Imperial College London

MEng Individual Project

Department of Computing

Imperial College of Science, Technology and Medicine

An aggregated sparse matrix factorisation model for market trading

Author:
Harry Brown

Supervisor:
Dr. Thomas Lancaster

Second Marker:
Dr. Konstantinos Gkoutzis

June 15, 2020

Submitted in partial fulfillment of the requirements for the Masters of Engineering degree of Imperial College London

Abstract

Thanks to the rise of social media, particularly Twitter, individual conversations around the globe are now taking place in earshot of any and all, accessible simply via a key word search. Armed with a variety of natural language techniques, this report presents a pipeline to combine these *tête-à-têtes* with content from news articles, and interpret their relevance to the stock market via a sparse matrix factoriser, a relatively new mathematical procedure for extracting meaningful features and patterns from data vectors. By building a separate evaluator for a variety of stocks, constantly monitoring their operations and allocating funds to the top performers, an aggregated trading model can be constructed.

Compared against the only two previous papers to have employed sparse matrix factorisation in a trading context, this model achieves a simulated annualised return of 40.6% and Sharpe ratio of 1.73 (pre-coronavirus). This is a marked improvement upon the 18% and 38.5% average returns, and Sharpe ratio of 1.35, reported in the two prior papers, and even more so when contrasted with a number of mainstream index funds. Furthermore, it separately demonstrates unprecedented resilience in periods of market uncertainty.

Acknowledgments

First and foremost I am immensely grateful for the invaluable support, guidance and encouragement from Thomas, my project supervisor and personal tutor, without whom the project and year as a whole would have been considerably less enjoyable.

Secondly, I am also indebted to Iain, Katie and Anthea, my Computer Science teachers at Saffron Walden County High School, whose fantastic teaching and unsurpassed enthusiasm for the subject drove me to study here in the first place.

Finally, a thank you to my parents, for forever putting my education first and providing me a platform to follow my dreams; to Josh, Jacob and Henry, for years of laughs, tears and entertainment; and to my friends, too many to mention, who have enriched my time at university no end.

Contents

Chapter 1

Introduction

1.1 Social media as a window to the world

Social media, particularly Twitter, provides a real-time source of news and information from across the globe. Breaking news stories are often first reported on social media before making their way to more traditional news outlets, and any users who find themselves at the scenes of stories can post reports, photos and videos, allowing anybody to be a journalist. A recent example might be the 2019 Hong Kong protests, where updates from residents on the ground provided round-the-clock coverage, when more traditional news outlets may have been less able to get access due to safety concerns or lack of transport [1]. Figure 1.1 shows a perfect example of this, a video of military vehicles assembling across the border in China that was watched globally over 6 million times; without Twitter, it is considerably less likely that such footage would have made it to the public domain.

Another feature of social media is how it can be used to quickly ascertain the general public mood relating to a specific topic. A search on Twitter of any current affairs story, brand or public figure immediately serves up multifarious commentary from all users who have mentioned this topic in any of their tweets. This provides an ever-growing library of mass public sentiment, a zeitgeist, on virtually any topic that can be fit into 280 characters or fewer, and can be considered one of the standout examples of what we today refer to as *big data*. One of the earliest sectors to engage with this wealth of information was in marketing and brand management, where by taking these tweets and feeding them through a sentiment analysis tool, it becomes possible to conduct a form of opinion mining [2][3], enabling companies to obtain insights on how people are talking about their business.

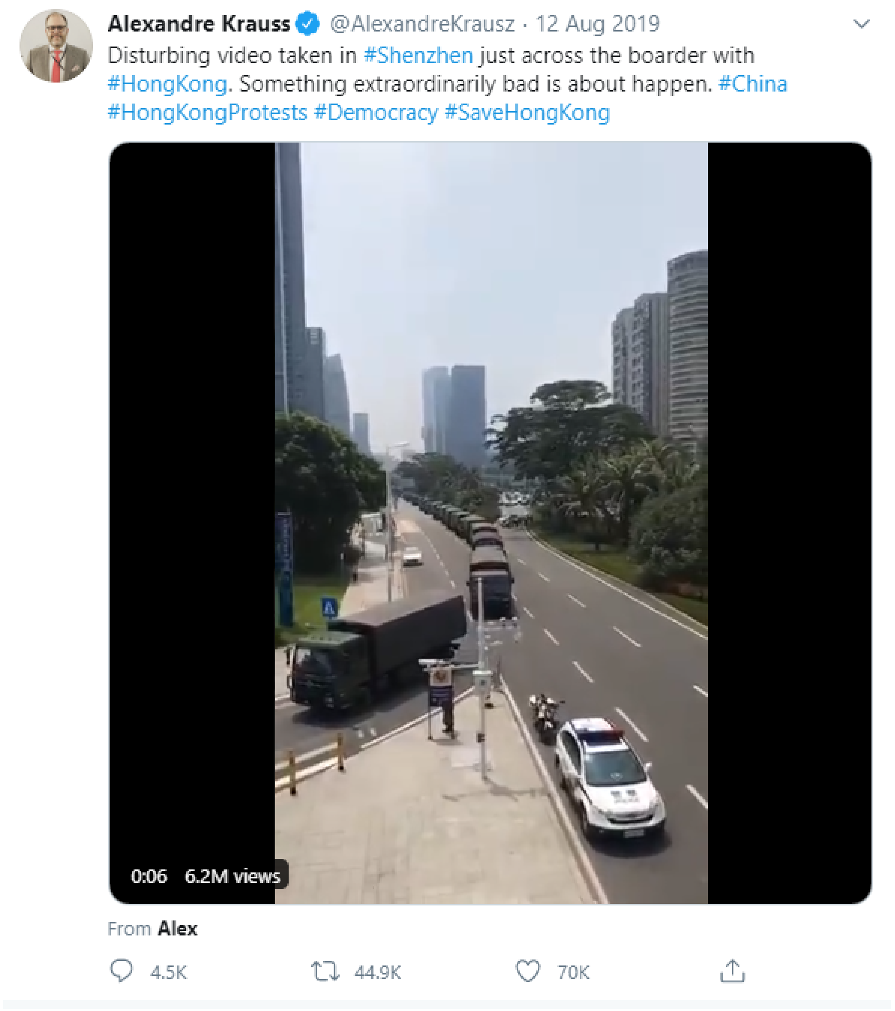

Alexandre Krauss @AlexandreKrausz · 12 Aug 2019
Disturbing video taken in #Shenzhen just across the boarder with #HongKong. Something extraordinarily bad is about happen. #China #HongKongProtests #Democracy #SaveHongKong

Figure 1.1: Example tweet from the 2019 Hong Kong protests

1.2 Applications in financial technology

With mass uptake of social media only really taking place in the last decade or so, academic research into new and unheralded uses of its data remains a key topic of interest. It has, among others, seen early signs of potential from the field of financial technology (or *fintech*), where improvements in processing power available to commercial users in the twenty-first century have allowed algorithmic trading to gain real traction over manual, human-led trading. Indeed, in 2016, over 80% of forex trades were carried out by computers, rather than people [4]. However, the majority of trading algorithms until recently have focused on mathematical regression-based methods; only in the last few years, coincidentally almost perfectly in sync with the rise of social media, has machine learning emerged as a genuinely new approach to trading. As such, the vast quantities of text data available through Twitter can now serve as a basis for a new trading strategy, one that eschews price or technical analysis altogether in favour of a text-based procedure, by aiming to spot patterns and correlations between the public voice and associated stock prices as its means of market prediction.

Whilst numerous investigations into the potential uses of social media as a basis for making investment decisions have already been carried out, many of these have been conducted with slight caveats. Some [5][6] use finance-based forum boards as their data source, rather than tweets; while this approach simplifies much of the data collection and processing stage, it inherently limits the range of opinions and voices being taken into account. Considerably less research has looked at tapping into “the voice of the public” - everyday users of social media - rather than specifically the views of people with a strong existing interest in stocks and finance. Twitter is a much better place from which to draw such data. Others [7][8] are specifically targeted at news articles, which while containing much richer texts from which to derive content are not considered entirely real-time, although the results from a number of these, particularly in the work of Ming, Wong, Liu and Chiang [9], are highly encouraging.

1.3 Project goal

The proposal for this project is therefore to combine the best of both worlds seen in the literature, taking in both tweets and news articles, analysing trends in the texts, and using these to form the basis for predicting the future movements of stocks.

The public and free-to-access nature of social media means that, just as desired, it is now trivial to obtain the world’s opinion on a subject matter of choice, but the flipside to this is that content is generated by users with all manner of intentions. To build a more well-rounded model that is shielded against some of the issues caused by this, it makes sense to instead aggregate text content from multiple sources - not just Twitter, but also from reputable news outlets that, while providing much less content to work with than Twitter, carry considerably more weight than individual tweets. The trends analysed may be in the form of sentiment analysis, word-frequency analysis, tweet volumes or otherwise. These trends will form the basis of a model, which will then be used to predict future movements based on new data.

The effectiveness of the method will be measured by taking the predictions of the model and laying these over the historical stock price movements, to produce a trading simulation. The individual stock performances in the model will be compared against both one another and against any similar models found in the literature. By then performing a more low-level analysis on the individual models, underlying trends between stocks may be obtained that aid future decision-making. It may well then transpire that these trends are not only detectable between stocks, but also among stock sectors.

Of course, it is unlikely that such an approach would work for all stocks. Specifically, the stocks being invested in must have a relatively unique name, such that any searches for them do not throw up results for unrelated content; and they must be embedded enough in the public consciousness to warrant regular discussions and comments on social media.

1.4 Ethical considerations

The *partially* free nature of Twitter's data makes the uses of its content an interesting ethical debate. Its terms of service specifically state that users' public posts will be made available to third parties, and it is not possible to open an account on the platform without agreeing to this. Nonetheless, data extracted from the website contains personal information, meaning it is subject to relevant data protection legislation such as the EU's General Data Protection Regulation (GDPR). This typically would require informed consent from users to allow researchers to use their data in their work. Of course, on a project of the scale envisaged here, obtaning consent from every single user whose tweets are collected is simply infeasible. [10] notes that, during a previous project involving collecting Twitter data for infectious disease research, no efforts were made to obtain informed consent from users on the assumption that it would be overly labour-intensive and would likely garner few, if any, responses. They remark that this "highlights the need for researchers to work alongside social media companies, for instance, asking users at the sign up phase whether they are OK with their content being used for research purposes".

A 2014 report [11] investigated users' attitudes to their own social media posts being collected for scientific research. Users who believed that obtaining consent to collect their social media data was unnecessary stated two main reasons why: firstly, the responsibility falls ultimately upon the user to determine the privacy levels of their post, and indeed whether to post at all; and secondly, if the platform used makes it clear that posts will be public, obtaining consent to retrieve these becomes redundant. Those of the opposing view stated variously that obtaining consent helps to promote trust between the researcher and participants, ascertains whether or not the post was intended to be posted publicly before using it, or allows users to evaluate the quality and purpose of the research before allowing their data to be used.

In [12] it is stated that, in cases where it is not possible to obtain informed consent from users, "a social researcher should establish the fair and lawful basis for collecting personal information". Thus: any tweets collected as part of this project will have virtually all personally-identifying information, such as the username or geolocation, removed from the tweet data as soon as possible. This takes place either directly at the point of requesting the data, or once it has first been stored, whichever option is feasible within the constraints of the API(s) to be used. This leaves solely the text body of the tweet as the only data to be carried forward into the pipeline. As soon as is then possible within the pipeline, after the tweet bodies have been transformed into whatever data format is required, the original tweets will be deleted altogether. At no point will individual users be identified in the data analysis, nor will any effort made to retrieve further information about the users.

1.5 Report structure

In Chapter 2, the technical background to this project is presented. Various techniques are described in detail for text analysis and information extraction, with some discussion of how these can be applied in a price prediction context. A few trading concepts are also included, to provide some background for the later evaluation stages.

In Chapter 3, various papers making use of these techniques are reviewed. The reported effectiveness of each strategy is taken into consideration, and thoughts are outlined about how these methods could then be applied to the goal of this project. Some focus is also given to shortcomings of these papers, looking for gaps in the methodologies that could be improved upon.

After deciding upon a strategy based on the conclusions of Chapter 3, Chapter 4 presents a detailed look at the methodology to be employed in this report. Each individual component of the pipeline is thoroughly explained, as well as the evaluation procedures that will follow.

Chapter 5 sees these evaluation procedures put into practice, both presenting the results and also offering an insight into why they come out the way they do. The results are tackled from a multitude of angles to give the best possible insight into the workings of the model. Comparisons are made against similar results obtained in some of the papers studied in Chapter 3.

Finally, Chapter 6 reviews the overall project, looking at some of the key takeaways and deliberating where it can be taken next.

Chapter 2

Technical Background

2.1 Overview

The concept itself of using social media data to forecast stock movements is something that has been explored in a number of ways in the last couple of years. In this section, an overview is presented of a variety of approaches tried and their underlying technical bases. Considerations will be made as to how each method can be applied specifically to obtain price predictions for stock movements. Not all of these approaches will be desirable for this project; some will have methodologies applicable only in certain setups, while others will have been explored numerous times already in literature and do not warrant further research. Accordingly, in Chapter 3, a literature review follows up on these approaches by studying how well they have performed in existing papers, and looking at potential flaws in the methodology that could be improved upon.

2.2 Natural language processing

The most obvious starting point for text-based analysis is to look at natural language processing (NLP) techniques. Broadly, natural language processing covers a range of procedures for programmatically storing and analysing text-based information. Early forms of NLP used handwritten grammars and rules to interpret inputs, but technological breakthroughs now allow for statistical-based interpretations, usually via machine learning. Today, natural language processing covers a number of different tasks; ones relevant for the direction of this project will be introduced here.

2.2.1 Part-of-speech tagging

Perhaps the best place to start is with part-of-speech (POS) tagging. This is the process of taking an input of text and marking each word with its part-of-speech, i.e. whether it is a noun, verb, article, and so on. For a simple sentence such as this,

We played football with our friends.

it is trivial to match up each word to its part of speech; ‘we’ is a pronoun, ‘played’ is a verb, ‘football’ is a noun, and so on. However, now consider this small change:

We played football at the park with our friends.

Everything remains fine until it comes to categorising the word ‘park’. As humans, we can instinctively see that in this case, ‘park’ is a noun, but of course park also exists as a verb. In this case, the presence of the determiner ‘the’ in front of ‘park’ gives away that it is a noun, but things are rarely always this clear-cut.

The Duchess was entertaining last night.

Is ‘entertaining’ a verb or an adjective?

A naive approach to POS tagging actually turns out to be surprisingly successful. By simply assigning each word its most frequent POS tag, and then assigning all unknown words to “proper noun”, tagging models can achieve accuracies of over 90% [13]. This works as many words are unambiguous, and out of those that aren’t, most overwhelmingly take on one specific part-of-speech. Improving on this approach is a form of probabilistic tagging, whereby one also considers prior words in order to come to a conclusion, and then further so with hidden Markov models, eventually arriving at an accuracy of around 97% [14]. Such functionality is now readily available to anyone via POS tagging libraries such as SpaCy and NLTK.

2.2.2 Lemmatisation

Lemmatisation is the process of taking an inflected or conjugated form of a word, and returning its base form, known as its *lemma*. This allows all forms of a word to be analysed as a single unit.

For example, consider the sentence

Nobody saw the Prime Minister for three weeks.

Using the part-of-speech tagger introduced in the previous section, it is now trivial for a model to parse each word and return its root form. The POS tagger, for example, will identify that 'saw' is a verb, rather than a noun, and so with this information the lemmatiser can correctly return the root form 'see'.

2.2.3 Term frequency - inverse document frequency (TF-IDF)

How is it possible to measure how relevant a word is to a forum post, tweet or document of some kind, in a larger overall collection of texts? One approach that can be used is TF-IDF (term frequency - inverse document frequency). This is in fact a product of two separate metrics, the *term frequency* - how many times a word appears within a text - and the *inverse document frequency*, the rarity of the word across the entire set of texts. The idea behind TF-IDF is that the number of times a word appears in a text should correspond directly to its perceived importance in that text, but is offset by the number of texts overall that contain that word. This has the effect of giving a higher score to words with a high frequency in a small number of texts, and a lower score to common words that have a high frequency across all texts (for example, 'a', 'the', 'as', ...) [15].

The two components of TF-IDF can be expressed in a number of ways.

For term frequency $TF(t, d)$, the simplest approach is to use number of occurrences of the term t in the text d.

$$TF(t, d) = freq(t, d)$$

Slightly more advanced variations include using a logarithmic scale [16],

$$TF(t, d) = log(1 + freq(t, d))$$

The inverse document frequency $IDF(t, D)$ of a term t in a collection of texts D is given by

$$IDF(t, D) = \log \frac{|D|}{1 + |\{d \in D : t \in d\}|}$$

i.e. the logarithm of the size of the corpus divided by the number of texts where the term t appears.

The TF-IDF is thus simply given by

$$TFIDF(t, d, D) = TF(t, d) \times IDF(t, D)$$

2.2.4 Sentiment analysis

The general process of sentiment analysis essentially involves mining a piece of text in order to obtain some sentiment score, a subjective indication of how the text views its subject matter. The text document is firstly broken down into various smaller parts, such as sentences, within which phrases carrying sentiment are identified. This is done using some form of sentiment library or sentiment dictionary, a large collection of words that have been manually scored by humans. One such example is the extended ANEW dictionary [17], a database of nearly 14,000 words and phrases. Usually, this sentiment score is a simple mark of ‘positivity’ falling between -1 and +1, although more in-depth approaches to sentiment analysis can produce a matrix of different feelings, such as ‘pleasant’ / ‘unpleasant’, ‘active’ / ‘subdued’, ‘relaxed’ / ’nervous’ and so on [18]. The scope of the sentiment analysis may vary.

- A **document-level** sentiment analysis would aggregate the sentiment scores collected above across the entire input text to produce a single output score;
- **sentence-level** sentiment analysis would produce one result per sentence;
- and then also **sub-sentence-level** sentiment analysis might look to provide a score for each clause of each sentence, say.

Regardless of the level of scope used, there are a number of issues that may be encountered when trying to apply this basic level of sentiment analysis to a sentence. For example, consider the final sentence here from *Wired*’s review of the Google Pixel 4 smartphone [19]:

> *... you can get a lot more features for your money if you go elsewhere – the OnePlus 7T is just one example: more than £100 cheaper, has greater storage as default and minimal Android bloat.*

A basic sentence-level or even sub-sentence-level sentiment analysis as described above might be able to detect the negative sentiment towards the Pixel 4, and the positive sentiment in this sentence regarding the OnePlus 7T. However, what if one were specifically in the market for a smartphone with a good camera, and don’t particularly care much for the storage options? The sentiment analysis here is too broad to provide any real insight into which phone might be best. These specific features of the phone - known as *aspects* - would give a much greater depth of information were we able to detect the individual sentiment towards each one, rather than just the phone overall. This concept forms the basis of aspect-based sentiment analysis.

2.2.5 Aspect-based sentiment analysis

Now that the basics of sentiment analysis have been explained, and some of its limitations have been acknowledged, it is time to move on to look at aspect-based sentiment analysis. In this context, an *aspect* is a property or feature of a *named entity*. Named entities can be divided into two: a standard *named entity* is typically a product, good, service, employee, etc that is made, sold, manufactured, employed by, ..., a parent *'higher-order' named entity*. For example, Apple would be considered a higher-order named entity, and the iPhone would be one of its associated named entities. But each of these named entities also have their own features and properties, and these are what are defined as an aspect. The aforementioned iPhone will have the aspects screen, battery, camera and so on. It is the calculation of the sentiment for these individual elements to then propagate back to the higher-order named entity, rather than on a more general sentence- or document-level basis, that distinguishes aspect-based sentiment analysis [20].

With all of this in mind, it can be concluded that aspect-based sentiment analysis is the process of deriving opinions of individual aspects within sentences, in order to asses whether there is an overall positive, negative or neutral sentiment towards them, and then use this information to better inform our sentiment calculations for the higher-order named entities.

Next it is important to know how to relate aspects in a piece of text to a specific named entity. This process is known as named entity-aspect aggregation. There are various approaches for this but one is outlined here, involving an unsupervised learning algorithm. A first pass through the text obtains a list of all named entities and all potential aspects. Then, a relation likelihood map is created, a series of values suggesting how likely it is that an aspect is related to any named entity. By default all are initialised to zero.

Each aspect is then passed through, and has its likelihoods updated for each named entity. This process follows a similar approach to Bayesian updating. Consider $P(R)$, the prior named entity-aspect relation likelihood, weighted by previous knowledge of the named entity $\alpha(R)$; and $P(RID)$, the likelihood of the relation being present in the document, weighted by the salience of the named entity in the document $\beta(R, D)$ (see section 2.2.3).

$$P(R|RID) = \frac{\alpha(R)P(R) + \beta(R,D)P(RID)}{\alpha(R) + \beta(R,D)} \tag{2.1}$$

Now that the likelihoods of a relation between a named entity and an aspect have been obtained, the true relations may be identified using thresholds. These thresholds are largely at the discretion of the user, but one approach is to find all named entities within a sliding sentence window and select the top named entity that meets a minimum likelihood value for the given aspect.

After identifying the true relations, it is possible to calculate the sentiments of each of the aspects. Then, the sentiment of all aspects is aggregated back to the named entity, to give an overall sentiment of the entity in light of the knowledge of its aspects. These aspect sentiments are combined with existing information about the named entity, weighting both as necessary.

2.2.6 Considerations for using natural language processing for price prediction

The most straightforward potential application of NLP for price prediction is likely to revolve around making guesses based upon the sentiment scores for various entities and their aspects. It is plausible that some correlation exists between rises and falls in sentiment scores for a particular company and corresponding movements in this company's stock price.

However, sentiment analysis is by no means perfect, and this introduces an extra layer of uncertainty for any prediction made by this method. In the literature review in Chapter 3, the efficacy of sentiment analysis-based methods for price prediction are considered.

2.3 Latent Dirichlet allocation (LDA)

Latent Dirichlet allocation (LDA), first introduced in [21], is a "generative probabilistic model", used in text mining to perform topic extraction. It is a way of representing the *composites* (in this case, collections of texts) as mixtures of topics that contain *parts* (words and / or phrases - a phrase of *n* words is referred to as an *n*-gram) with certain probabilities.

2.3.1 (Collapsed) Gibbs sampling

Gibbs sampling is a method of sampling from a joint distribution, which otherwise would be difficult to directly sample from. If a sample is desired from, say, a binomial or multinomial distribution, random numbers may simply be generated from the relevant range. What then for a multinomial distribution over more than one variable? Sometimes the distribution can be factorised if the variables are independent, i.e. $P(X_1, X_2, ..., X_n) = P(X_1) \times P(X_2) \times ... \times P(X_n)$. However, this is not always possible, in which case one may turn to Gibbs sampling. Here, samples are iteratively drawn from the conditional distribution of each X_i, a large number of times (k). After long enough, this leaves the group of tuples $(X_1, ..., X_n)_0, (X_1, ..., X_n)_1, ..., (X_1, ..., X_n)_k$. The samples approximate the joint distribution of all variables, and the expected value can be estimated by averaging over all samples. In the context of LDA, the sampler needs to compute the probability of a topic z being assigned to a word w_i, given all other topic assignments to other words [22].

The process of Gibbs sampling over a large number of variables is particularly computationally intensive. For some of these conditionals, it is possible to integrate out one or more variables to simplify the algorithm. This process is known as collapsed Gibbs sampling [23], and is useful as it accelerates the convergence to the target distribution (although comes with its own caveats, such as having a slightly altered target distribution). Collapsed Gibbs sampling forms the basis for LDA.

2.3.2 The LDA algorithm

Take a set of composites and a fixed number K of topics to extract, and do the following:

1. Pass through each composite, and randomly assign each part to one of the topics.
2. For each composite c,
   (a) For each part w,
      i. For each topic t, calculate the proportion of parts in c currently assigned to t, $P(t|c)$, and the proportion of assignments to t over all composites from w, $P(w|t)$. Recalculate w's topic by selecting a new topic t with probability $P(t|c) \cdot P(w|t)$.

The trick with LDA comes with how it assumes the composites are formed. It is a somewhat flawed set of assumptions, but provides a simplified model of how a piece of text may be written:

1. Select a unique set of parts. In this case of dealing with text data, this would mean effectively selecting a dictionary from which all words will come.
2. Select how many composites (items of text) to produce.
3. Select how many parts we want per composite (how many words per item of text), N. This is typically sampled from a distribution rather than a fixed number.
4. Select the number of topics to produce, k.
5. By sampling from the Dirichlet distribution over the set of k topics, $X \approx \prod_{i=1}^{k} {x_i}^{\alpha_i - 1}$ with some positive value of α, generate the probabilities of each part per topic (1), and then for each topic per composite (2).
6. For each composite, sample a topic based on the probabilities in (2), and then sample a part for this topic based on the probabilities in (1). Repeat until N parts are obtained.

The value of α used in (1) controls the distribution of parts per topic; a higher α results in a greater variety of parts, and vice versa. The α used in (2) controls the number of topics in a composite; the higher the value of α, the greater the mixture of topics. Because individual tweets are likely to be written about a single subject matter, a smaller number of topics per composite are desired, and of course not every word features in a tweet, so both α would be set to a small value (below 1).

The LDA algorithm produces two matrices as output. The first describes the probability of selecting a particular part when sampling a given topic, and the second describes the probability of selecting a particular topic when sampling a given composite. A simple example of the algorithm and the two output matrices is provided in table 2.1. In this example [24], there are four documents, consisting of (0) four club symbols, (1) four heart symbols, (2) four diamond symbols, and then (3) four of each of the above. The two tables are the output of the LDA algorithm. On the right is the probability of selecting a particular topic when sampling a given composite, and on the left the probability of selecting a particular part when sampling this topic.

2.3.3 Topic sentiment LDA (TSLDA)

Latent Dirichlet allocation is a powerful tool, but is not able to calculate both sentiment and topic categorisation simultaneously. In Nguyen and Shirai's *Topic Modeling based Sentiment Analysis on Social Media for Stock Market Prediction* [6], a new topic model, Topic Sentiment Latent Dirichlet Allocation (TSLDA) is proposed which can capture both.

TSLDA starts with the assumption that one sentence expresses only one topic, and one opinion on that topic. Topics are nouns, while opinion words are adjectives or

	Topic 0	**Topic 1**	**Topic 2**
♣	0.00	0.00	0.99
♡	0.99	0.00	0.00
♢	0.00	0.99	0.00

	Topic 0	**Topic 1**	**Topic 2**
Document 0	0.03	0.03	0.93
Document 1	0.93	0.03	0.03
Document 2	0.03	0.93	0.03
Document 3	0.33	0.33	0.33

Document 0	♣♣♣♣
Document 1	♡♡♡♡
Document 2	♢♢♢♢
Document 3	♣♣♣♣♡♡♡♡♢♢♢♢

Table 2.1: A simple visualisation of the LDA algorithm

adverbs. Every word in the document is classified into one of three categories c: a *topic word* ($c = 1$), an *opinion word* ($c = 2$), and then all others ($c = 0$). Every topic has an associated word distribution, i.e. other words which tend to appear in the context of this topic, and also an associated opinion word distribution, as the same qualifier may have different meanings depending on context. A full mathematical proof of the method is found in the paper [6], but to summarise, any nth word w in a sentence m and in a document d is drawn from the distribution

$$w_{d,m,n} \begin{cases} Multinomial(\Phi^b) & \text{if } c_{d,m,n} = 0 \\ Multinomial(\Phi^t{}_{z^t{}_{d,m}}) & \text{if } c_{d,m,n} = 1 \\ Multinomial(\Phi^t{}_{z^t{}_{d,m}, z^o{}_{d,m}} & \text{if } c_{d,m,n} = 2 \end{cases} \tag{2.2}$$

where Φ^b, $\Phi^t$and Φ^o are the distributions over background, topic and sentiment words respectively, and $z^t{}_{d,m}$and $z^o{}_{d,m}$ are the topic and sentiment assignments for sentence m in document d.

2.3.4 Joint sentiment/topic (JST) model

Introduced in [25], the joint sentiment/topic model (JST) is an unsupervised probabilistic modelling framework, based on LDA, that detects sentiment and topic simultaneously from text.

The predominant difference between LDA and JST is that JST associates a document with multiple sentiment labels, each associated with its own topic distribution, whereas LDA used only one document-specific topic distribution [6]. This feature provides a way for the JST model to measure the sentiment of topics. Documents are constructed by drawing a word from a distribution over words defined by the topic *and sentiment label*, in contrast to LDA which draws only from the topic. Otherwise, the two methods are largely similar.

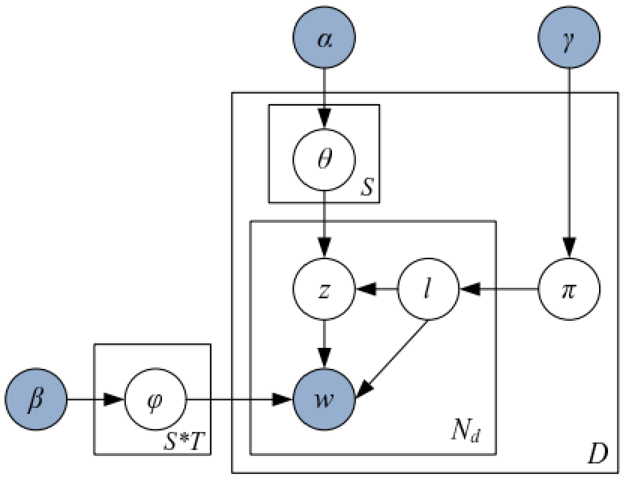


Figure 2.1: Graphical representation of JST

2.3.5 Considerations for using (TS)LDA for price prediction

Using TSLDA, it is possible to capture topics and sentiments of a document simultaneously. Once the sentiments are obtained, the process from here onwards largely follows that of one outlined in the sentiment analysis section. One potential limitation of this method is that the number of topics to extract, K, must be specified beforehand, rather than being determined during the process. Without actually reading all of the news content in advance to see how varied the discourse is, how possibly can the number of topics to divide the text into be known? A grid- or random-search approach, considering a variety of values for K for each individual stock, will likely be required.

2.4 Sparse matrix factorisation (SMF)

First introduced in a social media text analysis context in [9], the sparse matrix factorisation (SMF) model is a price prediction technique based on a characterisation of a stock (e.g. its sector) and the average investor mood on a given day. The price of a stock on any day is a function of the latent features of the stock and the investor mood at the time.

$$\hat{r}_{it} = u_i^T v_t \tag{2.3}$$

Each stock i is associated with a feature vector u_i, and the text data on a trading period t is associated with vector v_t. The dot product of the two vectors $\hat{r}_{it}$ captures the interaction between stock i and day t [26].

By then introducing a word frequency text vector y_t, v_t is replaced with Wy_t, where W is some mapping matrix to be learned. After dealing with overfitting, a prediction can be made for tomorrow's log return given today's text data.

2.4.1 Lagrangian method of multipliers

Understanding the methodology behind the SMF approach first requires knowledge of the Lagrangian method of multipliers.

The Lagrangian method of multipliers is a technique for finding the maximum or minimum of a multivariable function $f(x, y, ...)$, when there is some constraint on the input variables, $g(x, y, ...) = c$.

Firstly, the Lagrangian of the function g is introduced with use of a new variable λ,

$$L(x, y, ..., \lambda) = f(x, y, ...) - \lambda(g(x, y, ...) - c) \tag{2.4}$$

The variable λ is referred to as a Lagrangian multiplier.

Next, the gradient of L is set equal to 0, to find the turning points.

$$\nabla L(x, y, ..., \lambda) = 0 \tag{2.5}$$

For each solution i, with values $(x_i, y_i, ..., \lambda_i)$, calculate $f(x_i, y_i, ...)$. Whichever gives the largest (smallest) value gives the maximum (minimum) point.

2.4.2 Augmented Lagrangian method

The augmented Lagrangian method, or *penalty multiplier method,* is a technique to solve an optimisation problem with multiple inequality constraints [27].

Consider the problem

$$\begin{aligned} \text{minimise } & f(x) \\ \text{s.t. } & g(x) = 0 \end{aligned} \tag{2.6}$$

One way to look to solve this is via the introduction of a penalty parameter ρ.

$$\text{minimise } f(x) + \frac{\rho}{2}\|g(x)\|_2^2 \tag{2.7}$$

In order for this to produce an appropriate result, ρ must be allowed to grow very large, otherwise the constraint that $g(x) = 0$ will not be met. This is not always computationally feasible. Instead, the constraint $g(x) = 0$ can be relaxed by some value that allows much smaller values of ρ to be used [28]. The problem thus becomes

$$\begin{aligned} &\text{minimise } f(x) \\ &\text{s.t. } g(x) - \frac{\lambda}{\rho} = 0 \end{aligned} \tag{2.8}$$

Substituting this new constraint into (2.7) gives

$$\text{minimise } f(x) + \frac{\rho}{2}\left\|g(x) - \frac{\lambda}{\rho}\right\|_2^2 \tag{2.9}$$

$$= f(x) + \frac{\rho}{2}\|g(x)\|_2^2 - g(x)^T\lambda \tag{2.10}$$

(2.10) is known as the *augmented Lagrangian*, $L_A(x, \lambda, \rho)$. The method for solving this problem then proceeds thus:

1. Pick some values for λ and ρ
2. Minimise $L_A(x, \lambda, \rho)$ as a function of x
3. Using the new value of x, derive new values for λ and ρ
4. Repeat until convergence or a maximum number of iterations is reached.

2.4.3 Alternating direction method of multipliers (ADMM)

The alternating direction method of multipliers, or ADMM, is a variant of the augmented Lagrangian method. It can be used to solve the minimisation problem that underlies the SMF approach.

Consider a problem of the form

$$\text{minimise } f(x) + g(x) \tag{2.11}$$

By making a trivial adjustment to this problem,

$$\begin{aligned} &\text{minimise } f(x) + g(y) \\ &\text{s.t. } x = y \end{aligned} \tag{2.12}$$

a constraint has now been introduced that allows this problem to be solved via methods of constrained optimisation, i.e. the augmented Lagrangian method. The dual variables are updated one by one (i.e. solving for x with y fixed, and then solving for y with x fixed) before repeating the process again.

2.4.4 Bartels-Stewart algorithm

When applying ADMM in the context of SMF, one of the variable update steps involves a matrix equation that cannot immediately be rearranged for the right value (see Appendix A.2). Instead, by rearranging the equation to become of the form $AX + XB = C$, known as a Sylvester equation, it is possible to solve for X using the Bartels-Stewart algorithm, introduced in 1972 [29].

An overview of the Bartels-Stewart algorithm is as follows (proof can be found in [29] or [30]):

1. Compute the real Schur decompositions

$$R = U^T A U \tag{2.13}$$
$$S = V^T B^T V \tag{2.14}$$

   The Schur decomposition of a matrix A, $T = QAQ^{-1}$, produces the quasi-upper triangular matrix T, a matrix whose diagonal consists of 1×1 or 2×2 blocks, and whose eigenvalues are the diagonal values of A [31].

2. Solve $UF = CV$ for F, i.e. $F = U^T CV$
3. Solve $RY + YS^T = F$, where $Y = U^T XV$, via forward substitution
4. Obtain X via $X = UYV^T$.

2.4.5 Considerations for using SMF for price prediction

Sparse matrix factorisation particularly distinguishes itself from the two previous methods because its independence from sentiment analysis removes a layer of uncertainty from the prediction-making. The more reliable the data underlying the prediction methodology, the more reliable the results are as well. As will be seen in the next chapter, its applicability to trading in this such context is also far less researched when compared to sentiment analysis, making it a particularly interesting candidate approach.

2.5 Trading

As a bookend to this section, it may be useful to briefly adumbrate a few concepts within trading that will come up later in the project.

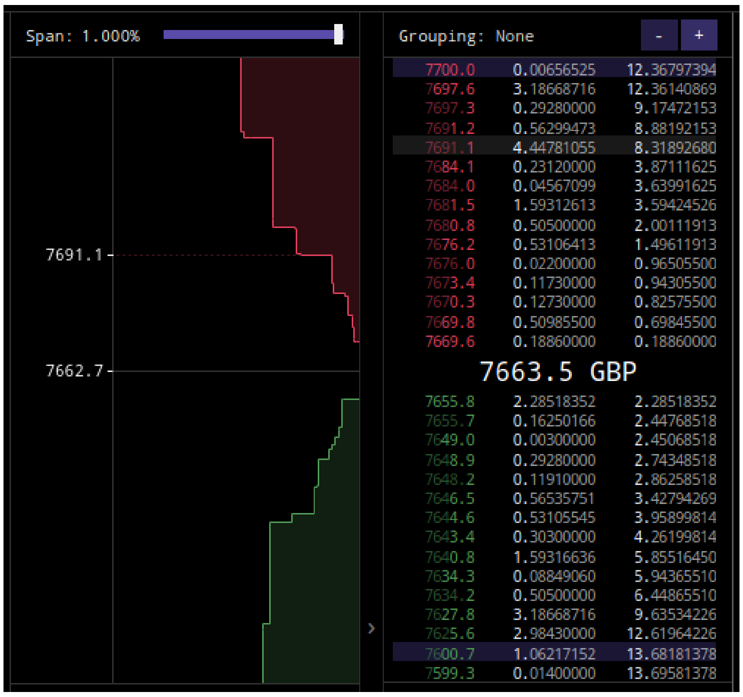
2.5.1 Longing vs shorting

The most intuitive and well-known action in market trading is buying, or *longing*, a stock. The stock is purchased at a given price (known as its asking price) and in a given quantity, and then becomes part of the trader's portfolio. As the value of the stock rises or falls over time, the value of the trader's investment rises or falls with it. At some point, the trader may choose to sell their investment. For an immediate sale, they can sell it at the highest bidding price, or otherwise they can list their desired price on an exchange and wait for a buyer to take it up. The aim, of course, is for the trader to make money by pulling off a sale at a higher price than what they paid for it.

Shorting, on the other hand, is the process of borrowing shares from a broker, selling them at the current bidding price, and then buying them back later - ideally at a lower price, allowing the trader to keep the difference as profit, but having to pay out extra if the shares are later bought back at a higher rate. Shorting is considered riskier than going long because while the greatest risk of the latter is simply to lose all initial capital (i.e. if the price should fall to zero), shorting can result in losing more money than is initially invested. If a trader has £10,000 and shorts that amount's worth of stock, and the stock doubles in price, the trader is sitting on a £10,000 loss - but if the stock then rises any further, the loss naturally rises with it, outstripping the trader's initial funds.

2.5.2 Trading fees

Trading fees can broadly be divided into two forms: a fixed-rate per-trade fee, or an implicit fee introduced via an enlarged bid-ask spread. A fixed-rate fee is simply a small sum added to every transaction; examples of brokers charging such fees include Barclays (£6/trade) or IG (£3-8/trade depending on prior usage). The alternative, an enlarged bid-ask spread, is a way around charging fixed fees that still enables the broker to make money. The bid-ask spread is simply the difference between the asking price and the bidding price for a stock - for example, at a given moment in time a stock may be purchased at £400, but sold for only £398, giving a bid-ask spread of £2. Some brokers, such as Plus500, fix this spread at a specific value, usually one slightly larger than found elsewhere, and take their profits from this. Such brokers tend to place "0% commission" as a central part of their advertising, reflecting the lack of any overt fees per transaction, but the enlarged spread can sometimes incur greater losses than an ordinary fee if the trader is not careful.



```
Span: 1.000%                Grouping: None

   7700.0   0.00656525   12.36797394
   7697.6   3.18668716   12.36140869
   7697.3   0.29280000    9.17472153
   7691.2   0.56299473    8.88192153
   7691.1   4.44781055    8.31892680
   7684.1   0.23120000    3.87111625
   7684.0   0.04567099    3.63991625
   7681.5   1.59312613    3.59424526
   7680.8   0.50500000    2.00111913
   7676.2   0.53106413    1.49611913
   7676.0   0.02200000    0.96505500
   7673.4   0.11730000    0.94305500
   7670.3   0.12730000    0.82575500
   7669.8   0.50985500    0.69845500
   7669.6   0.18860000    0.18860000
          7663.5 GBP
   7655.8   2.28518352    2.28518352
   7655.7   0.16250166    2.44768518
   7649.0   0.00300000    2.45068518
   7648.9   0.29280000    2.74348518
   7648.2   0.11910000    2.86258518
   7646.5   0.56535751    3.42794269
   7644.6   0.53105545    3.95899814
   7643.4   0.30300000    4.26199814
   7640.8   1.59316636    5.85516450
   7634.3   0.08849060    5.94365510
   7634.2   0.50500000    6.44865510
   7627.8   3.18668716    9.63534226
   7625.6   2.98430000   12.61964226
   7600.7   1.06217152   13.68181378
   7599.3   0.01400000   13.69581378
```

Figure 2.2: An example bid-ask spread for Bitcoin/GBP on the Kraken trading exchange.

In addition to the trading fees, stamp duty is charged at 0.5% on any shares purchased electronically in the United Kingdom.

2.5.3 Efficient market hypothesis

The efficient market hypothesis is a theory that states that a stock's price represents all public information about it, and that the only information that moves this price is therefore new information. If a piece of particularly negative news comes out about a company, its share price drops until this news has become "priced in"; it is then no longer useful for trading. This means neither technical (price) analysis nor fundamental analysis should, in the long run, produce any better returns than a randomly-selected portfolio of stocks to buy and hold. Beating the market on a risk-adjusted basis - i.e., formulating a trading algorithm based on news and tweets with a prediction accuracy of over 50% - should not be possible.

While the premise of the hypothesis is widely believed to hold true to this day, the implications derived from it are often disputed. Many economists today believe the markets are at least partially predictable. A considerable record of criticisms and evidence against the efficient market hypothesis is presented by BG Malkiel in [32], although interestingly after considering it all, he ultimately concludes that the markets are far *less* predictable than is sometimes believed, and do not present opportunities for particularly extraordinary rates of return.

2.5.4 Sharpe ratio

The Sharpe ratio [33] is a metric of risk-adjusted returns for a given fund. The higher a fund's Sharpe ratio, the better its returns have been relative to the risk.

$$S_a = \frac{E[R_a - R_b]}{\sigma_a}$$

where R_a is the asset return, R_b is the risk-free return, and σ_a is the standard deviation of the asset return. The Sharpe ratio is calculated for all stocks and, as it is based on standard deviation rather than straight values, can be used as a comparison tool between stocks to judge performance and desirability.

2.5.5 VaR and CVaR

Value at risk (VaR) and conditional value at risk (CVaR) are, as their names suggest, measures used to evaluate risk, with CVaR in particular being a useful approximation of potential losses of a stock. VaR and CVaR are most commonly measured at three different intervals - 95%, 99% and 99.9%.

VaR

If, say, VAR$(95) = 2\%$, this means there is a 5% (100 - 95) chance of losing 2% or more on a given day.

CVaR

If now CVAR$(95) = 3\%$, this means that in the worst 5% of returns, the average loss is 3%.

CVaR is usually preferred to VaR as it provides an average expected loss, rather than a whole range of potential losses.

2.6 Summary

A whole range of methods are available to use, each with their own strengths and weaknesses. Sentiment analysis seems the obvious place to start for such a project, but has been covered countless times already, and the sentiment estimates themselves, let alone the predictions they lead to, are not always accurate. Latent Dirichlet allocation is perhaps able to obtain more deeply-rooted sentiment than from a standard text classifier, but ultimately suffers the same drawbacks as the latter. It is the sparse matrix factorisation approach that seems the most promising - a relatively new method and one that avoids the uncertainty associated with sentiment-based techniques. In the next chapter, examples in existing literature are presented of each of these methods being used for stock market prediction, with the methodologies scrutinised and opportunities for improvement discussed.

Chapter 3

Literature Review

3.1 Overview

Each of the approaches for stock prediction reviewed in the previous section has already seen varying degrees of interest in existing literature. In this chapter, a round-up is offered of notable papers and the outcomes achieved in them.

The overarching theme of the papers collected in this section is their primary use of text data to inform investment decisions. The data sources themselves vary somewhat in style and quantity of data, so any papers using realtime social media are definitely of greater relevance to the plans for this project, but it is specifically the abilities of the models presented to predict stock movements that are of biggest interest at this stage. Some of the techniques described in Chapter 2, such as sentiment analysis, have been investigated in a considerable number of articles already, while only two known papers (at the time of writing) have attempted to study sparse matrix factorisation for text-based investment. At the end of the chapter, each method will be considered in the context of the goal of this project, and a decision will be taken on which one to bring forward to development.

3.2 Aspect-based sentiment analysis

Several attempts have been made to apply sentiment analysis to social media data for making stock predictions, with varying degrees of success. Nguyen, Shirai and Velcin's work [5] implements a sentiment analysis-based approach on messages from the Yahoo Finance message boards for 18 stocks, collected over a period of one year. On this particular website, the messages are categorised explicitly by the stock being discussed (e.g. AAPL), and it is possible for users to add a sentiment tag to their messages (one of five options - Strong Buy, Buy, Hold, Sell, Strong Sell). The existence of these sentiment tags and the fact that the messages are already clearly categorised by stock initially gave this approach some verisimilitude, by providing the authors with a fairly substantial advantage in that the need to use sentiment analysis is theoretically almost entirely negated. However, only 15.6% of messages were found to use the sentiment tag functionality, and so they did ultimately fall back on an aspect-based sentiment analysis approach in order to make use of the remaining messages. Issues with the accuracy of sentiment analysis have already been discussed in Chapter 3.

Regardless, the end result was that, in terms of prediction accuracy (how many price rises or falls the model guesses correctly), the aspect-based sentiment analysis outperformed regular sentiment analysis, LDA and JST (see sections 2.3 and 2.3.4) by 2.54%, 2.14% and 2.87% on average. The average prediction accuracy was a reasonable 54.41%, although being an average, on a number of stocks it did in fact perform worse than the aforementioned methods (and for a few stocks, actually performed worse than simply guessing). While it was encouraging to see aspect-based sentiment analysis used in this field, there are a number of potential issues with regards this project with the approach taken by the authors. The focus in this report to use Twitter as the primary data source stems from the desire to mine the opinions of the general public, and not just specifically people with a specific interest in investing. It is quite possible, even likely, that somebody posting on a stock's message board may well have a vested interest in the direction of the stock, and given a more niche stock, a particularly well-regarded user could even attempt to influence this direction themselves - perhaps less of a potential threat with traditional stocks but very much an ongoing problem in the world of cryptocurrencies [34][35]. There is no indication either as to how much better the model performed on text inputs that made use of the sentiment tag functionality, calling into question the usefulness of the Yahoo Finance message board as a data source.

3.3 TF-IDF as a feature weighting mechanism

In [5], TF-IDF is used in the process of classifying forum messages into one of five sentiment categories. By representing each forum message as a bag of words made up of the message title and body, and weighting each word by its TF-IDF score, the authors were able to determine which of the words in the messages were most relevant to estimating the messages' sentiment. This is an interesting approach to take and would translate well to the Twitter use case, where the messages would be replaced by tweets and the corpus simply the entire volume of tweets collected. However, other weighting mechanisms relating to the origin of the tweets collected are likely to be required in addition to this. A couple of questions that will require answering include:

- Should a higher weighting be assigned to tweets from accounts with greater influence (e.g. followers, average tweet engagement etc.)? Such a weighting would assume a positive correlation between influence and knowledge of the market in some way. Does this correlation actually exist?

- How are spam / bot accounts detected and what weighting, if any, should be assigned to their tweets?

3.4 LDA and TSLDA

An LDA-based method is also presented in [5], and is used as a price prediction baseline. The contents of all messages on the Yahoo Finance boards are divided into 50 topics, with probabilities on each transaction date deriving from the frequency of a given topic appearing alongside a price rise or fall. With 1,000 iterations and the 50 hidden topics, the LDA-based model achieved an average price movement prediction accuracy of 52.27%, slightly better than the JST-based model but worse than that with aspect-based sentiment analysis. In this example however, the LDA approach is perhaps critically flawed in that the number of topics to extract must be specified beforehand. When collecting a forum of messages, it is entirely possible that the entire thread would be a discussion of one single topic, meaning the decision to extract 50 topics would lead to very arbitrary categorisation. This limitation makes it very difficult to imagine a successful implementation of LDA for a Twitter-based approach.

Meanwhile, in [6], TSLDA is introduced in the context of predicting the direction of stock movements, rather than specifically the price. Five message boards from the stocks of Exxon Mobil, Dell, eBay, IBM and Coca-Cola are scraped for messages over a one-year period. Firstly, every word from every message is classified into one of the aforementioned three categories. All nouns are extracted first; next, any word that is not a noun but that appears in a list of opinion words in SentiWordNet [36] is classified as an opinion word; and then the remaining words are left as others. Just as with standard LDA, the TSLDA model is then trained using collapsed Gibbs sampling with 1,000 iterations.

3.5 Joint sentiment / topic

Joint sentiment / topic was introduced [25] in a more general sentiment analysis context, rather than specifically for social media analysis, but following papers have indeed applied it to social media, giving a useful overview of its effectiveness. In its introductory paper, the authors applied it to a dataset of film reviews, looking to classify them as either positive or negative. Incorporating no prior information, JST correctly identified 63% of positive reviews and 56.6% of negative reviews for an overall average of 59.8%. Various improvements to the algorithm, such as including a list of 42 “paradigm words” to scan for beforehand, the accuracy increases to 74.2% overall. The top-performing model used a filtered version of the MPQA subjectivity lexicon [37] to guide classification, achieving an overall accuracy of 84.6% (including an impressive 96.2% accuracy on positive sentiment classifications).

As with regular LDA, the number of topics must be specified beforehand. While this is feasible with a grid search approach for small numbers of stocks, [5] notes that *“since the running time of the Gibbs Sampling depends on the size of the dataset, it takes [a] very long time to run it repeatedly on ... 18 stocks for a long period. Therefore, a grid search cannot be tested in our experiment.”* Given that 18 stocks is probably at the lower end of any intended investigation in this report, the JST approach is likely to be problematic here as well.

In [6], JST is specifically used for sentiment analysis on social media data (see 3.4 for an explanation of the setup). Between using a price-only estimation, LDA, JST and TSLDA, the JST method returns the lowest total true positive identification rate of only 63%, some 10% behind the next-lowest. Interestingly however, it performs best in identifying true negatives; its 78% identification rate puts it 11% ahead of LDA and 12% ahead of price only and TSLDA. Overall it ranks comparably with the other methods.

3.6 Sparse matrix factorisation

Despite the limited research into its uses for text-based investing, sparse matrix factorisation is probably the method that has achieved the greatest success in literature, while also managing to avoid two of the biggest issues with the previously encountered methods. In [9], the paper in which it is first used in a text-based investment context, 3 years’ worth (1,008 trading days) of prices are used for the training set, a further 250 trading days for the validation set, and 188 trading days for the test set. The paper then analyses articles from the Wall Street Journal from these time periods, computing analyses on articles from a given day and then using this to form the basis of a prediction for the next day. Compared against three baseline data sets - stock indices, uniform portfolios and minimum variance portfolios - the SMF model was by some distance the highest performer, producing returns of a very impressive 56% in simulated trading for 2013 - over 30% more than the next highest, the NASDAQ index.

Similarly successful results are obtained in [26]. This paper particularly focused on social media, using StockTwits - a network designed in the style of Twitter but specifically for trading discussions - as its data source. The paper explicitly states that they considered using Twitter as the primary data source, but decided against it as “large amounts of Twitter content simply add noise to stock market prediction algorithms".

The historical close prices are obtained for 420 stocks between 2011 and 2015 (1,173 trading days). On the training set, the price direction was predicted correctly with an accuracy of 70.12%. Its calculated Sharpe ratio (see 2.5.4) of 0.38 was notably lower than that of three ETFs (0.97, 0.98 and 0.98). However, over the time period observed, the SMF strategy produced the greatest annualised returns when compared to the ETF funds and also to two portfolio holding strategies (an equally-weighted portfolio, and a minimum-variance portfolio), with an average annualised return of 18%.

Both of these implementations produce significantly higher returns than have been seen in previously-documented approaches. The sparse matrix factorisation method will be of considerable interest to this project going forwards.

3.7 Summary

While each method presented has its own benefits and drawbacks, there seems to be an increasingly clear candidate to take forward in this project.

- **Sentiment analysis** is a valid approach to achieve the goals of this project, but is already widely-researched and, without access to funds for cutting-edge sentiment analysis APIs such as Google Cloud Natural Language or Azure Text Analytics, will rely on sentiment classification algorithms which are themselves not always accurate,
- **(TS)LDA** presents an alternative method for deriving sentiment, but does not scale well larger data sets, where countless grid searches will have to be run to determine various hyperparameters,
- **JST**, likewise, does not scale easily to the scale envisaged for this project,
- **SMF** scales without an issue for larger numbers of stocks, does not require expensive APIs, and has seen comparatively little existing research.

On these grounds, an approach using sparse matrix factorisation will therefore be designed and described in detail in Chapter 4.

Chapter 4

Implementation

4.1 Overview

With sparse matrix factorisation elected as the method to take forward in this project, it is now time to detail its implementation, and provide a step-by-step overview of the full data pipeline required.

The pipeline shall be divided into three broad sections. The first stage is data retrieval. A stock of choice is defined by its name, ticker symbol, stock exchange it is listed on, and a start and end date to be used for analysis. Its related text data must then pulled from the sources of choice over the specified time period, using a mix of any available official APIs and hand-written scrapers otherwise. Historical stock prices shall also be collated from the relevant stock exchange for the same time period.

The second stage will involve processing the aforementioned data. The text data must ultimately be transformed to some numerical matrix before it can be used in the sparse matrix factorisation approach. Therefore, the text from each source shall be analysed, normalised and standardised to produce a word-frequency matrix, while the price data is used to produce a log-returns matrix for the same time period. These matrices will then feed to the minimiser, which will compute via ADMM (see 2.4.3) a prediction for the following close price of the stock.

At the end of the pipeline, a series of predictions shall be obtained. These can then be fed through a trading simulator to evaluate the model's performance in a real-world environment. Figure 4.1 provides a visual illustration of this proposed pipeline.

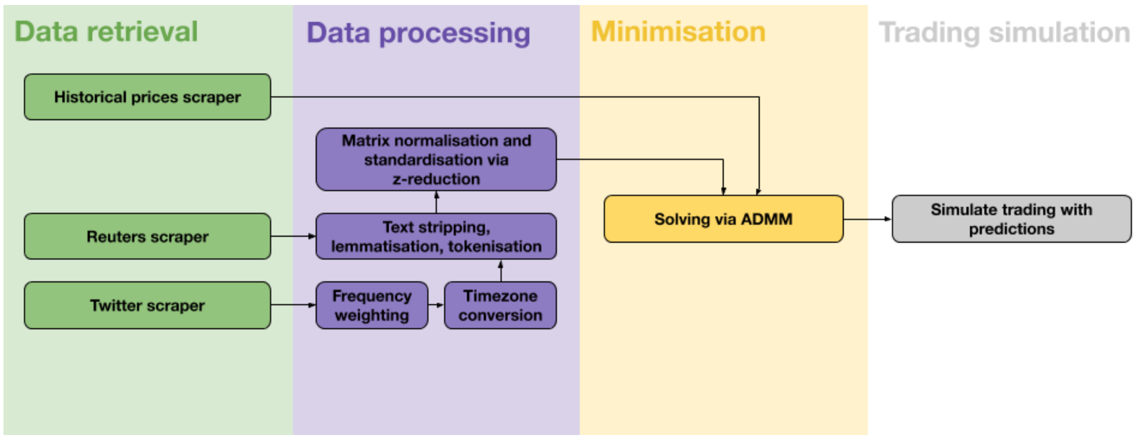

Data retrieval
Data processing
Minimisation
Trading simulation
Historical prices scraper
Matrix normalisation and standardisation via z-reduction
Solving via ADMM
Simulate trading with predictions
Reuters scraper
Text stripping, lemmatisation, tokenisation
Twitter scraper
Frequency weighting
Timezone conversion

Figure 4.1: High-level overview of the proposed three-stage pipeline, plus the simulation step

4.2 Stage 1: Data retrieval

The first step is of course to obtain the relevant data. When setting out the initial aims for this project, one of the key goals was that it would be predominantly based on Twitter data. While other investigations analysed in the literature review varyingly sought to use social media data from more niche social networks, or news articles from organisations such as the Wall Street Journal, as the basis of their data, one of the core aims for this project would be that it would tap into the "public mood" as a means of obtaining its content. [26] rightly raises a concern that Twitter data would be filled with everyday noise, and as such would not provide any particularly meaningful insight. This is true, but specifically is true for the methodology used in that paper (see section 3.6). It takes *all* text content from StockTwits, and uses it to predict the movements of 420 stocks simultaneously via a generalised model; the advantage of this approach is that this model can then be applied to new stocks that have very few (if any) mentions in the text corpus. However, here, the goal is to produce a specialised model on a per-stock basis. This is achieved by specifically polling Twitter for tweets including mentions of the desired stock. As such, this avoids the issue of background noise, as all tweets collected are guaranteed to be (at least somewhat) relevant.

That said, the matter of the Twitter data being at least slightly "noisy" remains a subject of interest. As early as 2011, reports were emerging of the rise in companies using Twitter to handle customer complaints [38]. A fair portion of any tweets mentioning a brand are potentially going to be angry customers looking for somewhere public to vent their frustration. While this noise is actually factored in to the model via standardisation (see 4.3.3), it was also of interest to include a second data source, consisting of less-frequent but higher-quality text data. It is important that this secondary source is as neutral and impartial as possible, as any political or economic slant in the news being reported may lead to important stories being covered in a partial manner. To this end, Reuters has been selected, as a high-output news source noted particularly for its impartiality [39].

4.2.1 Twitter

To begin the Twitter data gathering for a stock of choice, various methods are available. The most obvious of these is the Twitter API, the official entry point for programmatically obtaining any data from Twitter. Unfortunately, the API comes at a considerable price. For access to tweets older than 30 days immediately requires the *Premium* tier at a minimum, which comes in at $99 for 100 requests / month as the cheapest option. Clearly, more than 100 requests are going to be required to obtain all of the necessary data, so this API was not going to be accessible. An alternative, Twint [40], self-describes as “an advanced Twitter scraping tool written in Python that allows for scraping Tweets from Twitter profiles without using Twitter’s API”. As the functionality provided by Twint more than covers what is required for the project, this is the option that has been chosen.

Using the Twint API, tweets containing a reference to the stock of choice are collected for every day in the specified time range. Tweets are limited to English language only, to simplify the later text processing stage. Each day of tweets is saved to a CSV file, containing all information about the tweet, such as its text content, date and time posted, author, number of likes and retweets, number of replies, etc..

On parallelisation

The collection of the tweets is the stage of the model that takes the longest amount of time to run (assuming all stages were run linearly, i.e. without concurrency). This is in part due to the volume of data being collected, but in fact more due to timeout restrictions. While no exact figure exists in Twint’s documentation, Twitter imposes a cooldown period after so many requests are performed in a certain timespan. This means that the tweet collection algorithm includes a sleep command between each day, to avoid hitting the limit. A typical run collecting two years’ worth of tweets about Ryanair took over 24 hours running linearly and with a 100s sleep between days.

By utilising the Imperial College Computing Support Group’s setup of Condor [41], several steps across all stages of the pipeline are able to be parallelised, including the data collection. Condor allows for jobs to be submitted and distributed across almost 400 lab machines. In the particular case of the tweet collection, the number of simultaneous jobs was limited to 10, to avoid concerns about Twitter potentially misinterpreting the requests as some sort of DDOS attack.

4.2.2 Reuters

While it was possible to poll Twitter for data using pre-existing APIs, no such service existed for Reuters. This necessitated the construction of a manual scraper, that would search the UK edition of Reuters for all news articles containing the company of interest and pull the article content. Accepting a stock name and a start and end date, the scraper obtains the URLs of all of the news articles from Reuters' own search results pages; it then accesses each of the news articles one by one, strips out the relevant sections using preexisting knowledge of the HTML layout of a standard news article, and returns just the headline, body text and publish date of the article.

While this may sound like a long-winded process, the requesting of pure HTML content is a relatively low-intensity operation; as no browser is involved, the webpage does not have to be rendered at any point, but rather just parsed to extract the relevant details. As a result, the collation of Reuters articles does not therefore require any parallelisation.

4.2.3 Historical prices

It is also necessary to obtain the historical stock prices for the same time periods as the text data. Daily resolution historical stock data, including open, close, high, low, and volume metrics, can be obtained for virtually all publicly-listed stocks from Yahoo Finance [42], with additional data for U.S. stocks available from the Center for Research in Security Prices (CRSP) [43].

4.2.4 Google Trends

In early iterations of the project, a Google Trends scraper was also constructed. Google Trends provides historical data for any search term of choice, with a variety of formats and filters available. The intended use case for this was to be able to see how search terms relating to the stock of choice changed in popularity over time. If a stock suddenly started receiving more hits on Google, this could well be an indication of something having occurred in the news. Ultimately, this scraper was not included, for two reasons:

1. Of all the data sources included, Google Trends would by far have been the most "reactive". Rather than preempting news events, surges in search interest for a topic almost always come *after* the topic has gained some mainstream attention. If the efficient market hypothesis is to be believed (see 2.5.3), any edge that could be gained from observing a spike in search interest will almost certainly have already been priced in by the time a prediction is made.

2. The output of the Google Trends scraper is effectively a 2D matrix of datetimes and their corresponding relative search interest, available at hourly resolution. This data must somehow be transformed to be compatible with the rest of the inputs, which are all text-based. There is no obvious way to do this. The frequencies of the text inputs could be multiplied by some scaling factor relating to the stock's search popularity, but if there is, as believed, a lag in news being released and the search popularity increasing accordingly, would the wrong text not end up getting multiplied by the wrong scaling factor?

It would be interesting to find a way to incorporate the Google Trends information in the future, but for now this data source has been disregarded.

4.3 Stage 2: Data processing

With the data obtained, a considerable amount of processing has to be carried out before it can then be used for any modelling.

A number of core issues exist with the text data in its raw form.

1. The volume of tweets varies on a daily basis. At a baseline level, Twitter sees less engagement at weekends [44] than during the week, but weekend tweets are still just as relevant, as they are used to predict stock movements on the following Monday. Major fluctuations in daily tweet numbers throughout the data set are problematic if not dealt with. For example, on a day with a particularly large number of tweets, a rudimentary model may detect a large uptick in mentions of specific words that are actually not mentioned any more than usual as a *proportion* of the overall data set.
2. The veracity of tweets varies hugely between accounts posting them. As raised in 3.3, a debate exists as to how a potential weighting mechanism might work to assign a higher influence score to tweets from accounts with greater levels of engagement. However, this in itself is not at all as straightforward as it may seem. Does a correlation exist at all between accounts with greater numbers of followers, and some sort of superior knowledge of the topic being discussed in their tweets? If yes, how can this correlation be explored and quantified in such a way that could then be applied to this model? Verified accounts on Twitter are often organisations, or individuals connected therewith, and any tweets they put out would potentially be relating to news articles of interest. However, rumours often swirl online from unverified sources *before* the news article comes to print, and so would there be a tangible benefit in assigning a higher weighting to an account that is not actually the first to report news?
3. The volume of tweets far outweighs the volume of text from Reuters news articles.

4. The data arrives in a multitude of timezone formats:

    - Tweets and news articles are stamped with the date and time of the timezone of the machine running the data collection script,
    - Price data is recorded according to the timezone of the exchange, i.e. 13:00 in the data for a stock listed on NASDAQ represents 13:00 Eastern time,
    - For half of the year, many (but not all) countries observe daylight savings time. A tweet relating to a UK-based stock, written at midday on a day in January and collected on a UK-based machine in June, will be correctly marked as being in the `Europe/London` timezone [45], but will be recorded as 13:00 rather than midday. This would lead to tweets and news articles being assigned to the wrong day, producing unreliable output from the model.

These issues are perhaps best answered in reverse order. Regarding 4), the solution chosen is to store all data across the project - tweets, news articles and price data - using UTC. For each data type, this requires an extra layer of processing at the scraper stage. For tweets and news articles, it becomes necessary to collect an extra day of tweets either side of the desired time range, converting all to UTC, and then selecting those that fall within the desired UTC range. Daily price data for the majority of stocks is immune from this issue, apart from stocks listed on exchanges in Australia and the Far East, where their exchanges' opening times - say, 10:00 local time - would translate to late at night on the previous day in UTC. A quick check is therefore required to ascertain whether the UTC date of an exchange's opening time is not the same date as the local time, and if so, to subtract one from all dates on the price history.

The Python module `pytz` provides direct support for daylight savings time, by means of removing the hour offset automatically if required.

To deal with 3), a straightforward linear weighting is applied to the raw text frequency matrix of the tweets, one that scales down the word frequencies to such an extent that the most frequent word on a given day of tweets is somewhat comparable with the most frequent word on the same day of news content. A few scaling factors were tried, with $\frac{1}{100}$ proving to be the most effective. For stocks where there is a considerable disparity in terms of the ratio of tweets to news articles, a different scaling factor may ultimately produce better results, but stocks are specifically chosen for being "customer-facing" companies, and so the ratio should remain relatively similar across all.

Regarding 2), the ultimate solution adopted was to not weight tweets at all. Referring back to one of the core aims of the project - to use the public mood as the key basis for making predictions - it was deemed unnecessary to attempt to devise a way of giving preference to tweets from certain users. Furthermore, after the text frequencies had been scaled down, there was little tangible benefit to be gained from then weighting the frequencies further.

The solution to 1) involves a considerable amount of preprocessing, followed by a two-step process of normalisation and standardisation. These steps are applied to both the tweets and, where relevant, the news articles.

4.3.1 Preprocessing

Tweets regularly contain emojis, URLs, @mentions, #hashtags, and, less frequently, $cashtags. These are part and parcel of the core Twitter social platform, but serve no purpose for this model. As such, these are stripped from every tweet using a series of regex matchers.

The next preprocessing step is to perform lemmatisation across the entire remaining text set. Lemmatisation is the process of grouping together all inflected forms of a word by its lemma, or root form. As a trivial example, the lemmatisation of the sentence “Today I bought tickets for the cinema” would result in “Today I buy ticket for the cinema”. This means that the final word frequency calculated for the noun “ticket” would also include all mentions of the plural form “tickets”; the final word frequency of the verb “buy” would include all mentions of all conjugations across all tenses, and so on.

After this, all stop words are removed from the lemmatised text set. Stop words are very common words that provide no additional information to the text, such as “the”, “this”, “a”, etc.. The list of stop words is obtained from SpaCy’s `spacy.lang.en.stop_words.STOP_WORDS`. The remaining words are then tokenised; for example, the sentence “I play golf” becomes the array `[‘I’, ‘play’, ‘golf’]`.

The final step of preprocessing is to convert the lemmatised, tokenised daily tweets into daily word frequency matrices. All of the words remaining from the previous steps are fed into a counter that outputs a list of all words, ordered by their frequency across the entire text corpus.

Figure 4.2 displays the most frequent 1,000 words over a one-year period for tweets containing the word “Ryanair”, plotted against each word’s frequency. As can clearly be seen, the top 10 or so words dominate in terms of word frequency, while there is a relatively very small difference in frequency between, say, the 250th most common word and the 1,000th. This suggests that only a relatively small subset of all the words returned from the lemmatisation are actually likely to have any discernible correlation with the stock price. This cutoff, m, is arbitrarily set at 1,000, but could potentially be lowered (for example, the 750th most common word, ‘elderly’, has a weighted frequency of 16.21, while the 1,000th word, ‘fault’, has 11.34; neither of these words are likely to have any realistic correlation with the price).

For each day s in the text corpus, an $m \times 1$ matrix y is constructed, where y_i is the $(i, 0)$th value and represents the frequency of the ith ranked word on that particular day. For example, if the 5th most common word across the entire lemmatised text set

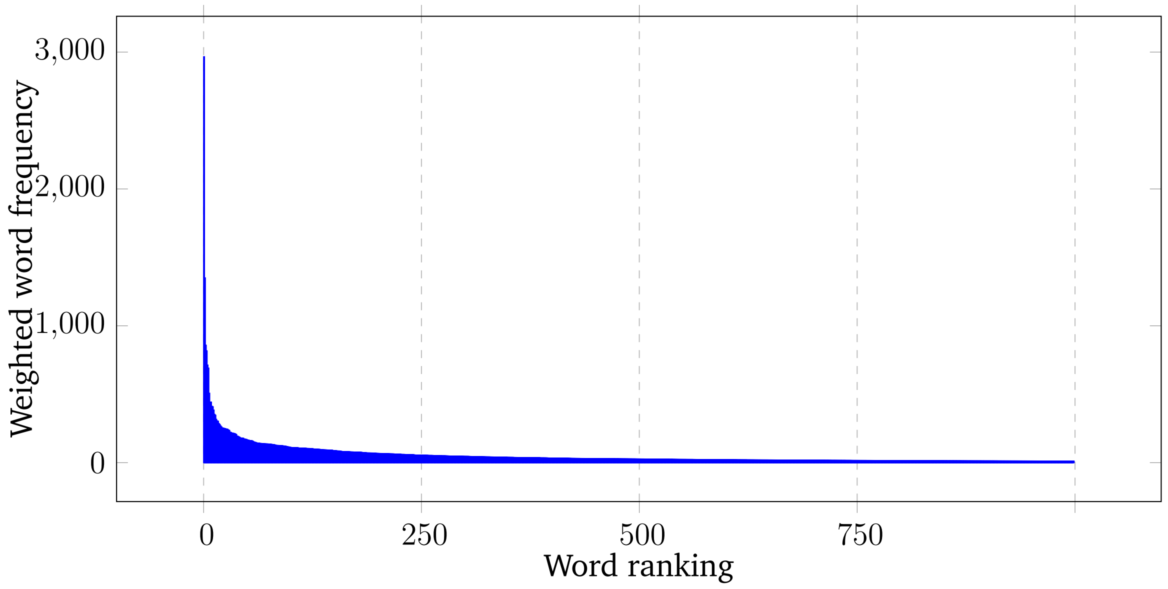


Figure 4.2: Word frequency vs. word popularity for the most frequent 1,000 words from tweets about Ryanair, 01/01/2018 - 01/01/2019

is ‘pilot’, then y_5 represents the frequency of ‘pilot” on that day. These matrices are then combined into a larger matrix $Y \in \mathbb{R}^{m \times s}$, where Y_{ij} is the frequency of word i on day j.

4.3.2 Normalisation

The next step is to normalise matrix Y, to counter against the variations in specific word frequency caused by the particular day having more or fewer tweets than usual. If a day of term frequencies is y, then the normalised frequencies for that day are $\frac{y}{||y||_2}$, i.e. the vector of frequencies divided by its L2 norm. This step is carried out for every day i in Y.

4.3.3 Standardisation

The final step is to standardise the term frequencies using their z-scores. This produces a matrix of how many standard deviations above or below the mean a given word appears on a given day, rather than simply its normalised frequency. The z-score for word i on day j is given by

$$z_{ij} = \frac{Y_{ij} - \bar{\mu}_i}{\bar{\sigma}_i}$$

where $\bar{\mu_i}$ and $\bar{\sigma_i}$ are the mean and standard deviation of the frequencies of word i calculated for all days $[0..j]$.

All values below 0 in this matrix are then pruned to 0. Naturally, a word that occurs less than average on a given day is unlikely to carry any significance when trying to predict the stock movement. Furthermore, to reduce the effect of outliers, any value greater than 3 (i.e. a word that has a frequency on a given day of more than 3 standard deviations over the mean) is capped at 3.

The contents of the final matrix Y will henceforth be referred to as the **z-reduced text content** for day i.

4.3.4 Simplified example

A demo corpus C of five tweets collected across a three day period after the 2019 UEFA Champions League final is found in figure 4.3.

$$C = \begin{cases} \textbf{Day 1} & \text{After two of the best semi-finals in recent memory, the \#UCL final} \\ & \text{last night was probably the biggest letdown of the entire season!} \\ \textbf{Day 1} & \text{@SkySportsNews spurs were awful, waste of space in the final,} \\ & \text{deserved win for liverpool} \\ \textbf{Day 2} & \text{shocker from the referee to award the early penalty so early on.} \\ & \text{killed the game for a very blunt spurs} \\ \textbf{Day 2} & \text{if Lucas started for Spurs you feel it would've been a different} \\ & \text{game...} \\ \textbf{Day 3} & \text{Still can't get over how bad @BTSport's live stream was for the} \\ & \text{\#UCL final!! Shambolic} \end{cases}$$

Figure 4.3: Example corpus C of tweets

Preprocessing

Removing the aforementioned Twitter features such as @mentions and #hashtags, lemmatising all words, removing stop words and then returning the remaining words as tokens yields the preprocessed corpus C':

Taking the top $m \leq |\{C'\}|$ (for this example, $m = 5$) most common words in C' gives the list in table 4.1,

$$C' = \begin{cases} \textbf{Day 1} & \text{[‘good’, ‘semifinal’, ‘recent’, ‘memory’, ‘final’, ‘night’, ‘probably’,} \\ & \text{‘big’, ‘letdown’, ‘entire’, ‘season’]} \\ \textbf{Day 1} & \text{[‘spur’, ‘awful’, ‘waste’, ‘space’, ‘final’, ‘deserved’, ‘win’, ‘liverpool’]} \\ \textbf{Day 2} & \text{[‘shocker’, ‘referee’, ‘award’, ‘early’, ‘penalty’, ‘early’, ‘kill’, ‘game’,} \\ & \text{‘blunt’, ‘spur’]} \\ \textbf{Day 2} & \text{[‘lucas’, ‘start’, ‘spur’, ‘feel’, ‘different’, ‘game’]} \\ \textbf{Day 3} & \text{[‘bad’, ‘s’, ‘live’, ‘stream’, ‘final’, ‘shambolic’]} \end{cases}$$

Figure 4.4: Preprocessed corpus C' of tweets

Rank	**Word**	**Frequency**
1	final	3
2	spur	3
3	early	2
4	game	2
5	good	1

Table 4.1: Top $m = 5$ most common words in C', with each word's frequency in the corpus

Construct the matrix $Y \in \mathbb{R}^{m \times s}$, representing the number of times each word m appears on day s.

$$Y = \begin{pmatrix} 2 & 0 & 1 \\ 1 & 2 & 0 \\ 0 & 2 & 0 \\ 0 & 2 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

Each column y_i is then divided by its L2 norm, to normalise the word frequencies.

$$Y = \begin{pmatrix} 0.8165 & 0 & 1 \\ 0.4082 & 0.5774 & 0 \\ 0 & 0.5774 & 0 \\ 0 & 0.5774 & 0 \\ 0.4082 & 0 & 0 \end{pmatrix}$$

The z-scores are calculated using the method described above. Each value of the first column is also trivially the mean for that row up to that point, and so the standard deviation is 0, which leads to an incalculable z-score due to a divide-by-0 error. To avoid this, the first column's z-scores are defaulted to 0.

$$Y = \begin{pmatrix} 0 & -1 & 0.9075 \\ 0 & 1 & -1.3552 \\ 0 & 1 & -0.7072 \\ 0 & 1 & -0.7072 \\ 0 & -1 & -0.7074 \end{pmatrix}$$

Finally, the lower and upper bounds on all values are capped at 0 and 3, respectively:

$$Y = \begin{pmatrix} 0 & 0 & 0.9075 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

With such a small initial dataset and m value, the example Y matrix produced here has few features of legitimate interest, but two observations that could be made are:

- The words ‘spur’, ‘early’ and ‘game’ appear an above average number of times on day 2, and ‘final’ on day 3
- The word ‘good’ *either* does not appear at all in the three days, *or* appears exclusively on the first day. Due to the z-score being incalculable for the first day, this is one of the limitations of the method.

4.4 Stage 3: Minimisation

4.4.1 Introduction

The basis for this model is the assumption that the price of a stock on any given day can be represented as a function of the stock’s characteristics, and the public mood towards the stock at that point in time. The mathematical derivation of this function follows loosely from the work in [9].

In [9], a model is built to predict the movement of a large number n of stocks simultaneously, by combining all prices across all days s into a single matrix $R \in \mathbb{R}^{n \times s}$. This requires all stocks sharing a single d-dimensional latent factor space - a mathematical description of the latent features of each stock. Instead, here a model is created for a single stock, doing away with the need for a multidimensional latent factor space altogether, as the stock no longer needs to be described using generalised features applicable to all other stocks at the same time. Thus, the stock of choice, i, can be represented as a singular-value feature vector $u \in \mathbb{R}^{1 \times 1}$. The public mood towards the stock can also be represented as a vector in the same (single-value) feature space, $v \in \mathbb{R}^{1\times 1}$. According to the assumption laid out at the start of this section, the price of the stock i on day t can therefore be obtained from the product of these two vectors, i.e.

$$r_{it} = u_i^T v_t, u_i \geq 0 \tag{4.1}$$

u is forced to be positive, as the public mood represented by v can be either positive or negative, and the resulting vector should reflect this sentiment. The public mood towards the stock i on day t, for this model, is represented by the tth column of the z-reduced text content vector Y, y_t, obtained in 4.3.1. As vectors v_t and y_t are of different shapes - v_t is of size (1×1) while y_t is $(m \times 1)$ - there must also exist a linear transformation $W \in \mathbb{R}^{1\times m}$ such that $v_t = Wy_t$. This ultimately gives

$$r_{it} = u_i^T W y_t, u_i \geq 0 \tag{4.2}$$

Given that the historical log returns and the text data are both known, this becomes a matter of solving to find the matrices u_i^T and W. More specifically, $R \in \mathbb{R}^{1\times s}$ represents the log returns for stock i for all days s, and Y represents the text data as previously seen; thus,

$$R = UWY, U \geq 0 \tag{4.3}$$

where $U = u_i^T$. The vectors U and W thus can be learned by means of an objective function,

$$\text{minimise} \quad \|R - UWY\|^2, U \geq 0 \tag{4.4}$$

and solving for U and W. Once U and W have been identified, a prediction for the return of stock i on day t can be made via

$$r_{it} = UWy_t \tag{4.5}$$

4.4.2 Reducing overfitting

For this model, a value of $m = 1000$ has been selected as the number of terms to calculate frequencies for per day. The rationale behind this number is touched on in 4.3.1, but generally, m should fall within a range such that it allows a sufficient variety of words to be analysed, while also not being too large so as to result in outlier words ending up being associated with a correlation that does not actually exist (for example, on the same day a stock α rises 5%, a news article summarising the results of various unrelated companies may also be published that includes multiple references to an entirely unrelated stock β; this is the only time in the entire dataset that the stock β gets mentioned, and so the model potentially builds an assocation between mentions of β and a price rise for α).

Nonetheless, even at $m = 1000$, overfitting remains a potential issue; this still means 1,000 parameters are being estimated in order to determine the price movement. Intuitively, while the top 1,000 words are being analysed each day, only a small subset of these are actually likely to have any correlation with the price - for example, in the Ryanair data analysed in figure 4.2, one might imagine “strike” would be associated with a downwards movement, whereas “oleary” - the surname of Ryanair chief executive Michael O’Leary - is probably less likely to be associated with any definitive movement direction. Of course, it is not the job of this paper to predict at this stage which words will or will not have any correlation with the price movement, but merely to restrict the number of words that the model uses to base its predictions upon in order to prevent overfitting. This is achieved by introducing a sparseness constraint to matrix W, to limit the number of nonzero columns. [9] achieves this via the use of a sparse group lasso regularisation term [46]. This is simply an amendment to the minimisation function that adds a penalty based on the sparseness of the columns of W; the more sparse the columns, the smaller their L2 norms, and thus the smaller the penalty, i.e.

$$\lambda \sum_{t=1}^{m} \|W_t\|_2$$

where λ is a hyperparameter to be tested with different values at training time. A second penalty can then be added using the L1 norm of the entire vector W, that penalises candidate W vectors that have a small number of columns with particularly high values, i.e. to reduce the effect of outliers. As outliers are partially removed in the text preprocessing stage, the value of the associated hyperparameter μ for this penalty is typically considerably smaller than that of λ.

$$\mu \|W\|_1$$

The final minimisation problem is thus

$$\text{minimise} \quad \|R - UWY\|^2 + \lambda \sum_{t=1}^{m} \|W_t\|_2 + \mu\|W\|_1, U \geq 0 \tag{4.6}$$

4.4.3 Solving for U and W with ADMM

The first step towards solving for U and W is to incorporate the bounding of U, i.e. $U \geq 0$, directly into the objective function. This can be achieved via the introduction of an indicator function $I_+(U)$, that returns 0 if U is positive or ∞ if U is negative.

$$\text{minimise} \quad \|R - UWY\|^2 + \lambda \sum_{t=1}^{m} \|W_t\|_2 + \mu\|W\|_1 + I_+(U) \tag{4.7}$$

This problem can be solved via an augmented Lagrangian method (see 2.4.2). In [9], it is shown that ADMM can be applied to the minimisation function encountered here. Firstly, the auxiliary variables $A = U$ and $B = W$ are introduced:

$$\text{minimise} \quad \|R - ABY\|^2 + \lambda \sum_{t=1}^{m} \|W_t\|_2 + \mu\|W\|_1 + I_+(U) \tag{4.8}$$

$$\text{subject to } A = U, B = W$$

Next, the augmented Lagrangian L_A for this objective function is formed with the introduction of the Lagrangian multipliers C and D:

$$\begin{aligned} L_\rho(A,\, B,\, U,\, W,\, C,\, D) = \|R - ABY\|^2 &+ \lambda \sum_{j=1}^{m} \|W_j\|_2 + \mu\|W\|_1 + I_+(U) \\ &+ \text{tr}(C^T(A - U)) + \text{tr}(D^T(B - W)) \\ &+ \frac{\rho}{2}\|A - U\|_F^2 + \frac{\rho}{2}\|B - W\|_F^2 \end{aligned} \tag{4.9}$$

A more in-depth derivation, and subsequent solving, of this augmented Lagrangian is detailed in Appendix A. The end result is the two matrices U and W, which can be used to calculate a prediction as per equation (4.5).

4.5 Evaluation

Given sufficient historical text and price data, the computation of the U and W matrices now allows a prediction of the stock returns for a day of choice.

4.5.1 Stock selection

The stocks chosen to be analysed have to fit three broad criteria:

- They must have a uniquely identifiable name that, when searched on Twitter, would result in most tweets returned being about this company and not some other entity or word altogether. Examples of good stocks that fit this point include Ryanair [LON:RYA] and easyJet [LON:EZJ], while bad stocks might include United Airlines [NASDAQ:UAL] (most people would tweet simply about "United", which would end up being conflated with various football clubs) or Zoom [NASDAQ:ZM] (a reasonably common verb).
- They must be i) reasonably-sized and ii) "customer-facing" companies; both of these points are linked to the volume of tweets and news articles produced. A small business, or a larger business operating exclusively in a restricted region, is unlikely to be garnering as many comments on social media, and even fewer articles on Reuters; likewise, even a large company involved in the supply of specialist hardware components to manufacturing businesses may get a few mentions from the mainstream news but is unlikely to be at the forefront of the general public's concerns. Greggs [LON:GRG] and Vodafone [LON:VOD] are two further examples of good stocks that meet this point; Severn Trent [LON:SVT] and Standard Chartered [LON:STAN] are two examples of stocks that do not.
- Lastly, their associated text content must be predominantly written in English. The NLP kits that are used in the data preprocessing stage for lemmatisation, tokenisation and stop word removal are built on English language libraries, and so any text content in a different language will fail to be processed correctly.

From this, the list of stocks chosen are listed in table 4.2, along with their sector (only if there are two or more stocks from the same sector; otherwise, they are listed as 'Other').

Stock	Ticker	Sector
Asos	LON:ASC	Retail
AT&T	NYSE:T	Telecommunications
Barclays	LON:BARC	Banking
Burberry	LON:BRBY	Other
Cineworld	LON:CINE	Other
Credit Suisse	SWX:CSGN	Banking
easyJet	LON:EZJ	Aviation
Ethereum	ETH-USD	Other
Greggs	LON:GRG	Retail
jetBlue	NASDAQ:JBLU	Aviation
Ocado	LON:OCDO	Retail
Ryanair	LON:RYA	Aviation
Sainsbury's	LON:SBRY	Retail
Stagecoach	LON:SGC	Other
Tesco	LON:TSCO	Retail
Vodafone	LON:VOD	Telecommunications

Table 4.2: Stocks selected for analysis

4.5.2 Simulating a prediction

For each stock, the text and price data are collected over a 1,127 day period between 1 April 2017 and 1 May 2020. There are roughly 777 trading days in this period for which a price can be predicted ('roughly' because this number varies by country, due to differing numbers of bank (national) holidays). To try to use as much data as possible for calculating the U and W matrices while also leaving sufficient data to test the model, the number of days of historical data used, s, is set to 400. The evaluation procedure advances as follows:

1. The text and price data for some stock i of days 1-400 are fed into the model to produce the matrices U and W. A prediction $r_{i,401}$ is then made for day 401, via $r_{i,401} = UWy_{401}$, where y_{401} is the text content starting immediately after the close of the market on day 400 up until midday of day 401.

2. The direction of the prediction (i.e. whether it is positive or negative) is recorded and compared against the actual direction for day 401. Three potential outcomes are possible; the model correctly predicted the direction, the model incorrectly predicted the direction, or the model produced a zero prediction. Zero predictions can occur either due to a lack of any data on day 401 that matches previously-identified trend indicators, or due to the choice of hyperparameters being overly penalising leading to no prediction being made. The outcome is recorded in a running tally.

3. Steps 1-2 are then repeated but for days 2-401, with a prediction calculated for day 402, and so on until all days have been exhausted (i.e. after predicting day 777). The result is a tally of predictions for 376 days. The total number of correct, incorrect and zero predictions are returned. Once $\geq$50 days have been simulated, if at any point the number of correct guesses falls below 10%, the job is terminated, to reduce time spent calculating predictions for models with very poor performances.

There of course also remains the matter of the three hyperparameters λ, μ and ρ. After some initial trial runs to find upper and lower bounds on hyperparameters to test, five different values for each hyperparameter are considered, leading to a total of 125 combinations. These are as follows:

$$\begin{aligned}\lambda &= [0.4, 0.5, 0.55, 0.6, 0.65]\\ \mu &= [0.2, 0.25, 0.3, 0.35, 0.4]\\ \rho &= [3, 4, 5, 7, 10]\end{aligned}$$

Making use of the Department of Computing's Condor setup (see section 4.2.1), steps 1-3 above are able to be run as 125 simultaneous jobs, each with a different combination of hyperparameters, vastly decreasing the time taken per overall run.

4.5.3 Performance evaluation: directional prediction accuracy

There are two ways to compare stocks' performances in the model - their directional prediction accuracy, and their simulated trading performance. The directional prediction accuracy is concerned with how well the model is able to predict the movement of the stock price for a given day, while the simulated trading performance provides an illustration of how profitable these predictions are (or are not). While the two are linked, one is certainly not indicative of the other - it is perfectly plausible that a model could arrive at a profit at the end of the simulated trading having actually had more lossmaking days than profitable days, and vice-versa.

At the end of the training pipeline, 125 results are produced for each stock, representing the performance of the model under each of the separate hyperparameter combinations. The means of determining which of the sets of hyperparameters has produced the model that is "best" comes down to how zero predictions are interpreted.

In one scenario, a zero prediction is considered an incorrect guess. This lends itself towards choosing models where the number of zero predictions, res_{zero}, is minimised. Using this approach, the *best* model is thus defined simply as the model with the highest proportion of correct guesses $res_{correct}$ out of the three categories; i.e. the models are ordered according to the sorting function σ,

$$\sigma = \frac{res_{correct}}{res_{correct} + res_{incorrect} + res_{zero}} \tag{4.10}$$

In a second scenario, the *best* model is instead described as the one with the fewest incorrect predictions $res_{incorrect}$. Zero predictions are not necessarily desirable, but also do not represent an incorrect decision, and so are just ignored. The sorting function σ is thus

$$\sigma = \frac{res_{correct}}{res_{correct} + res_{incorrect}} \tag{4.11}$$

The primary issue with this sorting function is that it can (and does) lead to models ranking top with 70% or 80% (or even higher) of their guesses coming out as zeroes. As the sample size of nonzero guesses is thus much reduced, it only takes a small run of luck for a model to come out on top - luck which would far more likely be evened out over the course of more predictions. A restriction $res_{correct} \geq c$ could be placed upon the output of this filter to ensure only models with a number of correct predictions greater than some cutoff c are included, but before too long the results of this sorting function tend towards the same results obtained from (4.10). For this reason, this second scenario is discarded, as it does not provide a meaningfully different way of assessing the stocks.

The first 100 days of the test data are used as the validation set from which to select the optimal hyperparameters. When overlaying these predictions onto the stock price to calculate the trading performances, these 100 days are included. The justification behind this is twofold;

- Firstly, the directional prediction accuracy and the trading performance are related, but not as strongly as may be assumed. Due to the fact that the model uses the previous day's close price as its assumed *current* price when making a prediction (see 4.5.4) as hourly data is not freely available going back to the start of the training period, any intraday movements between the open of the market and the lunchtime prediction will have a knock-on effect on the result of the prediction itself. It is perfectly possible to have a model that makes more correct predictions than incorrect, and yet ends up with a sizeable loss at the end of the simulation period.
- Furthermore, the test data is already relatively sparse, and the unexpected arrival of the coronavirus crisis and its impact on stock markets means that the final two and a half months of trading data must be treated separately anyway.

> The extra 100 days of predictions are deemed necessary to provide a larger experiment size, if perhaps at the expense of a small level of reliability in the results.

The set of hyperparameters selected by (4.10) will be referred to as the M^+ model.

4.5.4 Performance evaluation: simulated trading

With the directional predictions obtained, trading can now be simulated to get a direct appraisal of a model's market performance. There are two strategies for trading simulation that will be used here.

Strategy one: literature methodology

For each of the stocks, the directional predictions are laid over the historical prices by the trading simulator. The simulator is loaded with £10,000 of virtual money. At midday of each day, the then-current price is obtained from Yahoo Finance. If the directional prediction for that day is then upwards, a buy order (*going long*) is simulated using all of the available capital. At the end of the trading day, all shares are then sold back. The new balance is recorded, and then this process is repeated for every trading day. By plotting the balance over time, it is possible to observe how the model performs for different stocks, and then also compare its performance against those of three index funds: FTSE 100, the Dow Jones Industrial Average (DJI), and the S&P 500. The hourly data required for these trades is obtained via the `yfinance` module for Python, allowing for hourly data for up to the previous 730 days to be obtained. Once all stocks have passed through the model, the overall performance is obtained by simply averaging over all results.

The strategy presented above is borrowed directly from [26], as it allows for a direct comparison with the literature. However, it only makes use of half of the data - days where an increase is predicted - and so there is scope for improvement. Shorting, as introduced in 2.5.1, presents an opportunity to make use of both increasing and decreasing predictions. For this reason, the results of this evaluation method are presented in two separate graphs, one with and one without shorting.

Strategy two: an aggregated model

Not all stocks are likely to perform equally well in the model, and so a more nuanced approach that disregards those that are underperforming could well lead to an improvement on the trading strategy presented in [26].

Before the start of each week, every stock model has its recent performance assessed over three time periods: one week, one month and three months. Each performance is weighted for recency, with a $\pi : \tau : \upsilon$ ratio accordingly. The weighting formula ω_j for stock j is thus

$$\omega_j = \pi \frac{v_i}{v_w} + \tau \frac{v_i}{v_m} + \upsilon \frac{v_i}{v_{3m}} \tag{4.12}$$

where v_i is the value of that stock's model on day i, v_w is the value one week prior to day i, v_m the value one month prior and v_{3m} the value three months prior. A greater value of ω_j indicates a stronger short-to-medium-term performance of the model for stock j.

Each stock is then ranked according to its ω value. The top γ stocks are then selected, and the percentage of funds allocated to each of them on each trading day that week, I_j, is relative to the size of that stock's ω value relative to the overall sum, i.e.

$$I_j = \frac{\omega_j}{\sum_{i=1}^{\gamma} \omega_i} \tag{4.13}$$

If stock j produces a zero guess on day i, the allocated funds for that stock are simply not invested on that day. For the earlier periods of the model where the values of v_m and v_{3m} are undefined, these values are set to 0.

The optimal values for π, τ, υ and γ are to be determined via a grid search over the first 200 days, with those values then applied over the full 376 trading day period. The grid search is conducted with the following values:

$$\begin{aligned} \pi &= [0.4, 0.5, 0.6, 0.7, 0.8] \\ \tau &= [0.1, 0.2, 0.3, 0.4, 0.5] \\ \upsilon &= [0.0, 0.1, 0.2] \\ \gamma &= [1, 2, 3] \end{aligned}$$

These values are selected under the belief that a stock's *recent* short-term performance is more likely to be indicative of its *future* short-term performance, and that too large a value for γ would bring the model too much in line with the M^+ average.

4.5.5 Further indicators

Once the trading performances have been evaluated, it then becomes possible to conduct further statistical analysis. Metrics such as the Sharpe ratio and conditional value at risk (CVaR) will be calculated for each trading methodology, as well as the index funds, and compared against similar such values as reported in the literature. The Sharpe ratio is calculated with an assumed risk-free rate of zero, perhaps a slight simplification but also fairly in line with short-term interest rates, and also the same strategy used in [26].

The optimal hyperparameter values for each stock can also be visualised, to try to detect underlying patterns between both stocks and sectors.

4.6 Summary

The idea ultimately is that the model will be able to run autonomously from start to finish, taking a stock as an input and producing a series of predictions for all trading days as its output, via a series of steps as outlined in this section. The model is built to be able to handle stocks traded in all timezones, as long as their associated text content is written in English. The two more computationally-expensive steps of the pipeline - data retrieval and minimisation - are split into 11 and 125 jobs respectively, which are then run in parallel via the Department of Computing's Condor setup. The predictions produced are then run against the stock prices for those days, simulating the buying and selling of the stocks, to assess the model's performance in a stock trading environment.

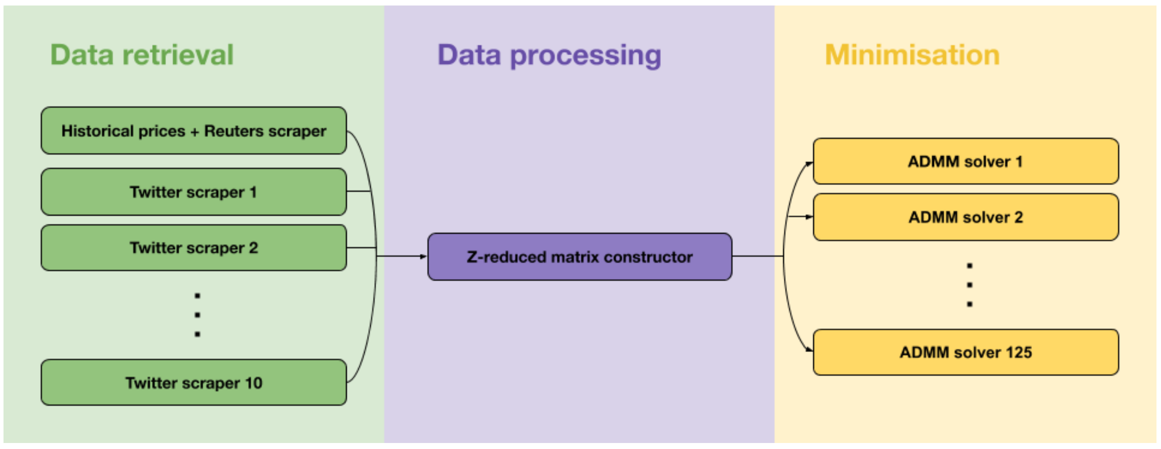


Figure 4.5: Overview of how the pipeline is divided up on Condor

Chapter 5

Results

5.1 Overview

Over the following pages, the evaluation methods detailed in Chapter 4 are now implemented and analysed. The most obvious starting point is the directional prediction accuracy, which is derived directly from averaging the predictions produced from the ADMM solver, and then later these predictions are laid over the stock price to investigate the trading performances. A number of mathematical analyses are also performed, investigating the model's Sharpe ratio and CVaR, before then digging into the W matrices used in the predictions to analyse two individual stocks and explain their performances. Shortcomings of the evaluation methods are also discussed, leaving room for improvement in any future work (see Chapter 6).

5.2 Directional prediction accuracy

Table 5.1 lists all stocks put through the model, along with the number of correct, incorrect and zero guesses produced for each. Chegg recorded no results as all combinations of hyperparameters led to the jobs being terminated before reaching completion (see 4.5.2, step 3).

Overall, the performances are fairly solid, achieving an average correct prediction accuracy of 53.43%, and an incorrect prediction accuracy of only 44.44%. While several reasons potentially exist to explain the failure of the Chegg model, the most likely is the relative paucity of text data collected for it - with only 13 news stories collected from Reuters, and most days averaging under 150 tweets, the model has far less information to work with than the other stocks.

Stock	Correct	Incorrect	Zero	**% Correct**	**% Incorrect**
Tesco	215	146	15	57.18	38.83
Ocado	212	164	0	56.38	43.62
Asos	210	166	0	55.85	44.15
easyJet	208	152	16	55.32	40.53
Sainsbury's	205	171	0	54.52	45.48
AT&T	202	166	5	54.16	44.50
Cineworld	202	170	4	53.72	45.21
Credit Suisse	196	167	3	53.55	45.63
Stagecoach	201	175	0	53.46	46.54
jetBlue	199	174	0	53.35	46.28
Burberry	200	173	3	53.19	46.01
Greggs	198	178	0	52.66	47.34
Ryanair	195	175	6	51.86	46.54
Vodafone	190	145	41	50.53	38.56
Ethereum	366	304	55	50.48	41.93
Barclays	183	186	7	48.67	49.47
Chegg	-	-	-	-	-

Table 5.1: Directional prediction performance of each stock, ordered by correct predictions

This average of 53.43% comes in higher than the ex-sample averages of 51.12%, 51.42% and 51.58% reported in [26], and slightly lower than the 53.9% and 55.7% reported in [9], although both of these were conducted over different time periods. Of course, the directional accuracy alone is no guarantee of turning a profit, so in the next section the trading strategies discussed in Chapter 4 are implemented.

5.3 Trading performance: literature methodology

5.3.1 Analysis

The 377 trading days covered feature two distinct time periods of interest - essentially, the pre- and post-coronavirus portions. From the start of the model testing on the 2nd November 2018 until the 20th February 2020 is considered the pre-coronavirus market, while everything after the 20th February constitutes the post-coronavirus market.

The visualised trading performances of all stocks, the averages, and the index funds are displayed in figures 5.1 and 5.2.

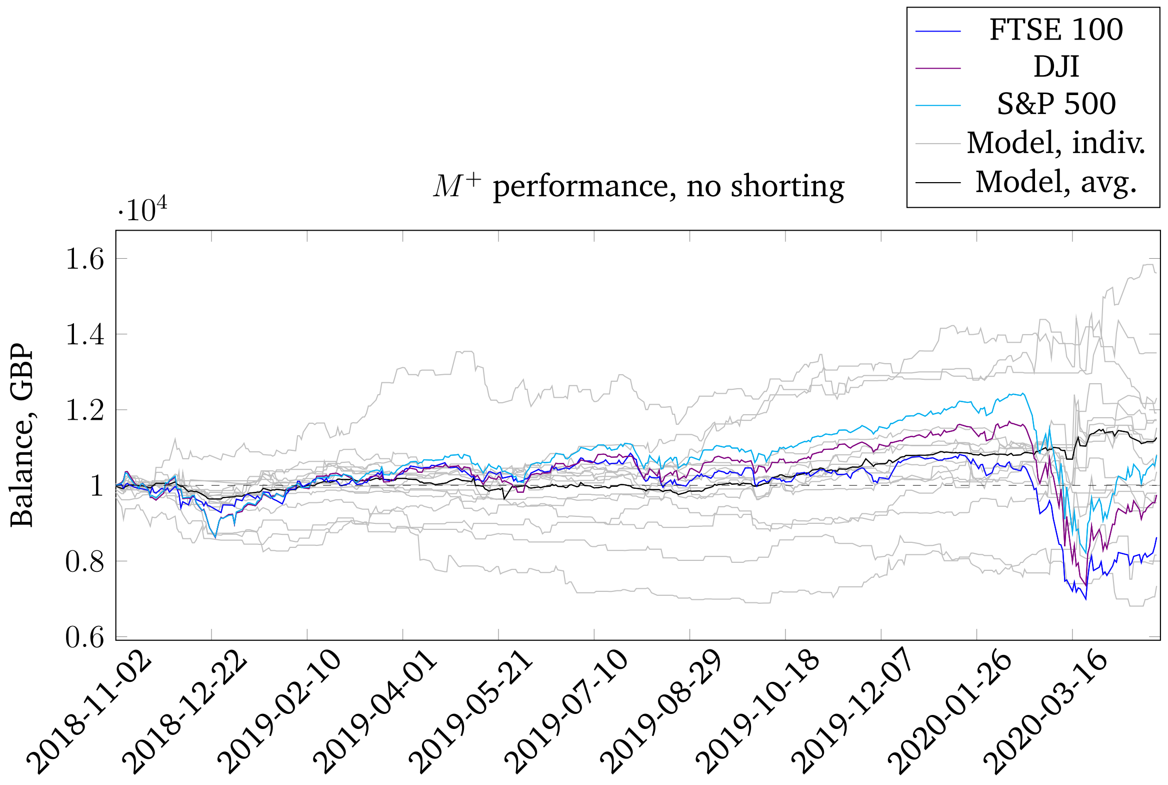


Figure 5.1: Simulated trading performance compared against three baseline indices, with no shorting.

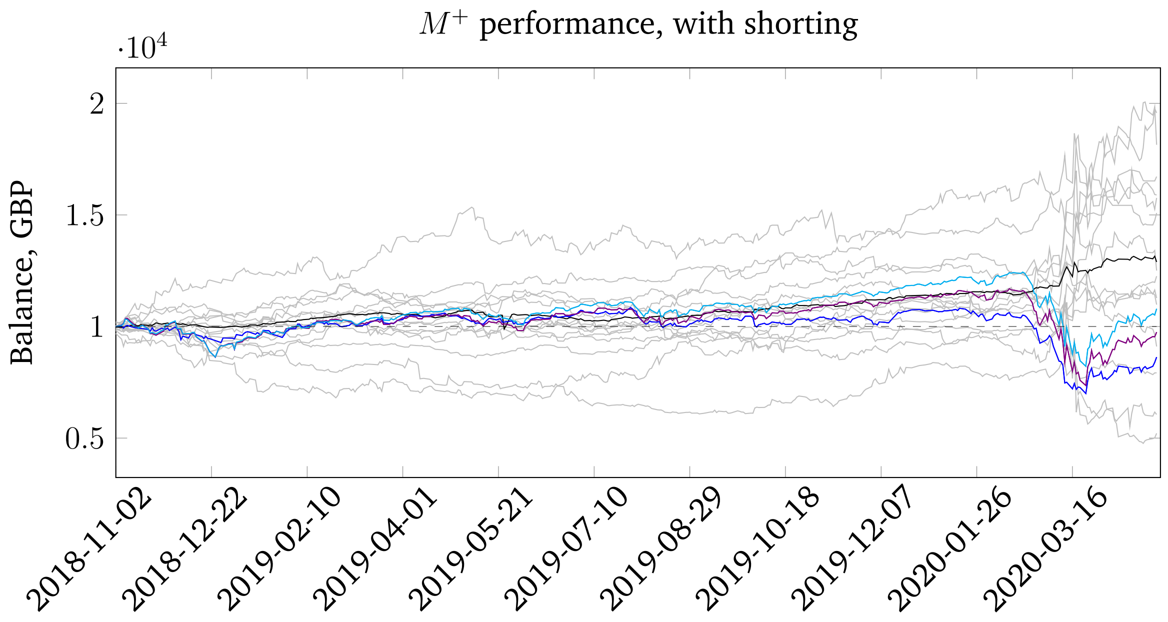


Figure 5.2: Simulated trading performance compared against three baseline indices, with shorting.

Pre-coronavirus

As coronavirus had such a significant impact on stock markets, with many recording their worst daily losses on record in March, it is more useful to first look at how the model performs during a period of relative normality. The pre-coronavirus market consists of largely steady growth for the DJI and S&P 500, with a less prominent performance from the FTSE 100. £10,000 invested into each of the three at the start of the modelling period would, on the 20th February 2020, have been worth £11,562.73 (DJI), £12,310.01 (S&P 500) or £10,495.22 (FTSE 100). These values are converted to annualised returns and presented in table 5.2.

Portfolio	Cum. return	**Ann. return**	Best daily return	Worst daily return
M^+ n.s. avg.	1.087	1.064	1.033	**0.972**
M^+ s. avg	1.151	1.113	**1.065**	0.942
FTSE 100	1.050	1.038	1.026	0.963
DJI	1.156	1.118	1.050	0.969
S&P 500	**1.231**	**1.173**	1.050	0.968

Table 5.2: Comparative performance of the model vs index funds, 02/11/2018 - 20/02/2020. Top performers in each column are highlighted in **bold**

From this table, it can be quickly observed that while the model averages are profitable, their returns are no more noteworthy than those of the standard index funds. Annualised returns of 6.4% without shorting, or 11.3% with, are an improvement on the FTSE 100 over the same time period but are weaker than both of the US-based index funds. As the majority of the stocks selected are based in the UK, the FTSE is probably the most relevant of the three comparisons, but nonetheless these results are nothing particularly worth writing home about.

Pre- and post-coronavirus combined

Next, the final 70 days between the 21st February and 30th April 2020 are also included, in addition to the previous period of time already discussed, to give the full 376-day window. Just by observing figures 5.1 and 5.2, it is clear from the performances of the three index funds that this was an unprecedented period of turbulence in the stock markets. For this reason it is all the more notable how well the model performed during this time. While many index funds suffered sharp falls of over 20% in March, the model coasts through this period, recording few losses of note at all; indeed, the trading strategy involving shorting sees an unprecedented *rise* during this time. Both trading strategies, with and without shorting, are tabulated in table 5.3 and record improved performances when compared to table 5.2, something which cannot be said for the index funds. The apparent resilience of this model to market-wide shocks is one of the key takeaways from the project.

Portfolio	Cum. return	**Ann. return**	Best daily return	Worst daily return
M^+ n.s. avg.	1.126	1.083	1.033	**0.972**
M^+ s. avg	**1.289**	**1.186**	1.065	0.942
FTSE 100	0.863	0.906	1.081	0.904
DJI	0.975	0.983	**1.114**	0.871
S&P 500	1.079	1.052	1.094	0.881

Table 5.3: Comparative performance of the model vs index funds, 02/11/2018 - 01/05/2020

Nonetheless, the unprecedented performance of the model during a historically uncertain time in the market somewhat disguises its lacklustre performance in the period of stability that came before it. By studying figures 5.1 and 5.2, it is apparent that there is a whole range of performances between individual models, which presents an opportunity for improvement with the aggregated model outlined in section 4.5.4. Potential reasons for this variety in model performance are investigated in section 5.7, but firstly the aforementioned aggregated model is tested, with its results following in section 5.4.

5.4 Trading performance: an aggregated model

As outlined in Chapter 4, the aggregated model methodology collates the best-performing stocks over the short-to-medium term via weighting parameters π, τ, υ, and then allocates funds to a subset γ of these on a rolling weekly basis.

5.4.1 Hyperparameter grid search

To recap, parameters π, τ and υ relate to how the weekly, monthly, and tri-monthly performances are weighted in the model evaluation (see 4.5.4), while γ controls the number of stocks selected each week to invest in.

Across the top-performing models on the validation data, two of the hyperparameters remained constant: $\upsilon = 0.0$ and $\gamma = 1$, i.e. the aggregated method works best looking at only the weekly and monthly performances, and selecting the single top-performing stock from the previous week to use for the next week of investing. The top-performing set of parameters was $\pi = 0.7, \tau = 0.3, \upsilon = 0.0, \gamma = 1$.

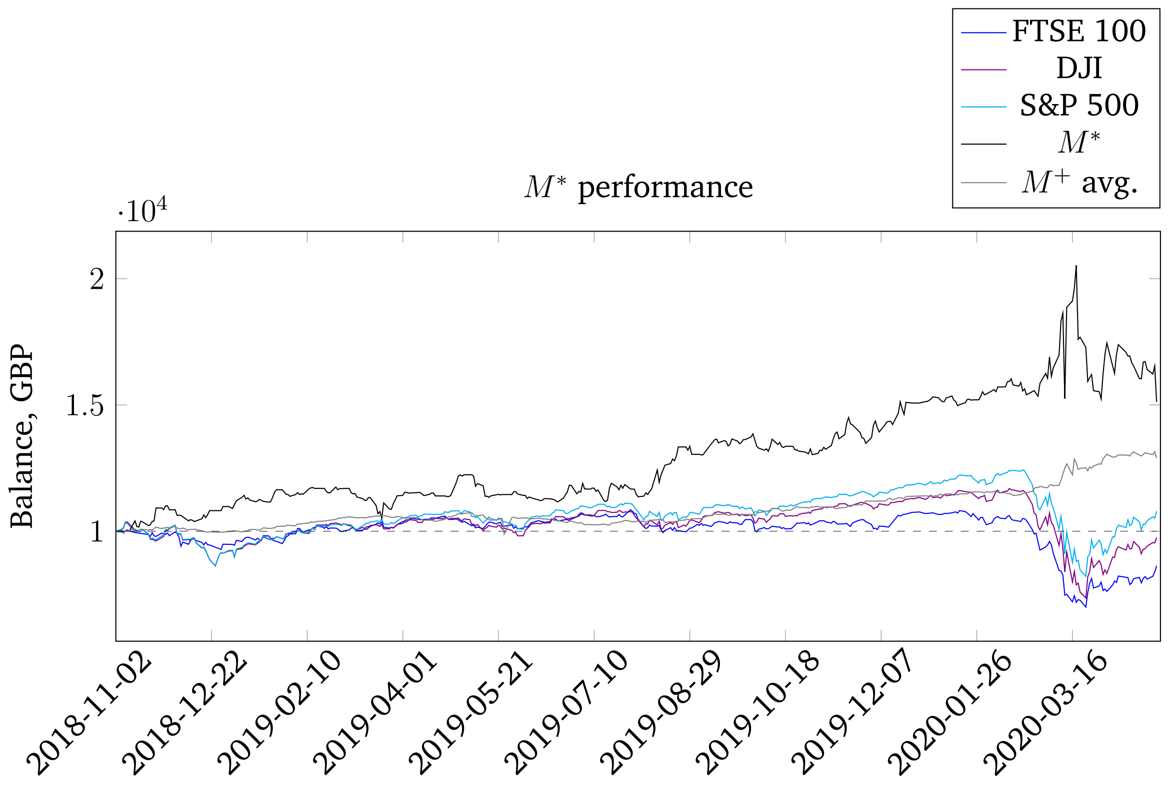


Figure 5.3: Simulated trading performance of the new M^* model compared against three baseline indices and the M^+ model average.

5.4.2 Results

Figure 5.3 shows the graphed performance of the M^* model using the aforementioned hyperparameters, compared against the three index funds and the M^+ model with shorting average. The M^* model's performance is hugely impressive, recording annualised returns of 40.6% during the pre-coronavirus period, dropping slightly to 32.0% for the combined pre- and post-coronavirus period.

The standout observation throughout this chapter has been the robustness of the model's performance during uncertain market conditions, so it is somewhat paradoxical that the M^* model actually achieves a small loss, via a huge amount of volatility, during the latter stages of the simulation. This is likely a corollary of the hyperparameter γ being set to 1. This is, without question, an optimal hyperparameter value for periods of market calm and stability, but in times of uncertainty, it may become preferable to hedge with a higher value of γ, splitting investment funds between multiple stocks. Figure 5.4 looks specifically at the post-coronavirus period, plotting the relative performances of this model against the same model, but with γ set to 3 instead of 1.

While the daily gains made by the hedged ($\gamma = 3$) model tend to be slightly smaller, there is much greater resilience against the market turbulence, leading to a period

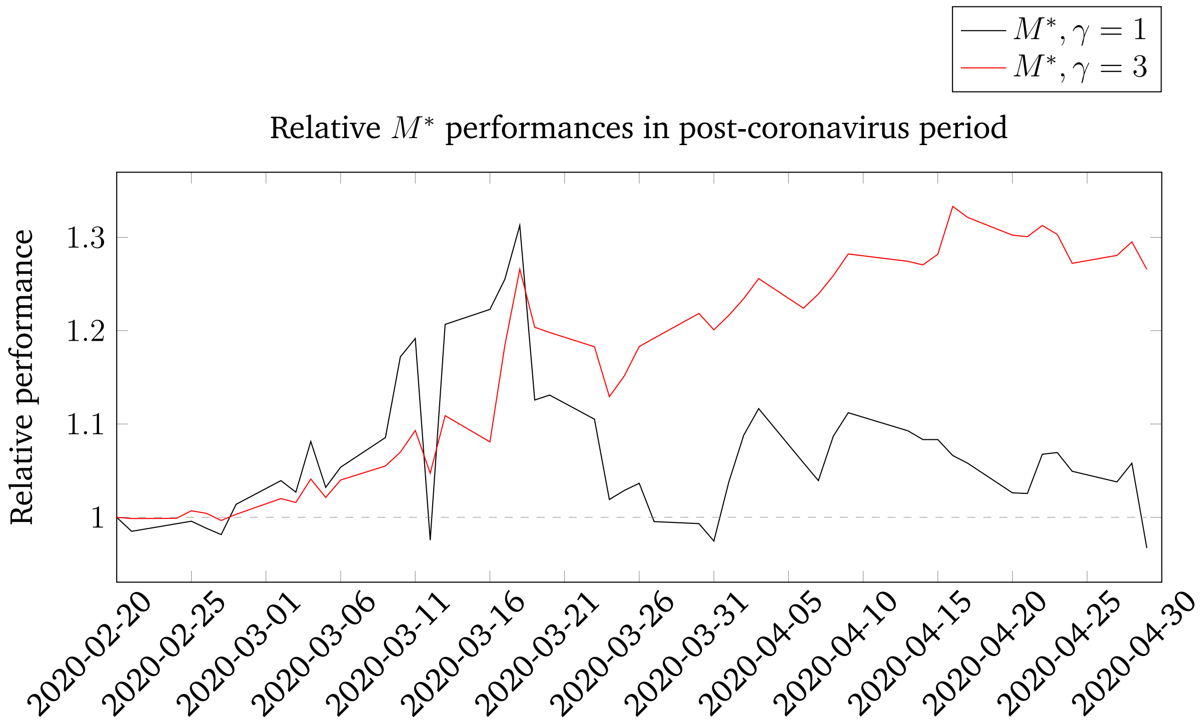


Figure 5.4: Comparing the M^* model performances during the coronavirus period with $\gamma = 1$ vs. $\gamma = 3$

of slow but steady increases, rather than sudden lurches up and down as seen in the $\gamma = 1$ model. As a future consideration (see 6.2.1), it will be of interest to build in some sort of volatility monitor that selects the best value of γ based on current market conditions.

5.5 Sharpe ratio

In tables 5.4 and 5.5, the Sharpe ratios (see 2.5.4) have been calculated for the trading strategies and the index funds, for both the pre- and post-coronavirus periods. Higher values indicate a better rate of return relative to risk.

Model	Sharpe ratio
M^*	**1.733**
M^+ n.s. avg.	1.135
M^+ s. avg	1.062
FTSE 100	0.355
DJI	0.842
S&P 500	1.224

Table 5.4: Pre-coronavirus Sharpe ratios

Model	Sharpe ratio
M^*	0.901
M^+ n.s. avg.	1.135
M^+ s. avg	**1.655**
FTSE 100	-0.353
DJI	0.092
S&P 500	0.325

Table 5.5: Post-coronavirus Sharpe ratios

The risk-adjusted rates of return for both trading strategies presented are considerably greater than those for the index funds. The pre-coronavirus M^* model achieves a Sharpe ratio of 1.733, markedly higher than the 1.35 achieved in [26] although incomparable to the 0.148 reported in [9] due to a different value used for the risk-free rate. Meanwhile, it is the M^+ model with shorting that achieves the greatest ratio for the pre- and post-coronavirus periods combined, at 1.655. The dropoff in the Sharpe ratio of the M^* model over this extended period of time is explained by the introduction of great volatility, while also not turning any new profits, over the course of the coronavirus period, and could be improved via the hedging technique described in section 5.4.2.

5.6 VaR and CVaR

Due to there being under 400 days of simulated trading data, testing VaR or CVaR values at either 99% or 99.9% (see section 2.5.5) would compute data based on a very small sample size. For this reason, the 95% level is used, conveniently the same as employed in [9].

The results of the values, distinguished between pre- and pre-and-post-coronavirus, are displayed in tables 5.6 and 5.7.

Model	VaR	CVaR
M^*	-0.018	-0.025
M^+ n.s. avg.	**-0.005**	**-0.009**
M^+ s. avg	**-0.005**	-0.010
FTSE 100	-0.013	-0.019
DJI	-0.018	-0.023
S&P 500	-0.017	-0.023

Table 5.6: Pre-coronavirus VaR and CVaR values

Model	VaR	CVaR
M^*	-0.022	-0.051
M^+ n.s. avg.	**-0.006**	**-0.011**
M^+ s. avg	**-0.006**	-0.013
FTSE 100	-0.020	-0.038
DJI	-0.028	-0.047
S&P 500	-0.027	-0.045

Table 5.7: Post-coronavirus VaR and CVaR values

Across both observed time periods, the least risky model as indicated by the VaR and CVaR metrics is the M^+ with no shorting. To recap, its VaR at 95% value of -0.005 pre-coronavirus (-0.006 including coronavirus) means there is a 5% chance of losing 0.5% (0.6%) or more on a given day of trading. This is particularly impressive in the context of the index funds - the Dow Jones Industrial Average, for example, reports a 5% chance of losing 2.8% or more on any given day including coronavirus, more risky by a factor of greater than 4.

The CVaR, for which the M^* model is in both tables the worst performer, can be seen as a measure of the average performance in the worst-case scenarios. As discussed around figure 5.4, the M^* model is particularly volatile during periods of extreme market uncertainty, and so the hedging improvements suggested there would likely help to improve the CVaR values. Regardless, the values obtained in this table paint a clear picture of the M^* model being a higher-risk approach (albeit only moderately higher-risk than, say, the Dow Jones), but when considered alongside the Sharpe ratio, this can be seen as a risk that is likely worth taking.

By means of a comparison, [9] reports CVaR values of -0.0126 and -0.0108 in simulated trading across 2012 and 2013 respectively, suggesting a slightly lower risk model than the M^* presented here, although [26] describes this particular market period as having "zero interest rates and rising equity market[s]... [which] are conducive to positive performance". Further issues with the evaluation methods are discussed in section 5.9.

Tables 5.8 and 5.9, found in the chapter summary, provide a final comparison between all models and baseline indices encountered in this chapter.

5.7 Why it works: investigating the matrices

From the previous sections of this chapter, it is evident that there is some basis to conclude that the methodology presented in Chapter 4 works. However, it is not possible from this data to assert *why* it works, or indeed to elucidate the differences in performance between certain stocks in the model.

To do this, a deep dive into the inner workings of the model will be carried out on two stocks: in particular, one of the best-, and one of the worst-performing stocks, in terms of directional prediction accuracy. These are Tesco (57.18% correct predictions and only 38.83% incorrect) and Ryanair (51.86% correct predictions and 46.54% incorrect).

5.7.1 Stock performances

Figure 5.5 shows the relative performances of these two stocks over the simulation period. From observing this graph, there is no immediately obvious reason why Tesco's performance was so strong. Such a reason would be, for example, a graph largely following one linear direction. Instead, Tesco's performance consists mainly of oscillations between 1 and 1.2$\times$ its starting price on 02/11/2018. While not necessarily a trivial stock to predict, it could be said however that this is easier to predict than Ryanair, which fluctuates wildly, almost doubling in price between August 2019 and January 2020.

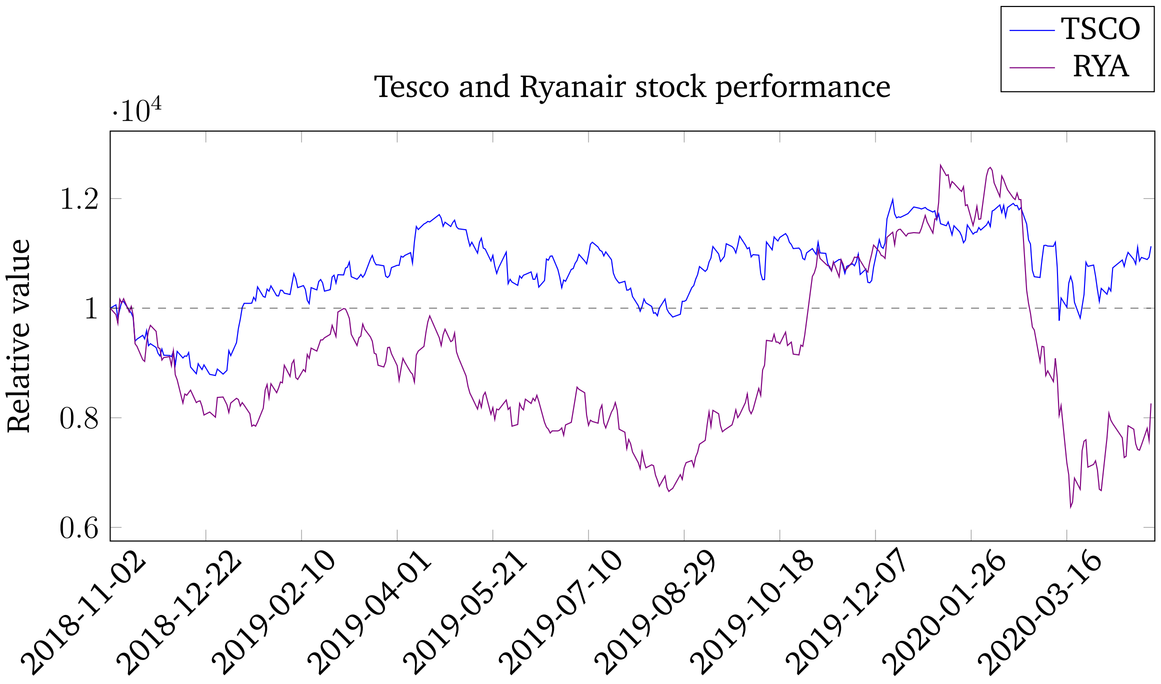


Figure 5.5: Relative performances of Tesco and Ryanair over the trading period

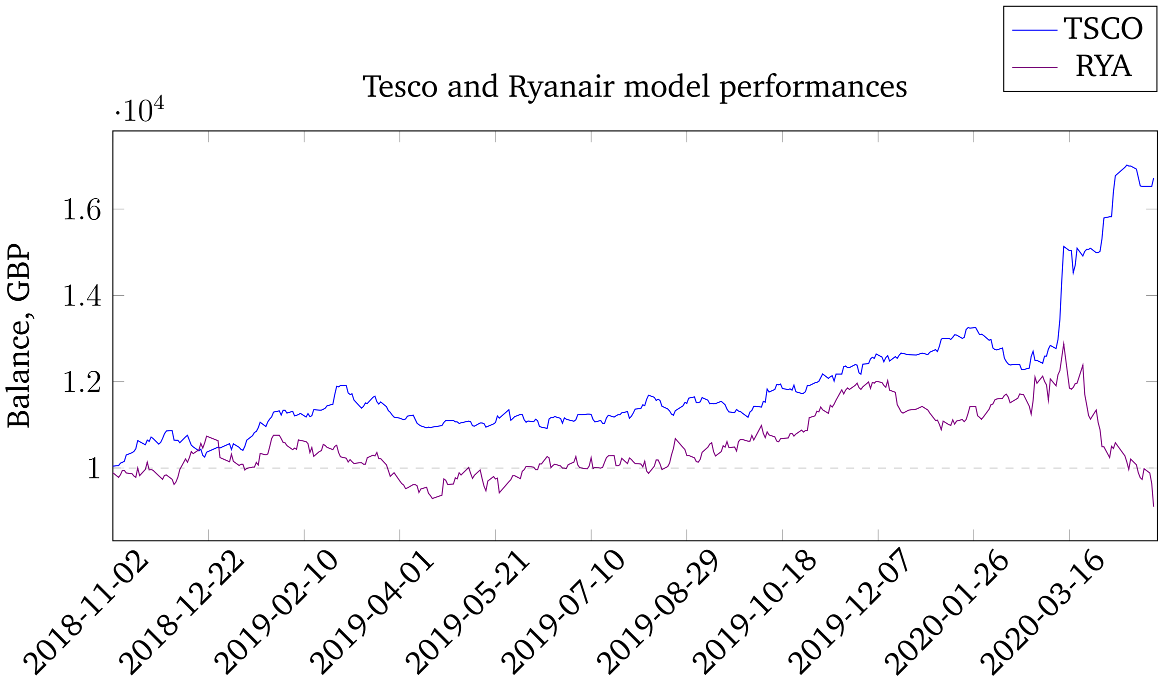


Figure 5.6: Performances of the Tesco and Ryanair models over the trading period

5.7.2 Z-reduced word frequency heatmaps

In figure 5.7, the z-reduced word frequency matrices (see section 4.3.1) are displayed for the full 777 trading day period across which all data was collected. Each individual column represents one trading day's worth of text content, with the rows the z-reduced frequency of the top 1,000 words for the respective stock. It is interesting to note how starkly the entire frame of discussion shifts once the coronavirus pandemic takes hold - observe the structural differences in the heatmap over the final 30 or so trading days compared to the rest. In Tesco's case, a selection of words see a huge jump in usage, including words such as "delivery" and "grateful". For Ryanair, far fewer words actually saw a rise in usage, likely reflecting the reduction in online commentary while all of their aircraft remained grounded, but words that saw a rise in usage include "law", "refund" and "money".

5.7.3 W matrices

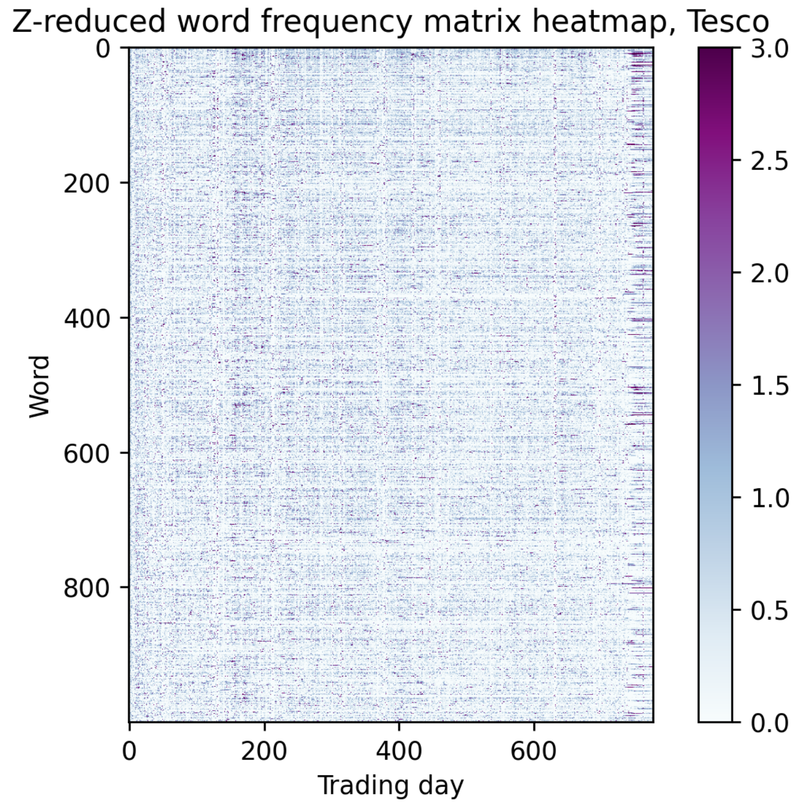
With the z-reduced text content extracted for the two stocks, it is now time to dig into the W matrices. From section 4.4.1,

$$W \in \mathbb{R}^{1 \times m}$$

i.e., the W matrices can effectively be seen as a direct prediction of the effect of each of the top m words on the stock price.

The W matrices are recalculated for every prediction day, using the previous 400 days of text data. As there are 398 overlapping days between two adjacent days' W matrices, there should be little difference between them on a micro (i.e. day-to-day) scale. In figures 5.8 and 5.9, a selection of W matrices are presented from different points in the testing cycles. Each matrix has m (i.e. 1,000) columns; a darker (more purple) stroke at column i represents a positive calculated relation between word i and the stock price, while a lighter stroke represents a negative calculated relation. Some words have been manually annotated onto the figures.

Figure 5.8 shows matrices computed during the simulation for Tesco. The first observation is how well the sparsity constraints (see 4.4.2) have worked. The vast majority of words across each of the matrices have their values set to 0, leaving only the strongest-correlated words to have nonzero values. The nonzero words themselves are, at first glance, a slightly odd selection, but this largely comes down to their relatively even distribution across all 1,000 columns; the more "obvious" choices of words would probably all fall in the first 50 or 100 values, so it is to be expected that the majority of those selected are actually less foreseeable choices. Furthermore



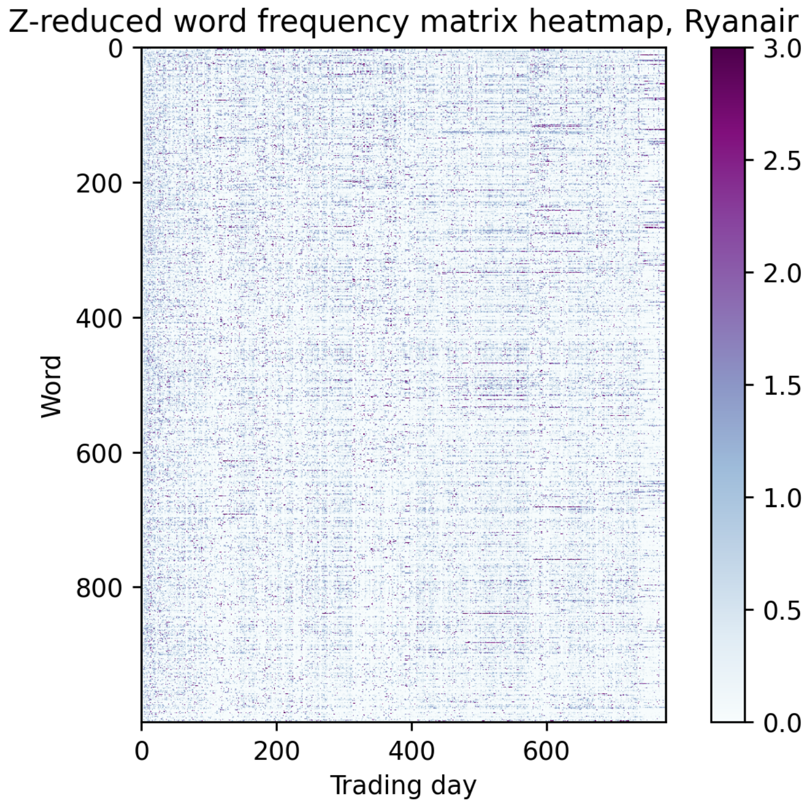


Figure 5.7: Z-reduced word frequency matrix heatmaps for Tesco and Ryanair

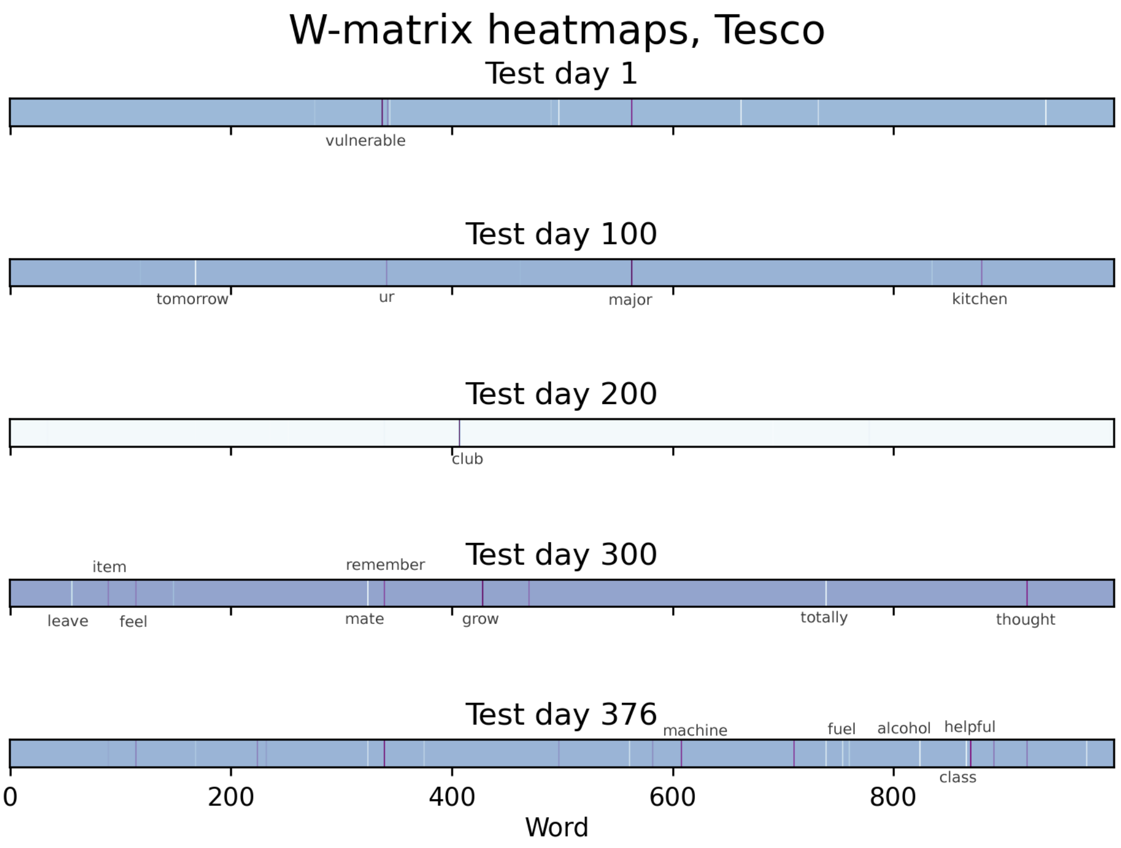


Figure 5.8: Visualisation of a selection of W matrices generated throughout the testing cycle for Tesco

it is also of course the entire point of the model that it is able to calculate underlying trends in the text data - trends that may not necessarily be immediately intuitive.

The matrix produced for test day 200 is perhaps an example of where the hyperparameters are too strict in enforcing sparsity. There is only one clear buy signal in this matrix, for the word “club”, and therefore it probably does not come as much of a surprise that the model's guess for this day was incorrect. The hyperparameters are never going to be perfect for all days - they are selected based on overall averaged performance and so naturally there will be outliers.

Moving on now to the W matrices computed during the testing for the Ryanair model, it is immediately obvious that there is a problem. While the first four matrices displayed appear as expected, the matrix computed on day 376 has completely failed to observe the sparsity constraints; virtually every word in the matrix is correlated with a price movement. This is almost certainly the reason behind the model's poor performance during the coronavirus period - in the final 40 simulated trading days, the model's directional prediction accuracy is only 30.77%, which in turn leads to very poor trading performance as visualised in figure 5.6. Sparsity, or lack thereof, is a direct result of the hyperparameter values, and so it appears that for this particular

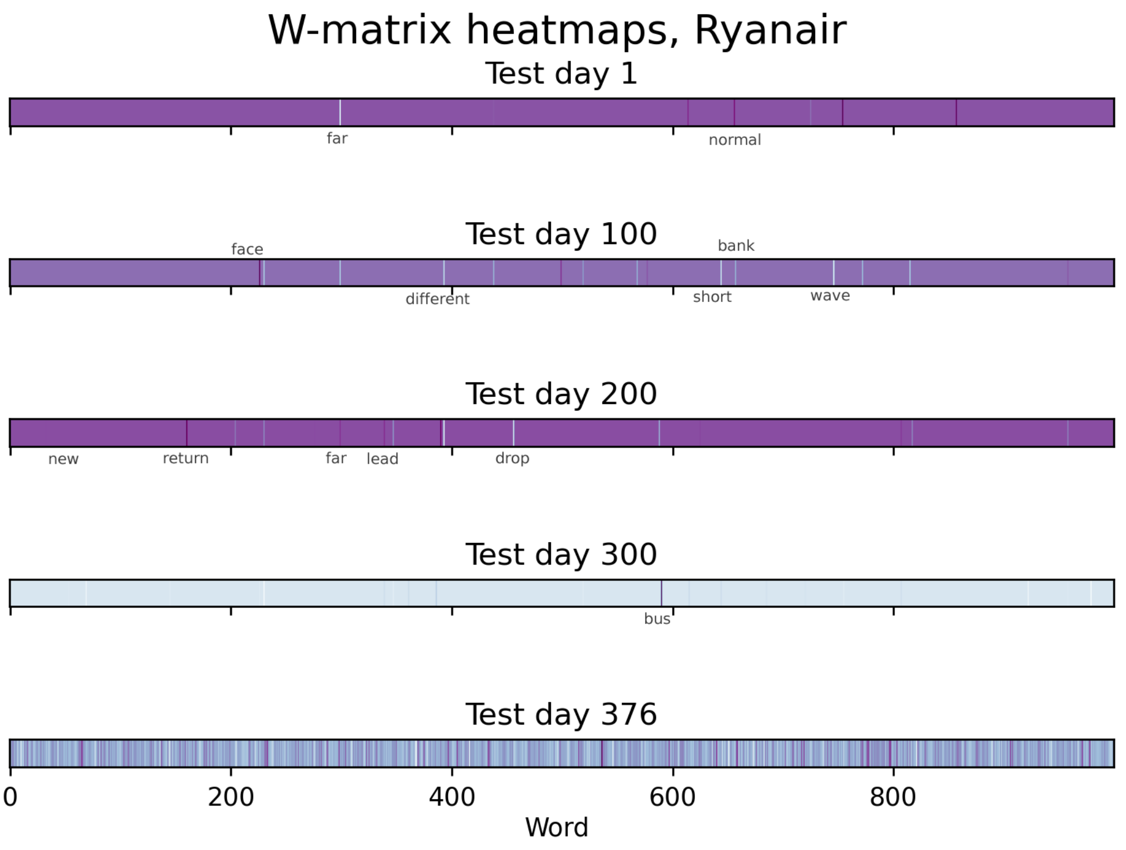


Figure 5.9: Visualisation of a selection of W matrices generated throughout the testing cycle for Ryanair

model the values selected were functional as expected during the pre-coronavirus period, but struggled to make sense of the rapidly changing textual inputs in the latter stages of the simulation.

5.8 Hyperparameter results

Another way to compare the validity of this approach, both generally and also against those found in the literature, is to observe the optimal hyperparameter values. The methods presented in [9] and [26] choose to combine all analysed stocks into one generalised model, with the aim that this model can then be used to forecast the movements of stocks not initially in their database. One set of hyperparameters is used in such models across *all* stocks. Conversely, with the much more fine-grained approach adopted here in this paper, with a separate model produced for each individual stock, a separate set of optimal hyperparameters is produced for each as well. These hyperparameters are displayed in figure 5.10, coloured according to industry sector.

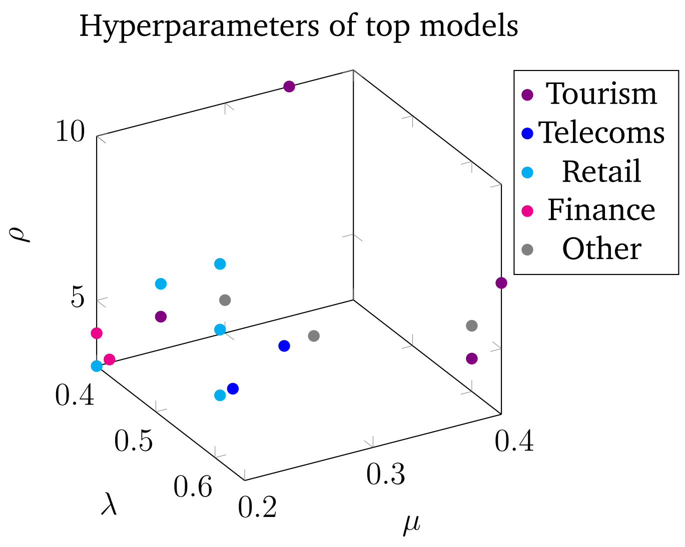


Figure 5.10: λ, μ and ρ values of the top-performing M^+ models.

The distribution of these optimal hyperparameters is a potential method for judging which approach - generalised vs. fine-grained - is better. If all hyperparameters produced by the fine-grained model show signs of clustering around a single point, this suggests that the generalised approach is valid - with largely similar optimal hyperparameters for each company, the benefit of being able to directly apply the model to a previously-unseen stock outweighs the slight reduction in performance from not having specialised hyperparameters. Conversely, if the hyperparameters show little or no overall correlation, this suggests that the fine-grained approach would be likely to lead to considerably higher profits, and gives more credence to the methodology presented in this paper.

What can be seen from figure 5.10 is a lack of *clear* overall correlation, but some bunching towards the lower bounds of each of the hyperparameter values. This bunching is more pronounced among some stock sectors than others; for example, the retail stocks are very much focused into this corner, while the tourism stocks see no apparent correlation whatsoever. Overall though, there are not enough data points to be able to draw a definitive conclusion from this chart.

There are, however, two potential implications for the observation that the hyperparameters show signs of clustering by sector:

- It implies industries are more likely to move as one. The coronavirus pandemic is a perfect example - airline stocks suffered large losses as flights were grounded, followed by a rise in technology stocks as more people staying at home led to increased usage of their services, followed by the eventual recovery of the airline stocks when flights resume at a reasonable scale again [47].
- The model is able to detect these underlying similarities between the movements of stocks sharing an industry sector.

In section 1.3, it was stated that "it may well then transpire that these trends are not only detectable between stocks, but also among stock sectors." While it remains true that a definitive conclusion cannot realistically be drawn from this number of data points, there does at least appear to be some degree of validity to this statement in the findings of figure 5.10.

5.9 Issues and limitations

The hourly resolution price data required for the trading simulation is available via Yahoo Finance, but only up to 730 days previously. This places an upper bound of 730 on the number of days that can be used for both validation and testing sets combined. This number is then whittled down further:

- The stock simulations were run over a period of around a month, meaning that by the time the final simulation is run, the first 30 or so days in the first simulation are not available in the final one;
- While the majority of the UK and US-based stocks have complete data, several of the foreign stocks, such as Credit Suisse, are missing days or even full weeks of data;
- The ADMM solver that produces the predictions does not run particularly speedily, even with the help of Condor. Other students and researchers also submit jobs to Condor on a daily basis, and it operates a priority system to ensure no single user can hold on to too many resources for too long.

Ultimately, with the data collection starting on April 1st 2017 and 400 trading days being used for each training iteration, this left a period of roughly only 376 trading days (depending on exchange) to split between the validation and test sets. The only ways to expand this number would have been to either cut the number of training days, or start the data collection from an earlier date. Cutting the training days is likely undesirable, potentially leading to a less accurate model; starting the data collection earlier, while perfectly feasible on paper, would have led to a considerable increase in data processing time that likely would have reduced the number of stocks included in this final report.

The simulation also assumes a perfect ability to time the trades. The stocks are "bought" exactly at midday and then are sold right on the dot of the closing bell. There is no other reasonable and consistent way to simulate the trading (for example, simulating the selling back of the shares at 15 minutes before the market close might make more sense on paper, but 15-minute resolution data going back over two years is not cheap), and nonetheless this approach is in line with previous papers, making it slightly easier to draw comparisons.

While the trading methodology is borrowed directly from [26], and therefore some comparisons can be made from headline accuracy figures, there are two primary issues with it. Firstly, the data collected in that paper, and the data collected here, are from different times. That paper runs its tests against data from 2013, 2014 and 2015, whereas this paper tests against November 2018 - May 2020. The reason for this is, as has been mentioned, the availability of historical hourly-resolution price data - it was not feasible to obtain hourly prices going back further than 730 trading days. Secondly, trading fees are not taken into account when calculating profit or loss. While simplifying the model, this leads to an inaccurate picture being painted of the ability to translate the model into a real-world trading situation.

Finally, as has already been demonstrated, a directional accuracy of greater than 50% is encouraging, but by no means a guarantee of a profitable return. One particular example is the M^+ shorting model for Cineworld. At the end of the 376 simulation days, the model comes to a halt with £6,058.30 - a near 40% loss in just over two years. However, delving deeper into the model reveals its poor performance is more down to, if anything, bad luck rather than poor predictions - with 181 profitable days and 180 lossmaking days, it actually (just about) maintains a directional prediction accuracy of greater than 50%, yet due to its incorrect guesses happening to cause a bigger loss than the profit gained from its correct predictions, the model ends up making a considerable loss.

5.10 Summary

Tables 5.8 (pre-coronavirus) and 5.9 (pre- and post-coronavirus combined) summarise all of the key statistics studied in this chapter. The M^* model cements its place as by far the highest-returning strategy, with its pre-coronavirus annual returns the highlight of the project. While its impressive Sharpe ratio, yet the lowest VaR and CVaR values in the entire dataset, may characterise it as a high-risk, high-return method, these latter numbers should be regarded in context of their peers. Compared to, say, the Dow Jones Industrial Average, the M^* records only a 0.2% worse CVaR pre-coronavirus, or 0.4% worse including coronavirus, while also posting rates of return several times greater. Viewed in this light, it is surely only fair to describe the M^* as *high-risk* if the same is also said of the index funds themselves, traditionally held as among the lower-risk financial investments available to the everyday investor.

Of course, as outlined in section 5.9, these highly impressive metrics cannot be assumed to be immediately transferable to a trading context. A number of simplifications are made in the simulation, both to be able to draw comparisons to the previous two studies and also to reduce the complexity of some of the modelling, which would need to be carefully reviewed and amended if the model were to be pointed to a live trading situation. There are also places for obvious improvement, such as the introduction of a volatility monitor mentioned in 5.4.2, which are investigated in more detail in the final chapter.

Portfolio	Cum. ret.	Ann. ret.	Best ret.	Worst ret.	SR	VaR	CVaR
M^*	**1.564**	**1.406**	**1.065**	0.961	**1.733**	-0.018	-0.025
M^+ n.s. avg.	1.087	1.064	1.033	**0.972**	1.135	**-0.005**	**-0.009**
M^+ s. avg	1.151	1.113	**1.065**	0.942	1.062	**-0.005**	-0.010
FTSE 100	1.050	1.038	1.026	0.963	0.355	-0.013	-0.019
DJI	1.156	1.118	1.050	0.969	0.842	-0.018	-0.023
S&P 500	1.231	1.173	1.050	0.968	1.224	-0.017	-0.023

Table 5.8: Comparative performance of the M^* model vs index funds, 02/11/2018 - 20/02/2020

Portfolio	Cum. ret.	Ann. ret.	Best ret.	Worst ret.	SR	VaR	CVaR
M^*	**1.512**	**1.320**	**1.237**	0.819	0.901	-0.022	-0.051
M^+ n.s. avg.	1.126	1.083	1.033	**0.972**	1.135	**-0.006**	**-0.011**
M^+ s. avg	1.289	1.186	1.065	0.942	**1.655**	**-0.006**	-0.013
FTSE 100	0.863	0.906	1.081	0.904	-0.353	-0.020	-0.038
DJI	0.975	0.983	1.114	0.871	0.092	-0.028	-0.047
S&P 500	1.079	1.052	1.094	0.881	0.325	-0.027	-0.045

Table 5.9: Comparative performance of the M^* model vs index funds, 02/11/2018 - 01/05/2020

Chapter 6

Conclusions and future work

6.1 Summary

While the idea of purchasing or selling a stock based on the contents of one single conversation may rightly be considered no more than a coin toss masquerading as an investment strategy, it would seem from the results presented in Chapter 5 that, hidden away in the wider global discussions, there is indeed a wealth of valuable information to be extracted. A pipeline has been introduced that has been shown to outperform several index funds across a whole range of metrics, as well as improving on the results of the only two previous papers to have investigated a similar methodology, insofar as comparisons may be drawn with work carried out over a different time period. The decision to forgo a more traditional sentiment analysis-based approach in favour of newer mathematical techniques seems to have paid off rather well.

The pipeline developed also adhered as intended to the ethical restrictions laid out in section 1.4. The tweets are collected and are saved into a CSV file, which initially contains all associated metadata. Once the tweets are passed into the data processing stage, the z-reduced word frequency matrix that is outputted consists solely of numerical values based upon the content of the aggregated tweets for each day. It is impossible to identify individual tweets, let alone any users, from this matrix. The CSV file containing the tweets is then deleted.

With all this said, there are always opportunities to improve the existing work, and so a couple of points are touched upon in the next section.

6.2 Future work

6.2.1 Volatility monitoring with the VIX

Created by the Chicago Board Options Exchange (CBOE) and launched as a realtime index in 1993, the CBOE Volatility Index (VIX) is a measure of the overall stock market's expectation of future volatility, based on S&P 500 options. A higher value represents greater investor fear over the short-term future of the market. Figure 6.1 plots the closing values of the VIX over the last 20 years, highlighting particularly how it rises during periods of global market uncertainty (firstly the 2008 financial crisis, followed by coronavirus in 2020).

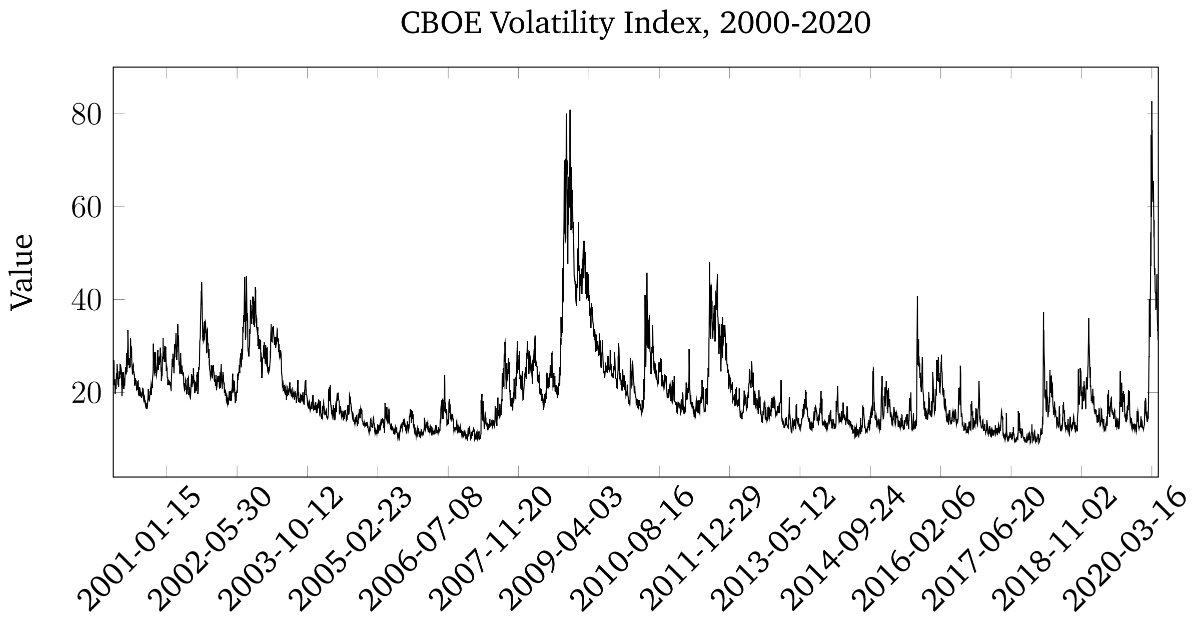


Figure 6.1: Daily closing values of the VIX over the last 20 years

As outlined in 5.4.2, the proposed M^* model performs better during periods of market stability than in times of widespread panic, thanks in part to its calculated optimum value of 1 for the γ hyperparameter. It is quite plausible that this model could benefit from the addition of a new scraper, one that determines a dynamic value for γ based on the observed values of the VIX. When the VIX is low, investor sentiment is more optimistic, and the γ value can be kept at 1. As the VIX rises and investors begin to turn more fearful, the model would be able to hedge itself against this uncertainty by increasing the γ value, spreading its funds across a wider variety of stocks and shielding itself from risk. The VIX levels at which the model decides to change γ would need further investigating; while the coronavirus period saw the index's highest-ever close at 82.69 [48], there is no theoretical upper limit on its value, so simply dividing its historical range of values into segments may not necessarily work as intended in the future.

6.2.2 Additional data sources

The pipeline was created from a UK-focused point of view, and accordingly the majority of the stocks that have been investigated are constituents of the London Stock Exchange. As part of this, the UK edition of Reuters is used for the news collection. One of the side-effects of this is a relative dearth in news content for non-UK stocks. US-listed AT&T, Chegg and jetBlue see far fewer articles collected on average when compared to UK-based companies, while a similar issue arises for Ethereum, a cryptocurrency whose field remains largely unreported in mainstream news. To counter this and improve the model's ability to analyse stocks outside of the UK, a more diversified (yet still neutral) array of news sources could be factored in to widen the coverage. Potential candidates may include Yahoo Finance News, MarketWatch [49] or CoinTelegraph [50], but a considerably more in-depth vetting process would have to be undertaken to ensure their neutrality before making any decisions.

6.2.3 From project to product

In its current form, the project is not yet suited to being able to make automated trading decisions, and a number of steps will have to be taken to make this final transition from project to product.

For the purposes of this project, the individual stages of the pipeline are automated, but the starting and stopping of jobs, transferring data between stages, and producing statistical analyses of each model's performance after the pipeline is complete, are all tasks that are done by hand. This presents no issue for producing the graphs, tables and figures of this report, but is of no use for a self-sustaining investment system. In the envisaged next stage of the project, at the core of the codebase will be a permanently-running job coordinator and scheduler, calling on the data scrapers, matrix producers and ADMM solvers automatically, as and when they are required. Some of the steps in the pipeline that were found to be more computationally-expensive during the simulations will be far less so when it comes to running in real time: rather than having to back-test for hundreds of historical training days, each model will only need to update once a day, with one day's worth of new data, eliminating the reliance on high-throughput computing systems such as Condor.

6.3 Final thoughts

As a relatively nascent approach towards trading and investment, there is no doubt that more work on sparse matrix factorisation in the future will shed light on further improvements that could have been implemented, yet the results produced here speak for themselves.

The efficient market hypothesis does suggests that, on a risk-adjusted basis, the level of returns produced in this report should not be sustainable over any reasonable period of time. While some simplifications have been made to the model in the trading simulation, it is unlikely that, if turned to a real-world situation, the removal of such simplifications would wipe out all of its profit, at least over the period tested. It will be of great interest going forward to incorporate some of the ideas for future work to see if the model continues to perform at such an impressive level.

Bibliography

[1] Runnacles J. *The Role of Social Media in the Hong Kong Protests*. Available from: https://www.diplomaticourier.com/posts/the-role-of-social-media-in-the-hong-kong-protests.

[2] Pak A, Paroubek P. *Twitter as a Corpus for Sentiment Analysis and Opinion Mining*. volume 10, 01 2010.

[3] Agarwal A, Xie B, Vovsha I, Rambow O, Passonneau R. *Sentiment Analysis of Twitter Data*. In *Proceedings of the Workshop on Language in Social Media (LSM 2011)*, pages 30–38, Portland, Oregon, June 2011. Association for Computational Linguistics. Available from: https://www.aclweb.org/anthology/W11-0705.

[4] Bigiotti A, Navarra A. *Optimizing Automated Trading Systems*. In Antipova T, Rocha A, editors, *Digital Science*, pages 254–261, Cham, 2019. Springer International Publishing. ISBN 978-3-030-02351-5.

[5] Nguyen TH, Shirai K, Velcin J. *Sentiment analysis on social media for stock movement prediction*. *Expert Systems with Applications*, 42(24):9603–9611, 2015. doi: https://doi.org/10.1016/j.eswa.2015.07.052.

[6] Nguyen TH, Shirai K. *Topic Modeling based Sentiment Analysis on Social Media for Stock Market Prediction*. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1354–1364, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1131. Available from: https://www.aclweb.org/anthology/P15-1131.

[7] Schumaker RP, Chen H. *Textual Analysis of Stock Market Prediction Using Breaking Financial News: The AZFin Text System*. *ACM Trans. Inf. Syst.*, 27(2), March 2009. ISSN 1046-8188. doi: 10.1145/1462198.1462204. Available from: https://doi.org/10.1145/1462198.1462204.

[8] Schumaker RP, Zhang Y, Huang CN, Chen H. *Evaluating sentiment in financial news articles*. *Decision Support Systems*, 53(3):458 – 464, 2012. ISSN 0167-9236. doi: https://doi.org/10.1016/j.dss.2012.03.001. Available from: http://www.sciencedirect.com/science/article/pii/S0167923612000875.

[9] Ming F, Wong F, Liu Z, Chiang M. *Stock Market Prediction from WSJ: Text Mining via Sparse Matrix Factorization*. In *2014 IEEE International Conference on Data Mining*, pages 430–439, Dec 2014. doi: 10.1109/ICDM.2014.116.

[10] Ahmed W, Bath P, Demartini G. *Using Twitter as a data source: An overview of ethical, legal, and methodological challenges*. 2017.

[11] Beninger K, Fry A, Jago N, Lepps H, Nass L, Silvester H. *Research using Social Media; Users' Views*. 02 2014.

[12] Williams ML, Burnap P, Sloan L. *Towards an Ethical Framework for Publishing Twitter Data in Social Research: Taking into Account Users' Views, Online Context and Algorithmic Estimation*. *Sociology*, 51(6):1149–1168, 2017. doi: 10.1177/0038038517708140. Available from: `https://doi.org/10.1177/0038038517708140`. PMID: 29276313.

[13] Charniak E. *Statistical techniques for natural language parsing*. *AI magazine*, 18 (4):33–33, 1997.

[14] Manning CD. *Part-of-speech tagging from 97% to 100%: is it time for some linguistics?* In *International conference on intelligent text processing and computational linguistics*, pages 171–189. Springer, 2011.

[15] Ramos J et al. *Using TF-IDF to determine word relevance in document queries*. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 133–142. Piscataway, NJ, 2003.

[16] Manning CD, Raghavan P, Schütze H. *Scoring, term weighting, and the vector space model*, page 100–123. Cambridge University Press, 2008. doi: 10.1017/CBO9780511809071.007.

[17] Warriner AB, Kuperman V, Brysbaert M. *Norms of valence, arousal, and dominance for 13,915 English lemmas*. *Behav Res*, 45:1191–1207, 2013. doi: https://doi.org/10.3758/s13428-012-0314-x.

[18] Healy C, Ramaswamy S. *Tweet Sentiment Visualization App*. Available from: `https://www.csc2.ncsu.edu/faculty/healey/tweet_viz/tweet_app/`.

[19] Burgess M. *Google Pixel 4 review: the ultimate Android phone has a big flaw*. Available from: `https://www.wired.co.uk/article/pixel-4-review`.

[20] Feldman R. *Techniques and applications for sentiment analysis*. *Communications of the ACM*, 56(4):82–89, 2013.

[21] Blei DM, Ng AY, Jordan MI. *Latent Dirichlet allocation*. *Journal of Machine Learning Research*, 3(4-5):993–1022, 2003.

[22] Darling WM. *A theoretical and practical implementation tutorial on topic modeling and gibbs sampling*. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 642–647, 2011.

[23] Xiao H, Stibor T. *Efficient Collapsed Gibbs Sampling for Latent Dirichlet Allocation*. *Journal of Machine Learning Research - Proceedings Track*, 13:63–78, 01 2010.

[24] Lettier D. *Your Guide to Latent Dirichlet Allocation*. Available from: `https://medium.com/@lettier/how-does-lda-work-ill-explain-using-emoji-108abf40fa7d`.

[25] Lin C, He Y. *Joint Sentiment/Topic Model for Sentiment Analysis*. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, page 375–384, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585123. doi: 10.1145/1645953.1646003. Available from: `https://doi.org/10.1145/1645953.1646003`.

[26] Sun A, Lachanski M, Fabozzi FJ. *Trade the tweet: Social media text mining and sparse matrix factorization for stock market prediction*. *International Review of Financial Analysis*, 48:272 – 281, 2016. ISSN 1057-5219. doi: https://doi.org/10.1016/j.irfa.2016.10.009. Available from: `http://www.sciencedirect.com/science/article/pii/S1057521916301600`.

[27] Saunders M. *Notes 9: Augmented Lagrangian Methods*, 2015. Available from: `https://web.stanford.edu/class/msande318/notes/notes09-BCL.pdf`.

[28] Volkova S. *Augmented Lagrangian method*. Available from: `https://www.cs.jhu.edu/~svitlana/papers/non_refereed/optimization_1.pdf`.

[29] Bartels RH, Stewart GW. *Solution of the Matrix Equation $AX + XB = C$ [F4]*. *Commun. ACM*, 15(9):820–826, September 1972. ISSN 0001-0782. doi: 10.1145/361573.361582. Available from: `https://doi.org/10.1145/361573.361582`.

[30] Golub G, Nash S, Van Loan C. *A Hessenberg-Schur method for the problem $AX + XB= C$*. *IEEE Transactions on Automatic Control*, 24(6):909–913, 1979.

[31] Weisstein EW. *Schur Decomposition*. Available from: `https://mathworld.wolfram.com/SchurDecomposition.html`.

[32] Malkiel BG. *The efficient market hypothesis and its critics*. *Journal of economic perspectives*, 17(1):59–82, 2003.

[33] Morningstar. *The Sharpe Ratio Defined*. Available from: `http://news.morningstar.com/classroom2/course.asp?docId=2932&page=4`.

[34] Mihov D. *SEC fines Floyd Mayweather and DJ Khaled $750K for cryptocurrency shilling*. Available from: `https://thenextweb.com/hardfork/2018/11/30/mayweather-khaled-cryptocurrency-fine/`.

[35] Thomas J. *Binance CEO Admits to Shilling Binance Coin and Bitcoin*. Available from: `https://beincrypto.com/binance-ceo-admits-to-shilling-binance-coin-and-bitcoin/`.

[36] Baccianella S, Esuli A, Sebastiani F. *SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining.* volume 10, 01 2010.

[37] University of Pittsburgh. *Subjectivity Lexicon*. Available from: `https://mpqa.cs.pitt.edu/lexicons/subj_lexicon/`.

[38] Lunn E. *Tweet your complaint to get a result*, Apr 2011. Available from: `https://www.telegraph.co.uk/finance/personalfinance/8453763/Tweet-your-complaint-to-get-a-result.html`.

[39] Media Bias. *Media Bias Fact Check: Reuters*, Feb 2020. Available from: `https://mediabiasfactcheck.com/reuters/`.

[40] Project T. *Twint Project*, May 2020. Available from: `https://github.com/twintproject/twint`.

[41] Imperial College London. *Condor*. Available from: `https://www.imperial.ac.uk/computing/people/csg/services/hpc/condor/`.

[42] Yahoo Finance. *Yahoo Finance – stock market live, quotes, business amp; finance news*, May 2020. Available from: `https://uk.finance.yahoo.com/`.

[43] Services WRD. *Wharton Research Data Services*, May 2020. Available from: `https://wrds-web.wharton.upenn.edu/wrds/ds/crsp/index.cfm`.

[44] Hearn I. *Research: The best times to post on social media in 2020 (by platform)*, Mar 2020. Available from: `https://www.impactbnd.com/blog/new-research-whats-the-best-time-to-post-on-social-media-in-2`.

[45] *List of pytz timezones*. Available from: `https://gist.github.com/heyalexej/8bf688fd67d7199be4a1682b3eec7568`.

[46] Friedman J, Hastie T, Tibshirani R. *A note on the group lasso and a sparse group lasso*. *arXiv preprint arXiv:1001.0736*, 2010.

[47] *US Nasdaq index recovers all of 2020's losses triggered by Covid-19*, May 2020. Available from: `https://www.theguardian.com/us-news/2020/may/07/us-nasdaq-index-wiped-out-all-of-2020s-losses-triggered-by-covid-19`.

[48] Macroption. *VIX All-Time Highs and Biggest Spikes*, Apr 2020. Available from: `https://www.macroption.com/vix-all-time-high/`.

[49] MarketWatch. *Home Page*, Jun 2020. Available from: `https://www.marketwatch.com/`.

[50] CoinTelegraph. *Cointelegraph Bitcoin amp; Ethereum Blockchain News*, Jun 2020. Available from: `https://cointelegraph.com/`.

[51] Boyd S, Parikh N, Chu E, Peleato B, Eckstein J. *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*. *Found. Trends Mach. Learn.*, 3(1):1–122, January 2011. ISSN 1935-8237. doi: 10.1561/2200000016. Available from: `https://doi.org/10.1561/2200000016`.

Appendix A

Solving for U and W

$$\begin{aligned} L_\rho(A,\, B,\, U,\, W,\, C,\, D) = \|R - ABY\|^2 + \lambda \sum_{j=1}^{m} \|W_j\|_2 + \mu\|W\|_1 + I_+(U) \\ + \operatorname{tr}(C^T(A - U)) + \operatorname{tr}(D^T(B - W)) \\ + \frac{\rho}{2}\|A - U\|_F^2 + \frac{\rho}{2}\|B - W\|_F^2 \end{aligned} \tag{A.1}$$

The augmented Lagrangian in (A.1) is the starting point for the execution of the ADMM algorithm.

ADMM itself is derived from two optimisation algorithms, **dual ascent** and the **method of multipliers** [51]. Both of these methods involve an iterative step in which individual variables are updated in sequence until some convergence criteria are met. Similarly, ADMM is derived by successively updating the augmented Lagrangian L_ρ with respect to A, B, U, and W, one at a time. The Lagrangian multipliers C and D are then updated after each iteration.

$$\begin{aligned} A_+ &= \text{arg min}_A L_\rho(A, B, U, W, C, D) \\ B_+ &= \text{arg min}_B L_\rho(A_+, B, U, W, C, D) \\ U_+ &= \text{arg min}_U L_\rho(A_+, B_+, U, W, C, D) \\ W_+ &= \text{arg min}_W L_\rho(A_+, B_+, U_+, W, C, D) \\ C_+ &= C + \rho(A_+ - U_+) \\ D_+ &= D + \rho(B_+ - W_+) \end{aligned}$$

To avoid rewriting a considerable volume of preexisting mathematical work, the derivations presented below are abridged. The full derivations of the individual variable updates for A, U and W are found in full in [9]. B follows a different derivation using the Bartels-Stewart algorithm, which is outlined in more detail in section 2.4.4.

A.1 Updating A

A is updated by differentiating, setting the derivative equal to zero and then rearranging for A.

$$A_+ = (RY^TB^T - C + \rho U)(BYY^TB^T + \rho I)^{-1}$$

A.2 Updating B

B follows a similar methodology, but cannot be directly rearranged for B. Instead, by rearranging to find a matrix of the form $AX + XB = C$ (known as a Sylvester equation), it is possible to solve for X via the Bartels-Stewart algorithm (see 2.4.4).

$$\left(\frac{1}{\rho}A^TA\right)B + \left(YY^T\right)B = \left(\left(\frac{1}{\rho}(A^TRY^T - D\right) + W\right)YY^T$$

Substituting into the Bartels-Stewart algorithm with

$$\begin{aligned} A &= \left(\frac{1}{\rho}A^TA\right) \\ B &= \left(YY^T\right) \\ C &= \left(\left(\frac{1}{\rho}(A^TRY^T - D\right) + W\right)YY^T \end{aligned}$$

gives a computation for B_+. An implementation of the Bartels-Stewart algorithm is provided as part of the `scipy.linalg` package.

A.3 Updating U

U_+ reduces to

$$U_+ = \left(A + \frac{1}{\rho}C\right)^+$$

A.4 Updating W

W is updated columnwise, where W_j represents column j.

$$
\begin{aligned}
W_j &= \left(\frac{\|w\|_2 - \lambda}{\rho \|w\|_2} \right)^+ w \\
w &= \rho \text{sign}(v) \left(|v| - \frac{\mu}{\rho} \right)^+ \\
v &= B_j + \frac{D_j}{\rho}
\end{aligned}
$$

A.5 Return values

The steps from (A.1) to (A.4) are undertaken, followed by updating the Lagrangian multipliers C and D, until either the U and W matrices both reach convergence, or the maximum iterations are reached. Considering a tradeoff between accuracy and running time, the maximum number of iterations `MAX_ITER` is set to 100 in this model. The matrices U and W are then returned.