. There are many other 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 conjunction, 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.

### 2.1.3 Sentiment Assignation based on Previous Data

Lexicon based approaches to sentiment analysis historically consist of a simple key value structure, assigned by an individual or group. However, a more advanced method of labelling words can lead to more accurate results. Tetlock (2007), Tetlock, Saar-Tsechansky, and Macskassy (2008), and Loughran and McDonald (2011) all examine the market reaction to the tone of newspaper headlines and 10-K filings, and concluded that such media evokes a market reaction relative to the number of positive, negative, and total words in a document. These approaches only consider a singular, consistent positive or negative value for each entry, and it is likely that some words are more impactful than others. Jegadeesh and Wu (2013) developed a new method that assigns word weight based on previous market reactions to 10-K filings. This method is more flexible than the more rudimentary dictionary methods discussed earlier, and shows promise that such a method could be applied more generally.

## 2.2 Sentiment Extraction via Screening and Topic Modelling

Sentiment Extraction via Screening and topic model (SESTM) is a novel text mining algorithm that makes use of a teaching signal developed by Ke, Kelly, and Xiu (2019). It is based on a similar method proposed by Jegadeesh and Wu (2013), utilising 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 headlines.

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 challenge is to 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 as sentiment score increases, the probability a headline has of returning positive returns also increases.

Next, we observe the distribution of word counts in an headline. 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 sentiment 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 is 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 this distribution, 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 headlines, 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, under the assumption that if a word appears in more headlines that result in positive returns than those with negative returns, 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 $n$ headlines producing a dictionary of $m$ words.
- The word count of headline $i$ 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 headline $i$ is defined as the submatrix $d_{[S],i}$.

For a word $j$, we define $f_j$ as the ratio of positive headlines to total headlines it appears in:

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

Here, $sgn(y)$ is simply the sign of the difference in tagged returns of an headline. Let $y$ be the tagged return of an headline:

$$sgn(y) = \begin{cases} +1 & \text{if } y >= 0 \\ -1 & \text{otherwise} \end{cases} \tag{2.9}$$

If an headline is released on day $t$ (specifically, between 4pm of day $t-1$ and 4pm of day $t$), the headline 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 headlines.

Next, we calculate the positive and negative thresholds for $f_j$. We define $\hat{\pi}$ to be the fraction of headlines that have $sgn(y) = +1$. These are headlines 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 ($\alpha_+$ and $\alpha_-$) such that any word with $f_j \geq \hat{\pi} + \alpha_+$ is determined to be sentiment positive, and the inverse for $f_j \leq \hat{\pi} - \alpha_-$. This filtering technique accepts words that are further away from the expected average sentiment, meaning only those words with non-neutral sentiment are included. To ensure that words appearing infrequently do not skew sentiment values significantly (for example a word appearing in exactly one headline that is maximally positive ending up with $f_j = 1$), we introduce a third parameter $\kappa$, restricting words that appear in fewer than $\kappa$ headlines. The number of headlines in which word $j$ appears is defined as $k_j$. The set of sentiment charged words is therefore defined by:

$$S = \{j : f_j \geq \hat{\pi} + \alpha_+ \cup f_j \leq \hat{\pi} - \alpha_-\} \cap \{j : k_j \geq \kappa\} \tag{2.10}$$

The three parameters introduced here $(\alpha_+, \alpha_-, \kappa)$ are hyperparameters and are tuned via cross-validation (see section 3.3.1 for usage). The best choice for each parameter is selected based on minimising a cost function.

### 2.2.2 Learning Sentiment Topics

As the wordlist $S$ has been 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 headline 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:

$$p_i = \frac{\text{rank of } y_i \text{ in } \{y_l\}_{l=1}^{n}}{n} \tag{2.11}$$

where the return of a headline is represented by $y$, and headlines are ranked in ascending order (meaning an headline 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 headline $i$. Using model 2.6, we know that the expected distribution of these sentiment word frequencies can be modelled as:

$$\mathbb{E}h_i = \mathbb{E}\frac{d_{[S],i}}{s_i} = p_i O_+ + (1 - p_i) O_- \tag{2.12}$$

or in matrix form:

$$\mathbb{E}H = OW, \quad \text{where } W = \begin{bmatrix} p_1 & \cdots & p_n \\ 1 - p_1 & \cdots & 1 - p_n \end{bmatrix}, \text{ and } H = [h_1, h_2, \ldots, h_n] \tag{2.13}$$

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 $S$:

$$h_i = \frac{d_{[S],i}}{s_i} \quad \text{where } s_i = \sum_{j \in S} d_{j,i} \tag{2.14}$$

$W$ is estimated using the ranks of returns described in 2.11, leading to the final representation of:

$$O = [h_1, h_2, \ldots, h_n] W'(WW')^{-1}, \quad \text{where } W = \begin{bmatrix} p_1 & p_2 & \cdots & p_n \\ 1 - p_1 & 1 - p_2 & \cdots & 1 - p_n \end{bmatrix} \tag{2.15}$$

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

### 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 headline 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:

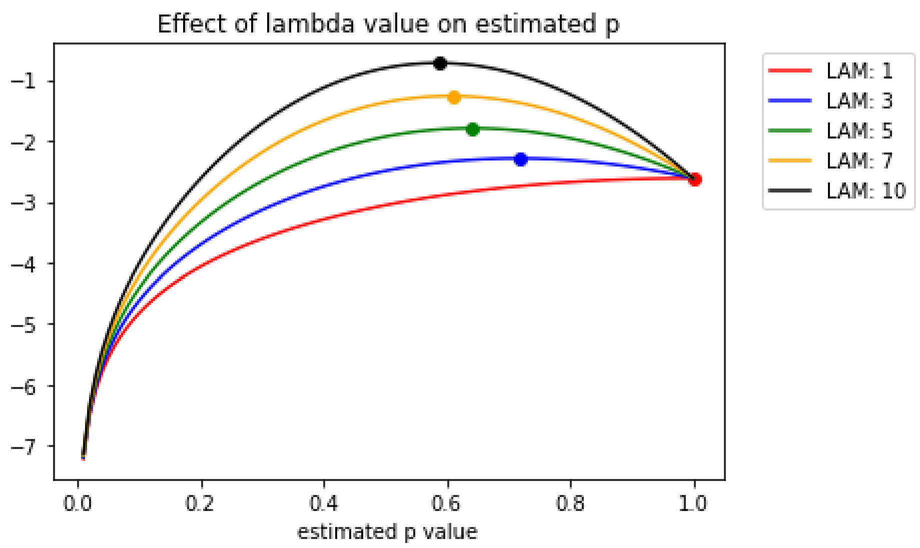


Figure 2.1: Effect of different values of $\lambda$ on estimated $p$ value. The large point on each line is the maximum value, and reflects the returned estimated $p$ of a headline

$$\widehat{p} = \arg\max_{p\in[0,1]} \left\{ \widehat{s}^{-1} \sum_{j\in S} \log(pO_{+,j} + (1-p)O_{-,j}) + \lambda \log(p(1-p)) \right\} \tag{2.16}$$

where $\widehat{s}$ is the total count of word from $S$ in the new headline, $d_j, O_{+,j}, O_{-,j}$ are the entries for word $j$ in the corresponding vectors and $\lambda$ 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 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 positive or negative sentiment and buying the most positive stocks while selling the negative ones. This gives a good indication of the predictive power of a word list.

### 2.3.1 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 are usually very slight, 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:

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

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:

$$\frac{S_t}{S_{t+n}} - 1 \tag{2.18}$$

Of course, these simple formulae neglect 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 three major strategies that can be used to construct a portfolio: equal weighting, price weighting, and value weighting. Each strategy has its own unique set of advantages and disadvantages.

The equal weighted is perhaps the most simple 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.

Price 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 $P = [s_1, \dots, s_n]$ has market value $P_{i,t}$ at time $t$, the weight of stock $s_i$ would be:

$$w_{i,t} = \frac{P_{i,t}}{\sum_n^1 P_{n,t}} \tag{2.19}$$

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

Finally, value weighted portfolios combine both the price of the stock with the number of outstanding stocks, which creates a much larger bias towards large companies. As before, if stock $s_i$ in portfolio $P = [s_1, \dots, s_n]$ has market value $P_{i,t}$ at time $t$, and has $s_o$ outstanding stocks, the weight of stock $s_i$ is:

$$w_{i,t} = \frac{P_{i,t} \times s_o}{\sum_n^1 P_{n,t}} \tag{2.20}$$

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. Price weighted construction lies somewhere in the middle of the two, being relatively simple to construct, as with equal weighting, but with slight consideration to the price of a stock. The downside of this constructon is that the weights assigned are somewhat arbitrary as the price of a stock can fluctuate greatly.

### 2.3.2 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 K. R. French, 1992). CAPM is used for describing systematic risk and expected return, especially for that in stocks. The equation for this is:

$$ER_i = R_f + \beta_t(ER_m - R_f) \tag{2.21}$$

where $ER_i$ is the expected return of the investment, $R_f$ is the risk free rate, $\beta_i$ 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 encompasses 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 (K. 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 $\beta$s detail the exposure to exposure to each of the risk factors while the $\alpha$, 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 revisited by Fama and French in 2015 where they observed two additional factors: robust minus weak (RMW) and conservative minus aggressive (CMA) (Fama and K. R. 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:

$$ER_i = R_f + \beta_1(ER_m - R_f) + \beta_2(SMB) + \beta_3(HML) + \beta_4(RMW) + \beta_5(CMA) + \alpha \tag{2.22}$$

# 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, Kelly, and Xiu, 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 Overview

### 3.1.1 Main Idea

The main concept of this project is to test the robustness of Ke, Kelly and Xiu's SESTM algorithm by attempting to use it with a dataset of headlines. This dataset would contain text that is much shorter than the full body of the headline and therefore could create some obstacles. I planned to begin the project by thoroughly reading and understanding the original literature, as I planned on implementing the algorithm, and to do so without full prior knowledge of the working would be impossible. After familiarising myself with the algorithm, I intended to locate an appropriate dataset and preprocess it accordingly. From this, I could begin the process of training the algorithm. Finally, with a model trained, I planned to design and execute similar experiments as were carried out in the original papers, and observe the results. As a point of reference, I also chose to compare the results with that of similar methods, two lexicons: Harvard IV and Loughran McDonald. From these results, I could analyse and conclude the success or difficulties the algorithm has in predicting returns using headlines as input.

### 3.1.2 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 part as an individual piece of code, rather than running and debugging an entire script each time. This streamlined the debugging phase of the project.

The full notebook along with accompanying datasets can be found on GitHub (Felmeden, 2022). I opted to use Git as version control for the project. It is used globally and I have previous experience with Git, therefore to keep track of changes made, and allow for fallbacks when problems were encoutered, I used Git.

```
        {
                "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 python dictionary headline entry

## 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 a Python dictionary for future usage. An example dictionary 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 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 headlines 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 alphabet 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 focus on more important news.

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

[2] https://pypi.org/project/yfinance/

[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/

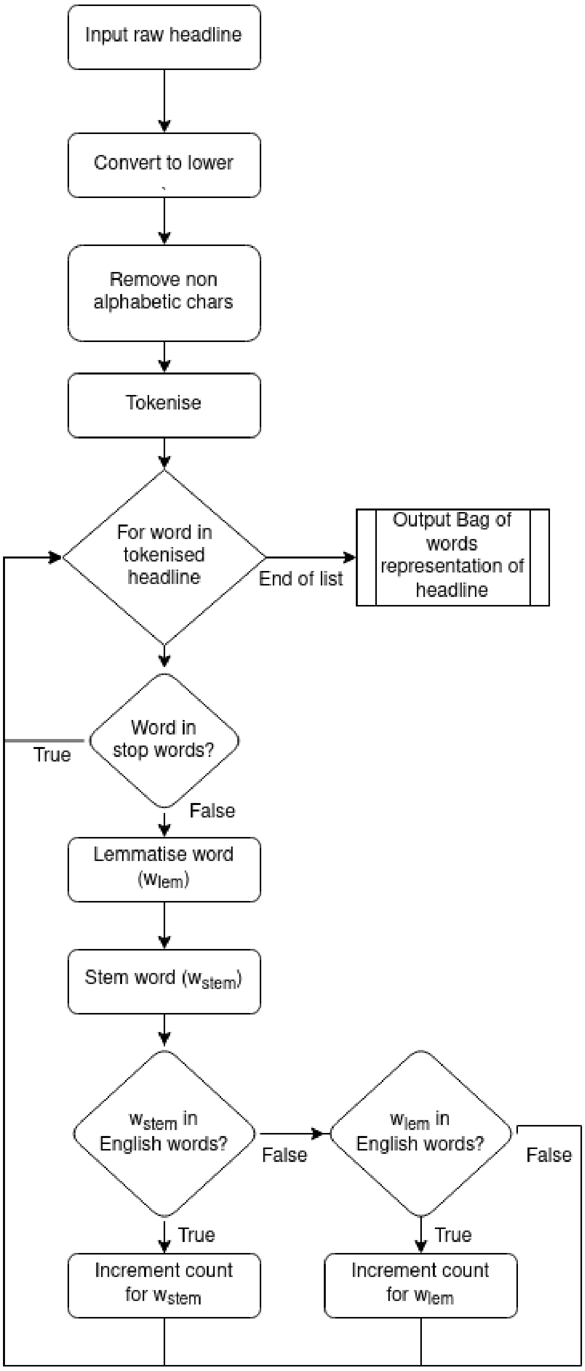

Input raw headline
Convert to lower
Remove non alphabetic chars
Tokenise
For word in tokenised headline
End of list
Output Bag of words representation of headline
Word in stop words?
True
False
Lemmatise word ($w_{lem}$)
Stem word ($w_{stem}$)
$w_{stem}$ in English words?
False
$w_{lem}$ in English words?
False
True
True
Increment count for $w_{stem}$
Increment count for $w_{lem}$

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

```
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

- 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)

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 sentment 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 Implementing the Algorithm

Having sourced the headlines, I was able to start implementing SESTM in Python. As I did not have access to the data used in the original paper, I could not simply input data and check if similar results were obtained. Instead, I created some dummy headlines and checked that the output result was as expected for each section of the algorithm. This method of working helped solidify the content of the original paper and aided my familiarisation with the algorithm. For these examples, I chose $\kappa = 0, \lambda = 1, \alpha_- = 0.2$ and

[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/

$\alpha_+ = 0$. This is because the sample size is incredibly small, and I did not want to remove any of the words from the set of sentiment words for low frequency in this example.

```
headline_1 = {
    'date': '2021-12-23 12:58:45.061000+00:00',
    'ticker': 'FDX',
    'mrkt_info': {
        'open': 233.7,
        'close': 200.3
    },
    'headline': 'Josh likes to watch football games'
}

headline_2 = {
    'date': '2022-01-26 07:11:46.774000+00:00',
    'ticker': 'ABDN',
    'mrkt_info': {
        'open': 229.2,
        'close': 241.0
    },
    'headline': '<p>Mary also likes to watch football games and films. She prefers films.</p>'
}

headline_3 = {
    'date': '2021-10-25 13:22:07.985000+00:00',
    'ticker': 'ABDN',
    'mrkt_info': {
        'open': 250.3,
        'close': 258.5
    },
    'headline': '<p>Carl likes to play football. He finds films boring.</p>'
}
```

Listing 3.3: Example input

| Word | $O_+$ | $O_-$ | $f_j$ |
|---|---|---|---|
| josh | 0.0 | 0.3111 | 0.0 |
| like | 0.1053 | 0.2778 | 0.6667 |
| football | 0.1053 | 0.2778 | 0.6667 |
| mary | 0.1128 | 0.0 | 1.0 |
| also | 0.1128 | 0.0 | 1.0 |
| film | 0.2707 | 0.0 | 1.0 |
| prefer | 0.1128 | 0.0 | 1.0 |
| carl | 0.0451 | 0.0333 | 1.0 |
| play | 0.0451 | 0.0333 | 1.0 |
| find | 0.0451 | 0.0333 | 1.0 |
| bore | 0.0451 | 0.0333 | 1.0 |

Table 3.1: Resultant output from example headlines

As shown in listing 3.3 and table 3.1, the matrix $O$ is as expected, with the one word appearing in both positive headlines (*film*) having the highest $O_+$, while the only word to appear in a negative headline having $O_+ = 0$ and high $O_-$. These respective values reflect the estimated probability that a word is in a maximally positive (for $O_+$) or negative headline (for $O_-$). The values for $O$ are calculated as detailed in 2.2.2. Notably, the words appearing in headline 3 all have a value for $O_-$ despite appearing only in positive headlines. This is because the headline is not maximally positive or negative, and therefore every word has slight negative weighting.

Furthermore, words that appear in multiple headlines have their $O_\pm$ scores compounded. The words *like* and *football* appear in all three headlines once each. One would expect the two values of $O_\pm$ to be very close. However, these words lean more negatively. This is because the count of sentiment words in headline 1, which is a negative headline, is low. SESTM therefore assigns more weight to the words in that headline, yielding the negative reflection in $O$. The 'word' value also shows the successful preprocessing of each of the words to their respective stem. This shows the successful estimation of sentiment in a small sample.

After the implementation of the training section, I implemented the scoring segment, described in 2.2.3. I once again created a dummy headline that was very similar to headline 3 from the dummy inputs to test if it would give a similar estimate. The input is shown in 3.4, and the $\widehat{p}$ value for this headline is shown in 3.2. As expected, the predicted value is very similar to the $p$ value of headline 3, although pushed slightly towards the neutral. This squashing is the result of the penalty term, $\lambda$. At higher values of $\lambda$, this estimated $p$ is pushed closer to neutral.

```
headline_4 = {
    'date': '2022-02-15 16:23:17.923000+00:00',
    'ticker': 'ABDN',
    'mrkt_info': {
        'open': 250.3,
        'close': 258.5
    },
    'html': 'Alice finds films and pizza boring'
}
```

Listing 3.4: Example validation headline

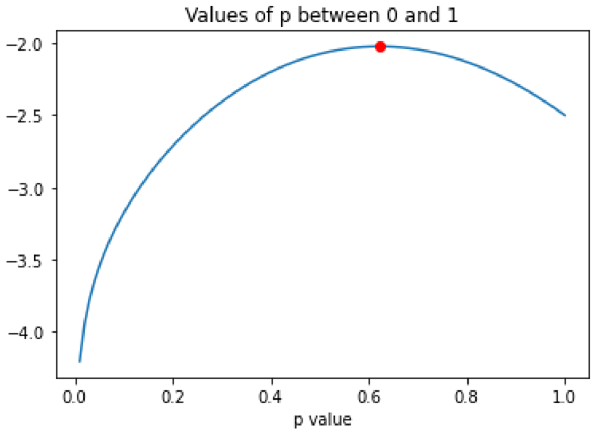


Figure 3.2: Estimated $p$ values, with maximum $p = 0.622$

### 3.3.1 Training the Model

Once the Kaggle sample has been pre-processed as detailed in 3.2, the model can be trained according to the algorithm outlined. 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 headlines 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 $\alpha$ must be larger than the current value, as the larger $\alpha$ is, the fewer words satisfy the condition. Similarly, the inverse for the case where the set of words contains fewer words than desired. $\kappa$ is selected to be the 86, 88, 90, 92 or 94th percentiles of word counts. Note that the $\kappa$ restraint is applied first such that a word is not selected via the $\alpha$ constraint that must then be removed due to the $\kappa$ constraint, leaving $S$ with fewer words than

| Window start date | $\|S\|/2$ | $\alpha_+$ | $\alpha_-$ | $\kappa$ | $\lambda$ | 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 3.2: Best configuration and error for each window. Smallest error window highlighted in **bold**

desired. Combining both the calculated $\alpha_\pm$ values and the calculated $\kappa$ value, the list of sentiment words for the training window is compiled, illustrated in listing 3.5. Finally, $\lambda$ 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 $\ell^1$ error is calculated for each, before selecting the setup with minimum error (as this is our loss function). $\ell^1$ 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 headline $i$ in the validation set. The loss function of $\ell^1$-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 A.

Table 3.2 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 $\ell^1$-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.

### 3.3.2 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.

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 single 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 headlines are divided into bigrams in the same way that the BOW representations are created. Stopwords are not

```
kappa_configs   = [86, 88, 90, 92, 94]
alpha_configs   = [25,50,100]
# fraction of positively tagged training headlines
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 3.5: Calculating list of sentiment words

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 article bodies, meaning there are far more bigrams than in that dataset than the one used here. For this reason, I chose to consider a higher percentage to encapsulate more bigrams to ensure correctness. After this filtering step, the rest of the training continues as before. The preparation for bigrams is shown in listing 3.6.

## 3.4 Out of Sample Testing

Using the optimally trained model (shown in table 3.2), the headlines 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 according to the $p$ 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

```
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 3.6: Calculating list of sentiment charged bigrams

| Window start date | $\|S\|/2$ | $\alpha_+$ | $\alpha_-$ | $\kappa$ | $\lambda$ | Minimum error | Avg Min Error |
|---|---|---|---|---|---|---|---|
| ***2010-1-1** | **100** | **0.0168** | **0.0218** | **90** | **5** | **20416.73** | **0.24739** |
| 2010-5-1 | 50 | 0.1123 | 0.0898 | 88 | 5 | 20716.47 | 0.24931 |
| 2010-9-1 | 25 | 0.1358 | 0.1013 | 92 | 5 | 20677.31 | 0.24977 |
| 2011-1-1 | 25 | 0.1175 | 0.0567 | 94 | 5 | 20238.53 | 0.24916 |
| 2011-5-1 | 100 | 0.0263 | 0.0243 | 92 | 5 | 19341.18 | 0.2494 |
| 2011-9-1 | 100 | 0.0697 | 0.0461 | 88 | 5 | 19354.6 | 0.2478 |
| 2012-1-1 | 100 | 0.0001 | 0.0 | 94 | 5 | 20568.7 | 0.24858 |
| 2012-5-1 | 100 | 0.0572 | 0.0579 | 88 | 10 | 21840.62 | 0.24962 |
| 2012-9-1 | 25 | 0.1472 | 0.1714 | 86 | 10 | 22286.96 | 0.24978 |
| 2013-1-1 | 25 | 0.1386 | 0.1638 | 88 | 5 | 21719.86 | 0.249 |
| 2013-5-1 | 25 | 0.1419 | 0.1619 | 88 | 5 | 22357.35 | 0.24843 |
| 2013-9-1 | 100 | 0.0206 | 0.0123 | 94 | 5 | 23398.95 | 0.24869 |
| 2014-1-1 | 100 | 0.0183 | 0.0142 | 94 | 5 | 24854.8 | 0.24853 |
| 2014-5-1 | 50 | 0.0999 | 0.1199 | 88 | 5 | 24305.78 | 0.24901 |
| 2014-9-1 | 100 | 0.0274 | 0.0281 | 94 | 5 | 23519.78 | 0.24886 |
| 2015-1-1 | 100 | 0.0238 | 0.0239 | 94 | 5 | 22768.1 | 0.24769 |
| 2015-5-1 | 100 | 0.021 | 0.0293 | 94 | 5 | 23613.92 | 0.24843 |
| 2015-9-1 | 100 | 0.0684 | 0.0618 | 90 | 5 | 26419.55 | 0.24827 |
| 2016-1-1 | 100 | 0.0823 | 0.0772 | 88 | 5 | 29914.03 | 0.24788 |

Table 3.3: Best configuration and error for each window using bigrams. Smallest error window highlighted in **bold**

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.

```
# list of unique tickers in list of
for t in outstanding_tickers:
    if not exists(t + '.json'):
        df,err = ek.get_data([t + '.N'], ['TR.CompanyMarketCap.Date', 'TR.TtlCmnSharesOut'], {'SDate': '2019-01-01', 'EDate': '2020-08-06', 'FRQ': 'D'})
        if (len(df.index) > 1):
            df.to_json(t + '.json', orient='records', lines=True)
        else:
            print("no outstanding stock information for " + t)
```

Listing 3.7: Collecting Outstanding Stock Information

I consider two methods of portfolio formation for each strategy: equal weighting (EW) and value weighting (VW). I decided to use these and ignore the price weighting strategy, as most of the information gained from testing this weighting method is captured by the combination of EW and VW. On each day of the out of sample time period, an arbitrary value is invested into each stock, taking either the long position for postively estimated stocks or the short position for negatively estimated stocks. This value is the same each day, although because the returns are calculated as a percentage, the raw value invested does not change the results. For the equally weighted portfolio, calculating the weights of each stock in a daily portfolio is simple and described in 2.3.1. For value weighting, more data is required, namely the outstanding stock information. I used Refinitiv Eikon Refinitiv (2022) to access this data, as I found it to be more reliable than the yfinance Python library for gathering outstanding stock data. For each unique stock in the list of headlines, the outstanding stocks are pulled and stored for later use, shown in figure 3.7. Eikon requires each stock ticker to have a suffix of the exchange, hence use of `t + '.N'`, which indicates the New York Stock Exchange. Importantly, the stocks purchased or sold on a given day by both strategies are almost identical, as some have no outstanding stock data and are therefore disregarded in the value weighted portfolio. The only difference between the two is the weighting.

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. This is to avoid the portfolio purchasing slightly negative stocks and selling slightly positive stocks. It also removes the possiblility of the portfolio selling and buying the same stock on the same day. On such occasions, fewer than 100 stocks will be traded on that day.

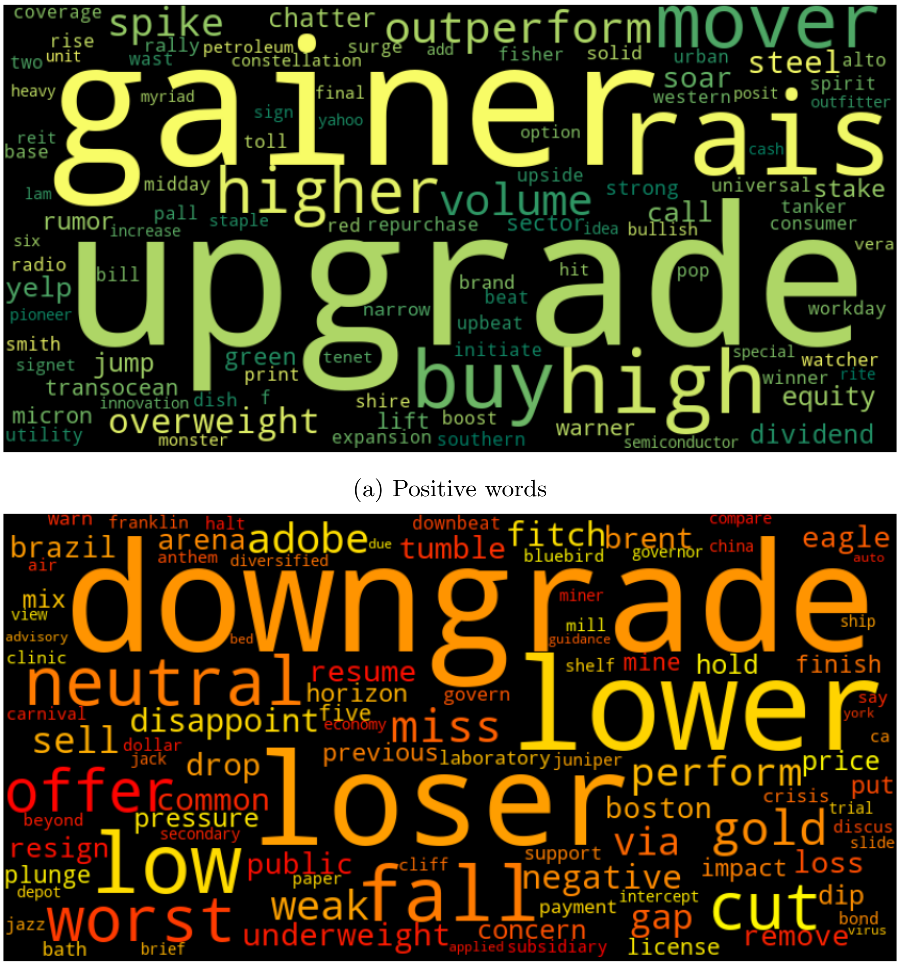
Generating portfolios from bigrams is done in a very similar manner. The bigrams are not used in solitude. Rather, the list of unigrams is augmented by the bigrams calculated by the model. Let the list of sentiment charged unigrams, and the associated matrix of probabilities be $S_u$ and $O_u$ respectively and the list of sentiment charged bigrams and respective matrix be $S_b$ and $O_b$. When considering the headline as a whole, the list of sentiment words is $S = (S_u \cup S_b)$. Each headline is decomposed into their respective bag of words representations in a similar combinatory manner. Let the unigram BOW representation of a headline be $d_u$ and likewise for bigram be $d_b$. The final BOW representation in this case would be $d = (d_u \cup d_b)$. The two sets are entirely distinct and will not interfere with one another, as it is guaranteed that no sentiment words from $S_u$ will also be in $S_b$, and consequently no bigram $j_b \in S_b$ will exist in $O_u$. This means that bigrams will have no associated value with respect to unigrams, leaving $\widehat{p}_i$ untouched. The same is true of the set of unigrams with respect to bigrams. Thus, the resultant value is an accurate reflection of the sentiment of both unigrams and bigrams.

For comparison purposes, I also examine the out of sample headlines with respect to two baseline dictionaries: the Loughran McDohanald lexicon and the Harvard IV lexicon. The portfolio construction strategy is similar to that of SESTM, where the headlines on day $t$ are examined, and the calculated sentiment values are used to form 50 positive and 50 negative stocks.

# 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 those 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 subconscious 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, in any learning based algorithm, overfitting must be considered. Overfitting in this instance refers to a model that is too closely trained to the training sample, meaning incorrect alignment with out of sample data. Included in the sample are headlines from 'Benzinga', which is a company that offers realtime news articles (Benzinga, 2022), 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). To account for this, the addition of bigrams (discussed further in 4.1.1) gives an insight into frequently occurring pairs of words, and helps to isolate context specific sentiment. If a word has a high estimated impact in a bigram, then the majority of the sentiment comes from its co-appearance with its partner word, helping to eliminate such issues.

Some words, such as 'rais' are cut off to their stem by the stemming and lemmatisation step (i.e. 'rais' from 'raise'), as the stem is also an English word. However, such words are likely to be very infrequently used words themselves, and are very unlikely to be used in a headline. These edge cases do not affect the overall predictive quality of the model, as evidenced by the words shown in figure 4.1. The sentiment for each word is still captured in the stem, and each article is processed in the same way, thus there is no sentiment or predictive power lost in this way. There is a slight complication when it comes to comparing the words included in the model to those in the H4 and LM lexicons, as these lexicons do not contain the stemmed words. For this reason, when checking if a word in the trained model exists in either dictionary, each word in either dictionary is preprocessed in the same way discussed in section 3.2

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 significantly.

### 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. Many of the frequently words unigrams also appear in the bigram.

This is partly due to the mutual information restriction on the bigrams consiered by the model, but also because still carry sentiment in combination with other words. Furthermore, somewhat surprisingly there are bigrams that appear in every training window. The headlines in this dataset have an average wordcount of 6.11 (excluding stopwords and non-English words), and therefore one may expect that there would be very few repeating bigrams, which is not the case. This signifies that headline writers may subconsciously use certain combinations of words to convey either positive or negative sentiment

The following bigrams appear positively in at least 13 of the 19 training windows, with those highlighted in **bold** appearing in all headlines:

*micron technology, repurchase program, spike higher,* **volume mover**, **week high**

The following bigrams appear negatively in at least 14 of the 19 training windows, with those highlighted in **bold** appearing in all headlines:

*general dynamic, bath beyond, bed bath, sector perform, secondary offer, week low, resume trade, offer common,* **public offer**

Several of the high frequency unigrams do not appear in the high frequency bigrams, indicating that the sentiment carried by these words is not tied to context, for example raise for positive and weak for negative. Due to the nature of the headlines, the model also picks up on especially highly or poorly performing companies, and this is sometimes reflected in the generated word lists. For example, '*bed bath*' and '*bath beyond*' both appearing in the negative bigram list is almost certainly a result of the poor performance of the company *Bed, Bath and Beyond*, whose stock prices stagnated and then fell from mid 2013 (yahoo, 2022). Also seen in figure 4.1, Adobe is likely to be in reference to the company 'Adobe'.

## 4.2 Daily returns

| | Sharpe | Average | Daily | FF3 | | FF5 | |
|---|---|---|---|---|---|---|---|
| Formation | Ratio | Return | Turnover | $\alpha$ | $R^2$ | $\alpha$ | $R^2$ |
| EW LS | 0.84 | 9.23 | | 5.59 | 4.11% | 5.57 | 4.16% |
| EW L | 0.69 | 9.75 | | 6.56 | 21.62% | 5.86 | 23.0% |
| EW S | -0.09 | -0.53 | | -1.66 | 25.71% | -0.98 | 26.68% |
| VW LS | 0.56 | 6.31 | | 4.73 | 0.83% | 7.16 | 4.58% |
| VW L | 0.86 | 10.47 | | 6.51 | 34.3% | 7.55 | 35.16% |
| VW S | -0.38 | -4.16 | | -2.48 | 32.54% | -1.08 | 34.08% |

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

Using the headlines that were saved for out of sample testing, a portfolio is constructed for each day. On average, 358 firms have articles linked to them on a given day, and of these, almost three quarters of these firms have headlines containing 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 trades with 49.5 stocks, while the short side has 48, so the model trades at very close to 100 stocks the majority of the time.

Table 4.1 describes the performance of the constructed portfolios. The long-short portfolio is a zero-net-investment portfolio, which is a theoretical portfolio that requires zero investment, as it buys the same value of stocks as it sells (Chen, 2022). The portfolio is theoretical as a true zero-cost investment is impossible in practice for a number of reasons, one of which being that to sell a stock and profit from the decline, there is collateral from the loan from the broker. However, for the purposes of the following experiments, it is sufficient. 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.

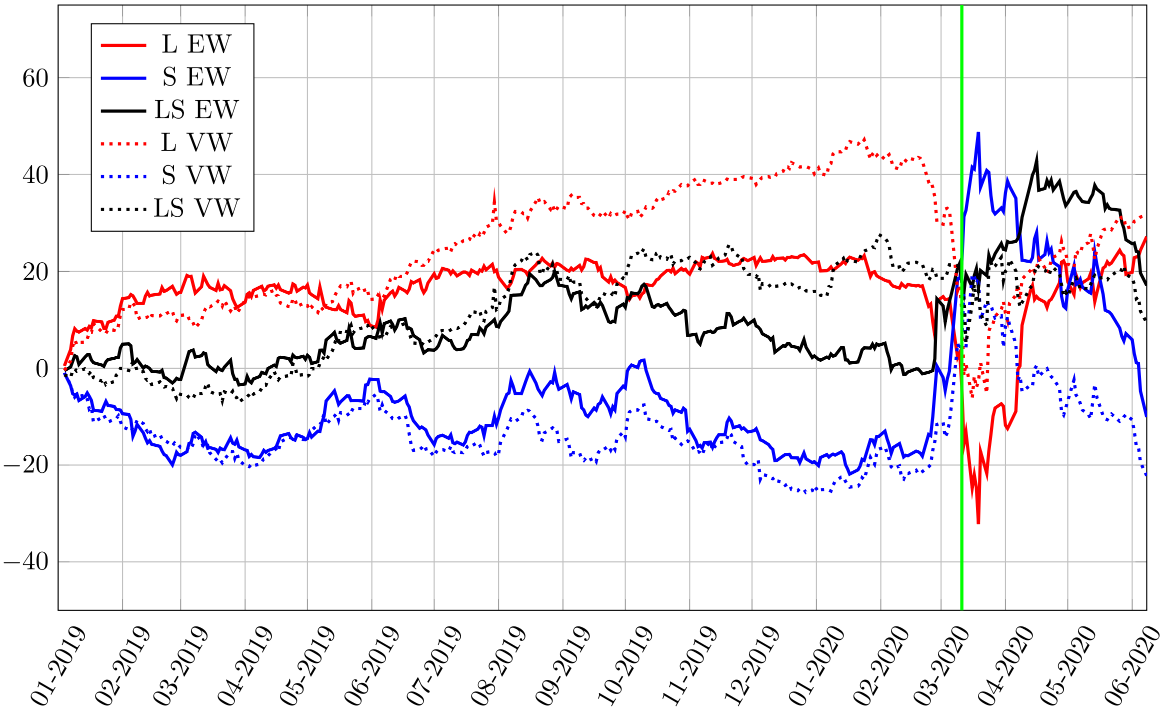


Figure 4.2: Cumulative log returns for each formation over the out of sample headlines

The Sharpe ratio refers to the calculated annualised sharpe ratio and this reflects the ratio of profit versus risk. Sharpe ratios are annualised by multiplying the daily sharpe ratio (shown in 2.3.2) by $\sqrt{252}$ as this is the number of trading days in a year, and will give a clearer image of the risk involved in the potential profit.

The FF3 and FF5 sections refer to Fama French 3 and 5 factor regression respectively, while the $\alpha$ concerns the intercept. The $\alpha$ value reflects how much the output outperformed the expected value calculated by the factors in the model and is generated by privately created information, or information that is not available in the market. In other words, if the $\alpha$ 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. The $R^2$ value is the measure of the proportion of variance that is for the dependent variable (in this case, daily returns for the portfolio) is explained by the independent variables (in this case, the Fama French factors discussed in 2.3.2). Therefore, if $R^2 = 25\%$, then around a quarter of the observed variation can be explained by the inputs. In terms of an investment, this corresponds to the percentage of movement that can be explained by movements in the independent variables.

This figure clarifies two key facts: firstly, most of the formations generate a slight profit with a fair amount of risk, reflected in the Sharpe ratios. None of the formations have a Sharpe ratio of $> 1$, meaning each of the portfolios carry significant risk. The second fact is that the equal weighted portfolio outperforms the value weighted portfolio overall, while the value weighted portfolio sees a slight performance on the long side.

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. Due to lockdown restrictions imposed wordwode, the stock market experienced one of the worst crashes, and was felt particularly by large stocks, before recovering a short while later. The markets began feeling the effect of the lockdown in February 2020, and the crash peaked in March 2020, before beginning to recover in late April (Ganie, Wani, and Yadav, 2022). This crash can be clearly seen in the affected profits, as the long side suffers significantly, while the short side succeeds greatly. This is simply due to the vasy majority of stocks falling in value, therefore shorting almost any stocks during this time would result in a profit. The extreme disparity in profit for the long side 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.

Graph 4.2 reveals slightly different information than that in table 4.1. Here, the long side of the value weighted graph is shown to outperform the equal weighted counterpart considerably, before taking a substantial fall during the pandemic induced market crash. On the other hand, the short sides of both weighting strategies fare similarly, until after the crash, where the equal weighted short side experiences a higher spike in profit. Clearly, SESTM is capable of predicting a positive shift in stocks on a one-day-ahead principle, shown by the positive returns in the graph. It is more successful when predicting larger stocks, as value weighed portfolios are influenced more by larger stocks. This represents SESTM's success in detecting positive sentiment in headlines concerning these large firms, which is likely due to increased coverage. As previously discussed, the primary goal for a headline is to capture as much information about the main article body as possible while the secondary goal is to attract the reader to click on the article. For this reason, the headline is far more likely to contain information about more well known stocks, as this satisfies the secondary goal of garnering clicks: if the headline's content is about a stock that is known to the reader, it is more likely that the rest of the article is read.

The two short formations perform very similarly, with both making slight losses. This contradicts the claims made concerning headlines containing more information about large stocks somewhat, although I believe this is due to a different matter. Gligorić et al. (2021) explore the success of a headline based on linguistic features, and conclude that the presence of positive-emotion words is negatively associated with the success of a headline (in terms of popularity), while the inverse is true for negative-emotion words. For this reason, if there is ambiguity on the potential future of a stock, a headline is more likely to take the negative approach, as it is more likely to be successful. In other words, if a news article is mostly neutral, the headline is more likely to contain negative-emotion words, as this garners more views. As a result, the negatively tagged words are less reliable than their positive tagged counterparts, which be one possbile explanation for the lack of profit made by either short formation.

| | Sharpe | Average | Daily | FF3 | | FF5 | |
|---|---|---|---|---|---|---|---|
| Formation | Ratio | Return | Turnover | $\alpha$ | $R^2$ | $\alpha$ | $R^2$ |
| EW LS | 0.84 | 9.23 | | 5.59 | 4.11% | 5.57 | 4.16% |
| EW L | 0.69 | 9.75 | | 6.56 | 21.62% | 5.86 | 23.0% |
| EW S | -0.09 | -0.53 | | -1.66 | 25.71% | -0.98 | 26.68% |
| VW LS | 0.56 | 6.31 | | 4.73 | 0.83% | 7.16 | 4.58% |
| VW L | 0.86 | 10.47 | | 6.51 | 34.3% | 7.55 | 35.16% |
| VW S | -0.38 | -4.16 | | -2.48 | 32.54% | -1.08 | 34.08% |
| Excluding market crash | | | | | | | |
| EW LS | 0.66 | 6.51 | | 5.52 | 2.24% | 4.16 | 4.51% |
| EW L | 0.64 | 5.46 | | 1.95 | 9.67% | 0.58 | 12.97% |
| EW S | 0.03 | 1.05 | | 2.76 | 16.16% | 2.77 | 16.96% |
| VW LS | 0.99 | 7.61 | | 6.46 | 0.81% | 6.67 | 0.9% |
| VW L | 1.52 | 10.98 | | 7.95 | 14.51% | 7.84 | 14.59% |
| VW S | -0.59 | -3.37 | | -2.3 | 17.95% | -1.98 | 18.21% |

Table 4.2: Performance of daily news sentiment portfolios one day ahead, ignoring headlines dated after January 2020

For closer inspection of the model's predictive ability, table 4.2 shows the same metrics as table 4.1, only without any of the out of sample articles dated after February 28th 2020 to observe the impact without the greatly added noise of a stock market crash. Here, the trend of the data is very similar, with the value weighted long formation still outperforming its equal weighting counterpart, and vice versa with the short side, meaning that it remains steadfast that headlines indicating negative sentiment about smaller stocks contain more reliable information than that with positive stocks. The most notable increase in Sharpe ratio is the value weighted long side, despite only a marginal increase in average return. This is likely due to the large plunge in prices shortly after the global pandemic announcement, which increases the standard deviation of the portfolio significantly, thus increasing perceived risk. Other than this formation, it is clear to see that the model struggles to see any profit devoid of serious risk, indicated by all Sharpe ratios being $< 1$. Headlines being so short in terms of word count, as well as the small out of sample dataset size, lead to difficulties in gaining sufficient information to consistently accurately predict stock movement based on headlines alone. The conclusions drawn from headlines leave large room for error and while there is still potential for information gain, the signal is not as clear as longer text forms.

| | Baseline | Baseline | Sharpe | Average | Daily | FF3 | | FF5 | |
|---|---|---|---|---|---|---|---|---|---|
| Formation | Sharpe | Returns | Ratio | Return | Turnover | $\alpha$ | $R^2$ | $\alpha$ | $R^2$ |
| EW LS | 0.84 | 9.23 | 1.75 | 20.69 | | 18.19 | 8.87% | 17.71 | 8.98% |
| EW L | 0.69 | 9.75 | 1.29 | 17.84 | | 14.55 | 18.81% | 14.02 | 19.93% |
| EW S | -0.09 | -0.53 | 0.15 | 2.85 | | 2.95 | 34.48% | 3.0 | 35.2% |
| VW LS | 0.56 | 6.31 | 1.62 | 15.45 | | 12.45 | 5.12% | 13.41 | 6.31% |
| VW L | 0.86 | 10.47 | 1.01 | 11.32 | | 8.21 | 32.42% | 9.13 | 33.53% |
| VW S | -0.38 | -4.16 | 0.27 | 4.13 | | 3.55 | 30.11% | 3.58 | 31.48% |
| Excluding market crash | | | | | | | | | |
| EW LS | 0.66 | 6.51 | 0.85 | 8.26 | | 7.15 | 2.49% | 6.09 | 3.77% |
| EW L | 0.64 | 5.46 | 1.12 | 9.19 | | 5.86 | 9.2% | 4.79 | 11.25% |
| EW S | 0.03 | 1.05 | -0.21 | -0.93 | | 0.47 | 16.02% | 0.49 | 16.59% |
| VW LS | 0.99 | 7.61 | 1.53 | 11.16 | | 10.54 | 1.43% | 10.78 | 1.54% |
| VW L | 1.52 | 10.98 | 1.35 | 9.93 | | 7.07 | 17.33% | 7.15 | 17.46% |
| VW S | -0.59 | -3.37 | 0.06 | 1.23 | | 2.66 | 21.48% | 2.81 | 21.59% |

Table 4.3: Performance of daily news sentiment portfolios using bigrams alongside unigrams. Baseline values are those achieved by unigrams alone

Table 4.3 shows the performance of the portfolios when augmenting the sentiment word list with bigrams. Most notable in this table is the vast improvement in both sharpe ratios and the average daily returns for the equal weighted strategy, particularly in the long leg of the portfolios. The sharpe ratio of 1.75 produced by the equal weighted portfolio shows that the profit generated has a very favourable risk to profit ratio. This large shift in performance indicates that sentiment in headlines as a whole is often conveyed using multiple words, and that context has importance in this text type. Ignoring articles from after the market crash, the performance of all formations is either comparable or better, reinforcing this notion. As before, a very high proportion of returns are alhpa, meaning private information not available to the market is present in this model.

## 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.

Table 4.4 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 shows the same information as 4.1 as a reference.

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 based on the sentiment 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 15.48 and 10.8 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. This method sees the largest profit of any formation, indicating that the freshness of news is significant when observing sentiment. In

| | Sharpe | Average | Daily | FF3 | | FF5 | |
|---|---|---|---|---|---|---|---|
| Formation | Ratio | Return | Turnover | $\alpha$ | $R^2$ | $\alpha$ | $R^2$ |
| | | | Day $t-1$ | | | | |
| EW LS | 15.48 | 210.11 | | 207.8 | 1.09% | 207.19 | 4.93% |
| EW L | 7.83 | 122.42 | | 120.91 | 21.43% | 120.44 | 23.06% |
| EW S | 5.87 | 87.7 | | 86.2 | 28.81% | 86.05 | 29.01% |
| VW LS | 9.27 | 110.04 | | 107.9 | 1.22% | 108.69 | 3.74% |
| VW L | 4.18 | 55.35 | | 52.18 | 34.77% | 52.99 | 36.55% |
| VW S | 4.55 | 54.69 | | 55.02 | 31.73% | 54.99 | 31.75% |
| | | | Day $t+0$ | | | | |
| EW LS | 15.67 | 234.84 | | 233.93 | 2.46% | 233.89 | 3.05% |
| EW L | 7.57 | 134.7 | | 134.21 | 10.26% | 134.64 | 11.18% |
| EW S | 6.21 | 100.15 | | 99.03 | 21.85% | 98.56 | 22.02% |
| VW LS | 12.51 | 139.79 | | 139.08 | 1.39% | 139.21 | 1.62% |
| VW L | 3.77 | 53.24 | | 51.09 | 15.87% | 51.56 | 17.01% |
| VW S | 6.87 | 86.55 | | 87.3 | 24.3% | 86.96 | 24.89% |
| | | | Day $t+1$ | | | | |
| EW LS | 0.84 | 9.23 | | 5.59 | 4.11% | 5.57 | 4.16% |
| EW L | 0.69 | 9.75 | | 6.56 | 21.62% | 5.86 | 23.0% |
| EW S | -0.09 | -0.53 | | -1.66 | 25.71% | -0.98 | 26.68% |
| VW LS | 0.56 | 6.31 | | 4.73 | 0.83% | 7.16 | 4.58% |
| VW L | 0.86 | 10.47 | | 6.51 | 34.3% | 7.55 | 35.16% |
| VW S | -0.38 | -4.16 | | -2.48 | 32.54% | -1.08 | 34.08% |

Table 4.4: Performance of Daily News Sentiment Portfolios day $t-1$ to day $t+1$

other words, the closer to the release of a headline that the sentiment can be utilised, the greater the potential profits.

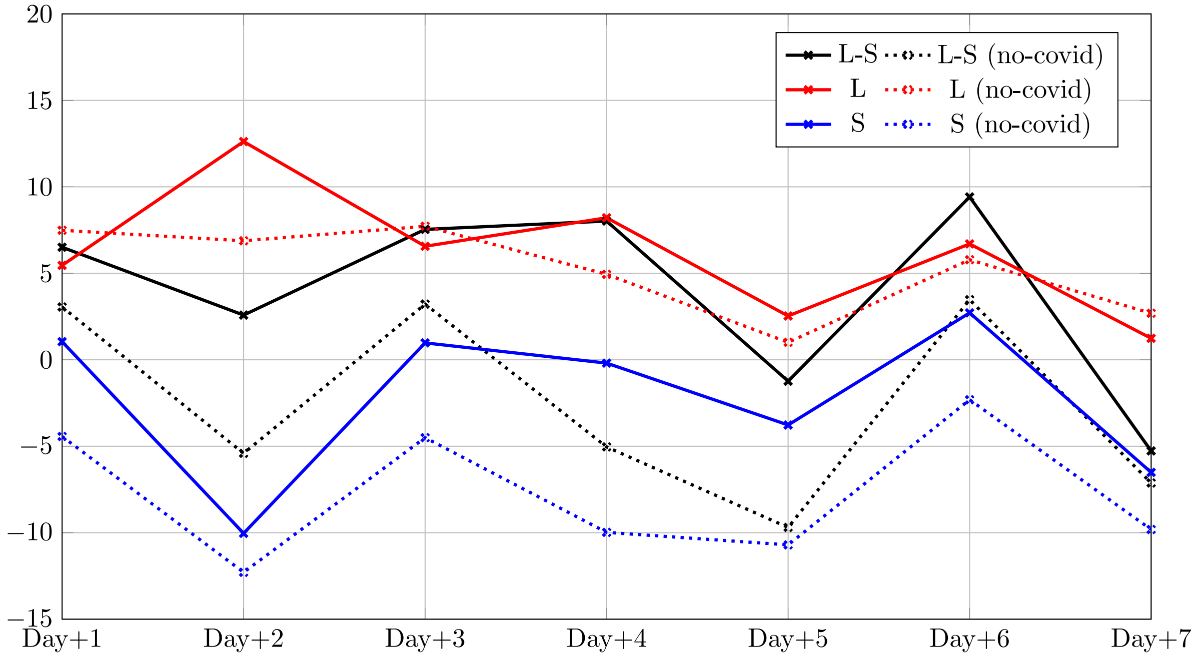


Figure 4.3: Average daily returns for different waiting periods. If article is released on day $t$ (between 9 a.m. of day $t$ and 9 a.m. of day $t+1$), portfolio is created on day $t+n$ and sold on day $t+(n+1)$ where $n$ is the value indicated by the x axis

Figure 4.3 shows the time taken for a headline to be fully assimilated into the stock prices. The graph indicates that SESTM picks up on sentiment that takes a few days to be fully incorporated into the

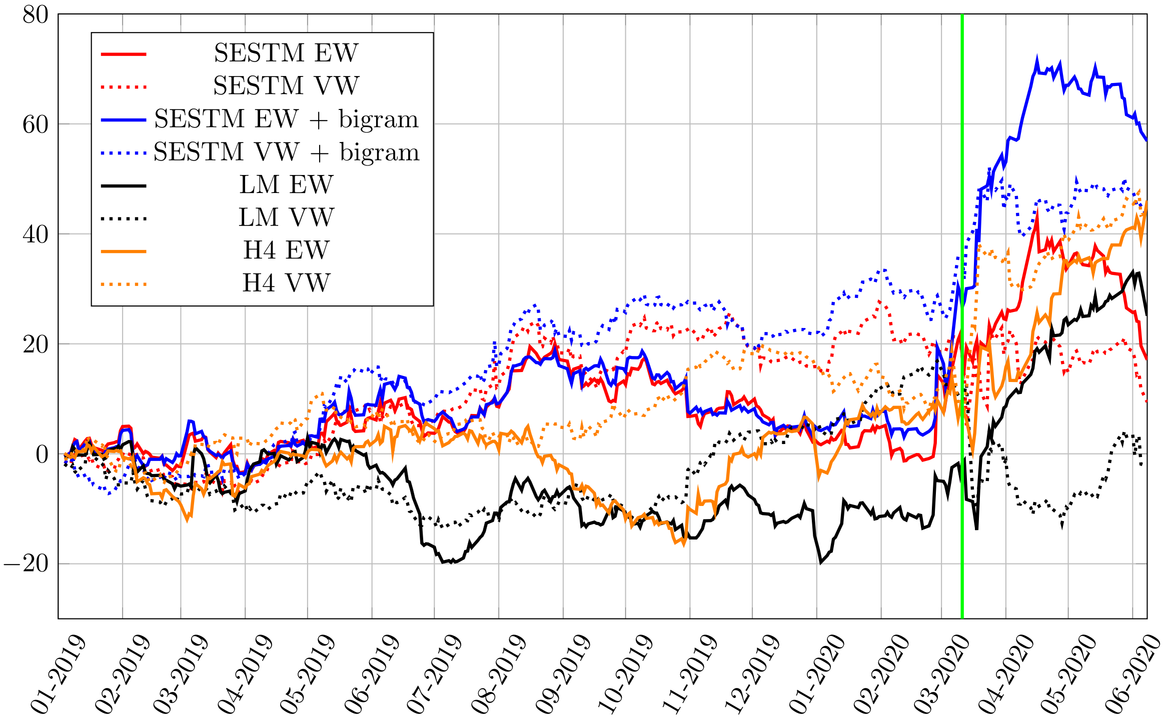


Figure 4.4: Cumulative log returns for each formation over the out of sample headlines

prices, as it takes until day +5 for the LS portfolio to reach zero. The short side is incorporated much more quickly, only taking around 2 days to be fully assimilated, while the long side takes around 5 days for full assimilation. The information shown by this graph is important as it can potentially show that a portfolio is able to pick up on complex sentiment that takes a number of days to fully assimilate into the prices.

## 4.4 Comparison to other methods

In this section, we compare the ability of SESTM against similar strategies. Similar to the methods used to construct portfolios for SESTM, we buy the top 50 positive stocks and short the top 50 negative stocks each day. Again, if a firm has multiple headlines, the average sentiment of all articles on that day is taken.

Table 4.5 compares the average returns of the three different methods. Here, SESTM is shown to outperform LM and H4 overall (considering both long and short sides) when the market crash is excluded, with H4 slightly outperforming LM. The LM dictionary was introduced as a solution to H4 in a financial context, so the outperformance is surprising, although not wholly unexplained. As described by Reah (2002), the language used in headlines is often unlike any other language, so a dictionary intended for use on 10-K filings is not the best fit for languages in headlines. The trend of all three techniques is similar: positive sentiment is easier to profit from that the negative counterpart. This is evidenced by the Sharpe ratios of 0.69, 1.03, and 1.36 respectively.

Figure 4.4 details the cumulative log returns for each formation, up to the date of the COVID-19 pandemic announcement. Once again, the market crash affected all of the formations significantly, and can be seen by the sudden spike on all four portfolio formations. Most notable in this graph is that SESTM performs consistently higher than either lexicon, despite having slightly lower average returns. This is due to the high volatility of stock exchanges during the market crash. Both LM and H4 fared considerably better than SESTM during this time, both more than doubling their respective long legs while also having slightly more profitable short legs. On the other hand, SESTM has a more stable short leg when this crash is not considered, meaning that it more consistently outperforms over the course of the entire sample. This could be due to the lexicons being better equipped to deal with such events, and therefore suffering slightly less.

| | Sharpe | Average | Daily | FF3 | | FF5 | |
|---|---|---|---|---|---|---|---|
| Formation | Ratio | Return | Turnover | $\alpha$ | $R^2$ | $\alpha$ | $R^2$ |
| | | | SESTM | | | | |
| LS | 0.84 | 9.23 | | 5.59 | 4.11% | 5.57 | 4.16% |
| LS (no-covid) | 0.66 | 6.51 | | 5.52 | 2.24% | 4.16 | 4.51% |
| L | 0.69 | 9.75 | | 6.56 | 21.62% | 5.86 | 23.0% |
| L (no-covid) | 0.64 | 5.46 | | 1.95 | 9.67% | 0.58 | 12.97% |
| S | -0.09 | -0.53 | | -1.66 | 25.71% | -0.98 | 26.68% |
| S (no-covid) | 0.03 | 1.05 | | 2.76 | 16.16% | 2.77 | 16.9% |
| | | | LM | | | | |
| LM LS | 1.42 | 13.04 | | 9.59 | 4.58% | 9.63 | 5.32% |
| LM LS (no-covid) | -0.03 | 0.63 | | -1.58 | 2.58% | -1.38 | 2.77% |
| LM L | 1.03 | 14.07 | | 11.72 | 22.42% | 10.99 | 22.66% |
| LM L (no-covid) | 0.44 | 4.17 | | 1.28 | 27.68% | 1.48 | 27.78% |
| LM S | -0.13 | -1.02 | | -2.83 | 20.29% | -2.06 | 21.11% |
| LM S (no-covid) | -0.56 | -3.53 | | -3.68 | 35.68% | -3.67 | 35.69% |
| | | | H4 | | | | |
| H4 LS | 1.44 | 14.26 | | 13.8 | 4.53% | 14.34 | 4.75% |
| H4 LS (no-covid) | 0.07 | 1.28 | | 0.39 | 3.42% | 0.76 | 3.85% |
| H4 L | 1.36 | 18.47 | | 17.67 | 21.17% | 17.79 | 21.24% |
| H4 L (no-covid) | 1.11 | 8.51 | | 5.97 | 21.6% | 6.35 | 21.98% |
| H4 S | -0.38 | -4.21 | | -4.56 | 22.78% | -4.15 | 22.98% |
| H4 S (no-covid) | -0.99 | -7.23 | | -6.4 | 30.61% | -6.4 | 30.65% |

Table 4.5: Performance of daily portfolios for different algorithms. The formations labelled (no-covid) indicate the data excludes days after March 1 2020, to account for the COVID-19 market crash. We only consider the equal weighted strategies here

| | Sharpe | Average | Daily | FF5 + SESTM | | FF5 + bigram | | FF5 + LM | | FF5 + H4 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Ratio | Return | Turnover | $\alpha$ | $R^2$ | $\alpha$ | $R^2$ | $\alpha$ | $R^2$ | $\alpha$ | $R^2$ |
| | | | | | SESTM | | | | | | |
| EW | 0.84 | 9.23 | | | | -8.21 | 74.08% | 3.29 | 6.56% | 4.61 | 4.68% |
| VW | 0.56 | 6.32 | | | | -1.58 | 37.35% | 7.22 | 4.78% | 4.83 | 6.37% |
| | | | | | SESTM w/ bigrams | | | | | | |
| EW | 1.75 | 20.69 | | 12.49 | 75.28% | | | 14.81 | 12.06% | 16.7 | 9.44% |
| VW | 1.62 | 15.45 | | 9.63 | 38.49% | | | 13.63 | 10.33% | 12.56 | 6.56% |
| | | | | | LM | | | | | | |
| EW | 1.42 | 13.04 | | 11.15 | 13.92% | 9.42 | 14.7% | | | 9.08 | 17.63% |
| VW | 0.18 | 2.46 | | -1.48 | 10.96% | -4.15 | 14.61% | | | -0.45 | 10.94% |
| | | | | | H4 | | | | | | |
| EW | 1.44 | 14.26 | | 11.91 | 3.11% | 11.23 | 3.07% | 8.8 | 9.11% | | |
| VW | 1.57 | 15.13 | | 14.92 | 3.28% | 14.44 | 1.73% | 15.79 | 1.62% | | |

Table 4.6: Table showing regression of each technique on each other

Table 4.6 highlights the novel information revealed from each individual lexicon. The regression method is the same as calculating the FF5 factors, only the returns from each configuration are included, providing a matrix of results. Here, each technique can be seen to have its own unique information, evidenced by the alpha value for each regression being a significant portion of the average returns. The only exception of this being the regression of SESTM on SESTM with bigrams. The high $R^2$ value and low alpha is expected, as the two strategies have an almost identical sentiment word list and as a result share a lot of the information. Importantly for SESTM, when it is regressed on other lexicons, it has high alpha and low $R^2$, meaning that it is generating useful information that is not captured by either lexicon. This table suggests that using multiple strategies in combination could yield greater profit, as combining the private information into a single portfolio would benefit from the private information of all strategies.

# Chapter 5

# Conclusion

## 5.1 Contributions and Achievements

This project presents an implementation for the algorithm described by Ke, Kelly, and Xiu (2019) that is able to take labelled financial headlines and construct a model of words that can then be used to predict stock movement the day after a headline release.

To complete the model, I have written more than 3000 lines of code. The three main components of this project are the pre-processing, the model training (SESTM implementation) and the evaluation (generation and evaluation of portfolios formed). The choices I made regarding the pre-processing of the headlines are presented in 3.2. The full implementation of the SESTM algorithm is detailed in 3.3.1, while the evaluative methods of the constructed portolios are discussed in 3.4.

The level of the model's predictive quality has been evaluated according to volatility, as well as overall returns in 4.2, using simulated portfolios on out of sample headlines. The theoretical returns are calculated using real market data from *yfinance* (Aroussi, 2022), and as such reflect the real world capabilities of the model.

A comparison between the potential profits of the portfolio generated by this model and other available techniques is available in section 4.4. The comparison highlights the success of the algorithm in predicting positive sentiment, while also drawing attention to the shortcomings in gleaning negative sentiment.

## 5.2 Current Project Status

In the current state, a fully functional SESTM implementation is in place, with a number of evaluative steps taken as a measure of success. Of the original outlined aims, I have completed the majority of them. The first aim, to implement and optimise SESTM has been achieved fully as evidenced in Section 3.3. I spent considerable time optimising and tuning my implementation, as the runtime can become very long.

The second aim was to successfully train a model using the dataset of financial headlines, using stock returns as a teaching signal. This has also been fully completed, and can be seen in Section 3.3.1. The various models trained over the course of the project can be found at the following url: `https://github.com/fxlmo/fxlmo-masters/tree/main/kaggle/data/models`.

The third aim was to evaluate the success of the model by evaluating portfolios. This has been achieved somewhat, although the out of sample data contained data that was irregular, meaning it was less reliable than originally thought. The results can be seen in Section 4.2.

The fourth and final aim was to compare the performance of SESTM against other sentiment analysis techniques. This has been completed, and shown in 4.4. The performance is shown to be comparable to other methods, and even slightly outperforms by some metrics.

## 5.3 Future Work

While the project can be considered complete, it would benefit from further training with a larger dataset. As headlines are very short pieces of text, the required dataset for good accuracy would be very large, and so training and testing this model with another, larger dataset would be beneficial to further solidify the results. Additionally, further analysis on such a dataset would

## 5.4 Conclusion

Ultimately, SESTM is able to effectively estimate the positive sentiment of a financial headline, and use this to correctly predict the positive movement of a stock on the following day, particularly when trained with bigrams. Unfortunately, the performance is not the same when considering the negative sentiment of a headline, and predicting downward movement in a stock. This is potentially due to the small size of the dataset, and so would benefit from further investigation with more data points.

It is shown that the algorithm performs comparatively with other methods, and generates private information not shared by such methods. It is, however, more flexible than other methods, as it is easily able to be reconfigured when new data appears, and as a result is able to remain relevant, as the model can simply be retrained.

# Bibliography

Aroussi, Ran (2022). *yfinance*. URL: https://pypi.org/project/yfinance/.

Benzinga (2022). URL: https://www.benzinga.com/.

Chen, James (2022). *Zero-Investment Portfolio*. URL: https://www.investopedia.com/terms/z/zero-investment-portfolio.asp.

Fama, Eugene F and Kenneth R French (1992). "The cross-section of expected stock returns". In: *the Journal of Finance* 47.2, pp. 427–465.

— (2015). "A five-factor asset pricing model". In: *Journal of financial economics* 116.1, pp. 1–22.

Felmeden, Joshua (2022). *GitHub*. URL: https://github.com/fxlmo/fxlmo-masters.

French, Kenneth (2022). *Kenneth R. French - Data Library*. URL: https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html.

Ganie, Irfan Rashid, Tahir Ahmad Wani, and Miklesh Prasad Yadav (2022). "Impact of COVID-19 Outbreak on the Stock Market: An Evidence from Select Economies". In: *Business Perspectives and Research*, p. 22785337211073635.

Gligorić, Kristina et al. (2021). "Linguistic effects on news headline success: Evidence from thousands of online field experiments (Registered Report Protocol)". In: *Plos one* 16.9, e0257091.

Haddi, Emma, Xiaohui Liu, and Yong Shi (2013). "The role of text pre-processing in sentiment analysis". In: *Procedia computer science* 17, pp. 26–32.

Hofmann, Thomas (2013). "Probabilistic latent semantic analysis". In: *arXiv preprint arXiv:1301.6705*.

Jegadeesh, Narasimhan and Di Wu (2013). "Word power: A new approach for content analysis". In: *Journal of financial economics* 110.3, pp. 712–729.

John, Vineet and Olga Vechtomova (2017). "Sentiment analysis on financial news headlines using training dataset augmentation". In: *arXiv preprint arXiv:1707.09448*.

Ke, Zheng Tracy, Bryan T Kelly, and Dacheng Xiu (2019). *Predicting returns with text data*. Tech. rep. National Bureau of Economic Research.

Kirange, DK, Ratnadeep R Deshmukh, et al. (2016). "Sentiment Analysis of news headlines for stock price prediction". In: *Composoft, An International Journal of Advanced Computer Technology* 5.3, pp. 2080–2084.

Loughran, Tim and Bill McDonald (2011). "When Is a Liability Not a Liability? Textual Analysis, Dictionaries, and 10-Ks". In: *The Journal of Finance* 66.1, pp. 35–65. DOI: https://doi.org/10.1111/j.1540-6261.2010.01625.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1540-6261.2010.01625.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.2010.01625.x.

Madhoushi, Zohreh, Abdul Razak Hamdan, and Suhaila Zainudin (2015). "Sentiment analysis techniques in recent works". In: *2015 science and information conference (SAI)*. IEEE, pp. 288–291.

Manning, Christopher, Prabhakar Raghavan, and Hinrich Schütze (2010). "Introduction to information retrieval". In: *Natural Language Engineering* 16.1, pp. 100–103.

Nemes, László and Attila Kiss (2021). "Prediction of stock values changes using sentiment analysis of stock news headlines". In: *Journal of Information and Telecommunication* 5.3, pp. 375–394.

Python (2022). URL: https://wiki.python.org/moin/PythonSpeed/PerformanceTips#Loops.

Reah, Danuta (2002). *The language of newspapers*. Psychology Press.

Refinitiv (2022). URL: https://www.refinitiv.com/en/products/eikon-trading-software#overview.

Sharpe, William F (1966). "Mutual fund performance". In: *The Journal of business* 39.1, pp. 119–138.

Taboada, Maite et al. (2011). "Lexicon-based methods for sentiment analysis". In: *Computational linguistics* 37.2, pp. 267–307.

Tetlock, Paul C (2007). “Giving content to investor sentiment: The role of media in the stock market”. In: *The Journal of finance* 62.3, pp. 1139–1168.

Tetlock, Paul C, Maytal Saar-Tsechansky, and Sofus Macskassy (2008). “More than words: Quantifying language to measure firms’ fundamentals”. In: *The journal of finance* 63.3, pp. 1437–1467.

Teweles, Richard J and Edward S Bradley (1998). *The stock market*. Vol. 64. John Wiley & Sons.

TradingHours (2022). *Is the stock market open?* URL: https://www.tradinghours.com/open?.

yahoo (2022). URL: https://finance.yahoo.com/quote/BBBY?p=BBBY&.tsrc=fin-srch.

# Appendix A

# List of Optimisations

As part of the project I intended to optimise my implementation of the algorithm. This section of the appendix details the optimisations I made, along with justifications and overall speedup.The training and validation window is a simple three deep nested loop with $O(n^3)$, so keeping the inside of the loop as small as possible is paramount. Small increments in runtime here are compounded significantly. The original algorithm is shown in Listing A.2. In this listing, the first optimisation can be seen. Python allows the use of *list comprehension*, which is a shorthand method of creating list while using a loop. The basic idea behind this is shown in Listing A.1. As shown in the Python wiki Python, 2022, there is a slight speedup when using list comprehension, because in this example, it does not have to look up the list and the append method every iteration. When using many datapoints and inside a triple nested loop, this slight time gain is significant.

```
#round each value in oldlist to 2 dp and place into newlist
oldlist = [0.121, 0.232, 0.465, 0.987]

#standard for loop
newlist = []
for num in oldlist:
      newlist.append(round(num,2))

#list comprehension
newlist = [round(num, 2) for num in oldlist]
```

Listing A.1: List comprehension example

It is also possible to take some of the computation up a loop level. For example, calculating the `kappa_percentile` does not rely on either of the other loop values, and therefore can be taken to the first level of the loop. Similarly, the values for `alpha` only require information from `kappa`, and this can therefore be moved to the second loop. The training section only takes around 5 seconds in total to complete a single iteration, but this only needs to be computed around 15 times (once for each configuration of $\kappa$ and $\alpha$) rather than 45 (including $\alpha$), leading to a potential speedup of 150 seconds. Over all 19 windows, this equates to around an hour total removed from the runtime. These changes can be seen in a shortened version of the complete code in Listing A.3.

Furthermore, the efficiency of locating the values of $\alpha$ can be increased. Instead of incrementally searching values, a binary search can be employed. For each estimated value of $\alpha_\pm$, if the number of sentiment words calculated using this parameter is too large, then $\alpha_\pm$ is too small, otherwise it is too big. To shorten computation, by halving the change in value each time, the possibilities are also halved, reducing time from $O(n)$ to $O(\log(n))$. The updated $\alpha$ calculation is shown in Listing A.4

Finally, I opted to use multithreading to calculate each of the $\lambda$ values at the highest loop level. This algorithm lends itself to multithreading because the order in which the calculations are returned do not matter. For this reason, I chose to run each configuration of $\lambda$ on its own thread. The updated code for calculating $\lambda$ can be seen in Listing A.5.

To quantify the speedup of the various steps, I ran a series of experiments, and timed the computation.

```python

for alpha in alpha_configs:
    for KAPPA in kappa_configs:
        for lam in lambda_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
            num_pos_words = len([w for w in total_j if f[w] >= train_pi + ALPHA_PLUS and total_j[w] >= kappa_percentile])
            num_neg_words = len([w for w in total_j if f[w] <= train_pi - ALPHA_MINUS and total_j[w] >= kappa_percentile])
            while(num_pos_words > alpha):
                ALPHA_PLUS += 0.0001
                num_pos_words = len([w for w in total_j if f[w] >= train_pi + ALPHA_PLUS and total_j[w] >= kappa_percentile])
            while(num_neg_words > alpha):
                ALPHA_MINUS += 0.0001
                num_neg_words = len([w for w in total_j if f[w] <= train_pi - ALPHA_MINUS and total_j[w] >= kappa_percentile])
            sentiment_words = [w for w in total_j if f[w] >= train_pi + ALPHA_PLUS and total_j[w] >= kappa_percentile])

            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)]

            (s, d_s)    = calc_s(sentiment_words, train_d)
            h           = calc_h(sentiment_words, train_d, s, d_s)
            O           = calc_o(p,h)

            # VALIDATION
            error_arr = np.array(0)
            for val_index in range(len(val_d)):
                est_p = 0.5
                val_bow = val_d[val_index]

                testing_s = sum(val_bow.get(w,0) for w in sentiment_words)
                if (testing_s > 0):
                    est_p = fminbound(equation_to_solve, 0, 1, (O,val_bow, sentiment_words,testing_s,LAM))
                error_arr = np.append(error_arr, est_p - val_p[val_index])
            normalised_error = np.linalg.norm(error_arr, 1)
            lam_trial = {
                'alpha': alpha,
                'alpha_plus': ALPHA_PLUS,
                'alpha_minus': ALPHA_MINUS,
                'kappa': KAPPA,
                'lam': LAM,
                'o': O,
                'sentiment_words': sentiment_words,
                'norm_err': normalised_error
            }
            trials.append(lam_trial)
            best_config = min(trials, key=lambda x:x['norm_err'])
```

Listing A.2: Original training and validation window

I tested a single configuration for each configuration of $\alpha$, as this has the most impact on computation time (as intuitively, the more words in the word list, more computation required in validation). The results in Table A.1 are calculated using an Intel Core i5-4460 CPU. Each cell in the table refers to the average runtime for all $\lambda$ configurations for $\kappa = 86$, and $\alpha$ is as defined in the table over 5 repetitions. The total speedup refers to the speedup from the original to the most optimised (multithreading).

```python
for KAPPA in kappa_configs:
    kappa_percentile = np.percentile(np.array(list(total_j.values())),KAPPA) # return the nth percentile of all appearances for KAPPA
    for alpha in alpha_configs:
        #calculate alpha vals
        ...
        for lam in lambda_configs:
            ...
```

Listing A.3: Moving computation up loop levels

```python
ALPHA_PLUS  = train_pi/2
ALPHA_MINUS = train_pi/2
delta_plus  = train_pi/4
delta_minus  = train_pi/4
delta_limit = 0.0001 # set a limit on number of searches
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 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 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 A.4: Binary search for alpha values

| Alpha value | Original | List Comprehension | Multithreading | Total speedup |
|---|---|---|---|---|
| 25 | 146s | 93s | 43s | 3.93X |
| 50 | 313s | 208s | 75s | 4.17X |
| 100 | 390s | 353s | 150s | 2.6X |

Table A.1: Speedup for each optimisation

```python
def validate_window(index, val_d, val_p, sentiment_words, LAM, trials):
    error_arr = np.array(0)
    # print(str(index) + " computing lambda of " + str(LAM))
    for val_index in range(len(val_d)):
        est_p = 0.5
        val_bow = val_d[val_index]

        testing_s = sum(val_bow.get(w,0) for w in sentiment_words)
        if (testing_s > 0):
            est_p = fminbound(equation_to_solve, 0, 1, (O,val_bow, sentiment_words, testing_s ,LAM))
        error_arr = np.append(error_arr, est_p - val_p[val_index])
    normalised_error = np.linalg.norm(error_arr, 1)
    lam_trial = {
        'alpha': alpha,
        'alpha_plus': ALPHA_PLUS,
        'alpha_minus': ALPHA_MINUS,
        'kappa': KAPPA,
        'lam': LAM,
        'o': O,
        'sentiment_words': sentiment_words,
        'norm_err': normalised_error
    }
    trials.append(lam_trial)

for kappa in kappa_configs:
      for alpha in alpha_configs:
            ...
            threads = []
            for index in range(3):
                x = threading.Thread(target=validate_window, args=(index,val_d, val_p, sentiment_words,lambda_configs[index],trials))
                threads.append(x)
                x.start()

            for index, thread in enumerate(threads):
                thread.join()
```

Listing A.5: Multithreadling Lambda

## Appendix B

# 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 B.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

| Positive | | | | | Negative | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Word | Sentiment | Count | LM | H4 | Word | Sentiment | Count | LM | H4 |
| week high | 0.023589 | 19 | 0 | 0 | downgrade | -0.023946 | 19 | 1 | 0 |
| volume mover | 0.018901 | 19 | 0 | 0 | loser | -0.016851 | 19 | 0 | 1 |
| upgrade | 0.016795 | 19 | 0 | 1 | lower | -0.016773 | 19 | 0 | 0 |
| gainer | 0.013524 | 19 | 0 | 0 | public offer | -0.007176 | 19 | 0 | 0 |
| high | 0.010034 | 19 | 0 | 0 | fall | -0.004345 | 19 | 0 | 0 |
| mover | 0.007526 | 19 | 0 | 0 | cut | -0.003035 | 19 | 1 | 0 |
| rais | 0.011851 | 18 | 0 | 0 | miss | -0.001481 | 19 | 1 | 0 |
| repurchase | 0.000298 | 17 | 0 | 0 | weak | -0.001191 | 19 | 1 | 0 |
| spike higher | 0.011535 | 16 | 0 | 0 | underweight | -0.000751 | 19 | 0 | 0 |
| volume | 0.002742 | 16 | 0 | 0 | low | -0.005291 | 17 | 0 | 0 |
| rumor | 0.00078 | 16 | 0 | 0 | public | -0.000609 | 17 | 0 | 0 |
| author | 6.8e-05 | 16 | 0 | 0 | neutral | -0.003327 | 16 | 0 | 0 |
| higher | 0.006567 | 15 | 0 | 0 | offer | -0.002486 | 16 | 0 | 0 |
| outperform | 0.002857 | 15 | 1 | 0 | negative | -0.000794 | 16 | 1 | 1 |
| spike | 0.002189 | 15 | 0 | 0 | offer common | -0.003422 | 15 | 0 | 0 |
| solid | 0.000432 | 15 | 0 | 0 | disappoint | -0.000759 | 15 | 1 | 0 |
| buy | 0.008344 | 14 | 0 | 0 | common | -0.000731 | 15 | 0 | 0 |
| green | 0.000711 | 14 | 0 | 0 | concern | -0.000485 | 15 | 1 | 0 |
| micron technology | 0.001827 | 13 | 0 | 0 | week low | -0.019854 | 14 | 0 | 0 |
| overweight | 0.001675 | 13 | 0 | 0 | resume trade | -0.003218 | 14 | 0 | 0 |
| soar | 0.001171 | 13 | 0 | 0 | secondary offer | -0.002503 | 14 | 0 | 0 |
| strong | 0.000545 | 13 | 1 | 0 | loss | -0.000699 | 14 | 1 | 1 |
| repurchase program | 6e-05 | 13 | 0 | 0 | remove | -0.000593 | 14 | 0 | 0 |
| move higher | 0.003403 | 12 | 0 | 0 | impact | -0.000458 | 14 | 0 | 0 |
| lift | 0.000659 | 12 | 0 | 0 | sector perform | -0.002324 | 13 | 0 | 0 |
| tender offer | 0.000494 | 12 | 0 | 0 | bed bath | -0.001656 | 13 | 0 | 0 |
| top gainer | 0.0146 | 11 | 0 | 0 | bath beyond | -0.001649 | 13 | 0 | 0 |
| time warner | 0.003665 | 11 | 0 | 0 | general dynamic | -0.001225 | 13 | 0 | 0 |
| urban outfitter | 0.002973 | 11 | 0 | 0 | tumble | -0.000678 | 13 | 0 | 0 |
| western union | 0.00197 | 11 | 0 | 0 | resign | -0.000534 | 13 | 1 | 0 |
| jump | 0.000856 | 11 | 0 | 0 | dip | -0.000478 | 13 | 0 | 0 |
| standpoint research | 0.000432 | 11 | 0 | 0 | worst perform | -0.01493 | 12 | 0 | 0 |
| special | 0.000154 | 11 | 0 | 1 | drop | -0.00074 | 12 | 0 | 0 |
| strength | 0.000147 | 11 | 1 | 0 | resume | -0.000661 | 12 | 0 | 0 |
| mention | 9.7e-05 | 11 | 0 | 0 | mix security | -0.000536 | 12 | 0 | 0 |
| alert call | 0.001606 | 10 | 0 | 0 | pressure | -0.000469 | 12 | 0 | 0 |
| marvel technology | 0.000905 | 10 | 0 | 0 | shelf | -0.00027 | 12 | 0 | 0 |
| chatter | 0.000873 | 10 | 0 | 0 | security shelf | -0.000228 | 12 | 0 | 0 |
| stake | 0.000815 | 10 | 0 | 0 | boston scientific | -0.003506 | 11 | 0 | 0 |
| jobless claim | 0.000489 | 10 | 0 | 0 | worst | -0.002948 | 11 | 1 | 1 |
| narrow | 0.000471 | 10 | 0 | 0 | sell | -0.000961 | 11 | 0 | 0 |
| pop | 0.000359 | 10 | 0 | 0 | secondary | -0.000194 | 11 | 0 | 0 |
| boost | 0.000324 | 10 | 1 | 0 | first solar | -0.009475 | 10 | 0 | 0 |
| dish network | 0.000207 | 10 | 0 | 0 | hold remove | -0.0027 | 10 | 0 | 0 |
| dynamic | 5.2e-05 | 10 | 0 | 1 | finish line | -0.001732 | 10 | 0 | 0 |
| western digit | 0.003229 | 9 | 0 | 0 | cliff natural | -0.001449 | 10 | 0 | 0 |
| alto network | 0.002602 | 9 | 0 | 0 | adobe | -0.001185 | 10 | 0 | 0 |
| office depot | 0.000438 | 9 | 0 | 0 | perform | -0.001148 | 10 | 0 | 0 |
| special dividend | 0.000385 | 9 | 0 | 0 | miss estimate | -0.00089 | 10 | 0 | 0 |
| expansion | 0.000243 | 9 | 0 | 0 | fitch | -0.000813 | 10 | 0 | 0 |

Table B.2: Top sentiment words including bigrams 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