Erasmus University Rotterdam

Erasmus School of Economics

Master Thesis Quantitative Marketing and Business Analytics

Predicting movement of the S&P 500 with sentiment, a Dirichlet-Hawkes based approach

Thomas Wieringa - 461169

Supervisor: dr. M.D. Zaharieva

Second assessor: dr. A.M. Schnucker

Abstract

This paper focuses on constructing sentiment indices from financial news articles which exploit topic clusters made by the Dirichlet-Hawkes process with as final goal to predict the next day movement of the S&P 500 in a VAR model. Our research compares the prediction accuracy of multiple sentiment indices and we find that sentiment indices which exploit the clusters made with the Dirichlet-Hawkes process can, in some cases, provide a greater prediction accuracy of the next day price movement of the S&P 500 than the base sentiment score which does not exploit the clusters.

March 5, 2021

Contents

1 Introduction

Making accurate predictions on future stock market movement can lead to very large rewards. Because of this, research about stock market prediction has always been of interest. In the 1930s research by Alfred Cowles proposed that stock market predictors of the day could not beat the market returns (Cowles 3rd, 1933). A theoretical explanation for this can be found in the Efficient Market Hypothesis (EMH) (Fama, 1995). The EMH states that stock prices reflect all of the available information. Changes in the price of a stock are due to the release of new information or overall market changes. Because new information and the overall market are unpredictable (Malkiel, 1973) proposed that the stock market can not be accurately predicted when looking at historic prices, because stock prices follow a random walk around a equilibrium. However, this absolute theory has proven to be not entirely accurate. Only five years later, Fama himself proposed multiple versions of the EMH, which correspond to different levels of information availability. This categorization itself shows that the original EMH does not hold in all cases. Also, empirical evidence against the EMH can be found in numerous studies. Butler and Malaikah (1992) studied thinly traded stocks in Saudi Arabia and Kuwait which showed significant departures from a random walk. Results of Kavussanos and Dockery (2001) similarly showed stocks traded in the Athens Stock Exchange (ASE) to be unsupportive of the EMH.

More recent research has also questioned the Efficient Market Hypothesis (Malkiel, 2003). Critics of the EMH come from different fields of research, namely, behavioral economics (Smith, 2003), behavioral finance (Nofsinger, 2005) and the Socioeconomic Theory of Finance (Prechter Jr and Parker, 2007). The latter one states that financial decision making is partly steered by non-rational sentiment driven motives, which results in non-mean-reverting dynamics, which is the opposite of the equilibrium seeking behaviour described in the EMH.

Building on the notion that markets are in reality rarely strongly efficient, and that financial decision making is steered by non-rational sentiment driven motives multiple studies have tried to link sentiment analysis to stock markets (Nassirtoussi et al., 2014). For instance, Bollen et al. (2011) did research on the correlation between public mood and stock prices. Their research uses the textual data contained in tweets to determine the public mood, which is shown to be positively correlated to the Dow Jones Index. Schumaker et al. (2012) use sentiment gathered from financial news articles to predict the stock movements. Closely relating to our research, Si et al. (2013) use tweets to compute a sentiment index which exploits the clustering of tweets into topics by a Dirichlet Process Mixture model. They compute the sentiment of a single tweet by counting positive and negative words from a sentiment lexicon. To exploit the clusters in the construction of a daily sentiment index they only use the sentiment of the tweets which appear in the biggest topic of the day. The resulting daily sentiment index is then used in a Vector Autoregressive model (VAR) setting to predict the next day movement of the S&P 100. These predictions are made by making a one day ahead prediction and evaluating whether it correctly predicted a rise or a fall. A possible shortcoming of this method of exploiting clusters to compute a sentiment index is that it disregards the dynamics of the topics and it does not consider the sentiment of the tweets which appear in any of the smaller clusters. Sentiments contained within a cluster which is not the biggest, but has a very rapid rising temporal dynamic, are not considered while they could be valuable.

Our research builds on Si et al. (2013) since we also construct daily sentiment indices by counting sentiment words from textual documents, in our case financial news articles. We also construct sentiment indices which exploit topic clusters. Then, we also use our sentiment indices in a VAR setting to make next day price predictions of a stock index. In our case the S&P 500, which contains the top 500 companies on the U.S. stock exchanges. Due to its broad composition it provides a reliable picture of developments in the US stock market.

But our research differs, because instead of a Dirichlet Process Mixture model we use the Dirichlet-Hawkes process (DHP) (Du et al., 2015) to cluster the textual documents. The DHP can be used to cluster textual documents based on their temporal dynamics as well as their textual contents, by combining Dirichlet and Hawkes processes. Dirichlet processes can be used to cluster textual documents, where the likelihood of a document belonging to a certain cluster depends on the size of the cluster. The DHP extends this by assigning a self exciting temporal point process, a Hawkes process, to each cluster. These self exciting temporal point processes are then also taken into account for determining the preferential attachment of a document to a cluster. This allows the DHP to improve clustering of continuous time document streams into topics, because it can separate documents both based on their textual contents as well as their temporal dynamics. It also allows for the forecasting of new arrival times for a cluster, based on the learned temporal dynamics.

Another difference compared to Si et al. (2013) is that we consider multiple ways to exploit the topic clusters to create sentiment indices. Their research only exploited the clusters by using the sentiment contained within the largest cluster for each day, this disregards sentiment not contained in the biggest topic cluster. To overcome this we create two indices which exploit the topic clusters and retain the sentiment contained within all of the articles. We do this by assigning a weight to the individual cluster sentiment based on the size of the cluster. We also differ because we create a weighted index which is based on the one day ahead size predictions of the topic clusters. In this way we also take into account topic dynamics, the idea behind this is if a certain topic cluster is not yet very large but is rising very fast, then its sentiment could be weighted too little when not taking into account topic dynamics. To compare our own sentiment indices we compare them with a few from the literature.

Within this setting we will tackle the following research questions:

- Is the Dirichlet-Hawkes process suitable for clustering financial news?
- Can S&P 500 prediction be improved by incorporating DHP cluster size into the sentiment index?
- Can S&P 500 prediction be improved by incorporating DHP cluster size and dynamics into the sentiment index?

The first research question will be tackled by examining the clusters created by the Dirichlet-Hawkes Process. We do this by running the DHP on a news article data set, for different parameter settings, and examining the resulting clusters.

The second and third research questions will be tackled by creating multiple sentiment indices which exploit the DHP clustering. The results will be compared to a base index, which does not use the clustering. The performance metric we use is the accuracy. To compute the accuracy we first convert each one day ahead price prediction into a correct or incorrect prediction of a rise or fall, also called a sign prediction. The fraction of correct sign predictions is the accuracy.

Our research finds that the next day price movement predictions with a VAR model with a sentiment index can, in some cases, be improved by exploiting topic clusters.

This paper will continue in Section 2, where the data will be presented. Afterwards, in Section 3, the methods and performance measures will be explained. Then, this paper will finish with a discussion of the obtained results in Section 4 and a conclusion on this research in Section 5.

2 Data

2.1 Data Overview

This research uses two data sets. Firstly, we use a data set which contains financial news articles. Secondly, we use a data set of the S&P 500 returns.

The financial news data set contains 306242 financial news articles. These articles are all published within a five month window, from January 1st to May 31st in 2018. Each article was published by one of the following sources: Bloomberg, CNBC, Reuters, WSJ and Fortune. Within these sources only financial oriented articles were extracted. Furthermore, all articles are written in English and published in the United States, which suits our research since we predict the S&P 500. The original data set, obtained from Kaggle (Jeet.J, 2018), contains the textual data of the articles as well as a lot of metadata, such as the url at which the article was uploaded and date of crawling. This research only uses the textual contents and time of publication of the articles.

Figure 1 shows the amount of articles published per day for the entire data set. This plot shows that the articles have a fairly constant publishing rate. We do however see a reoccurring pattern, which takes place around 4 times every month. When we look at the average amount of publications per weekday in Figure 2 it becomes apparent that this reoccurring pattern is caused by a weekly pattern, namely a dip in publications in the weekend.

Figure 3 shows the average number of publications for each hour of the day, where we see that publications start building up around 9:00, reach their peak at around 13:00, and then slowly decrease till just after midnight.

Previous figures only showed information about the temporal attribute of our data set.

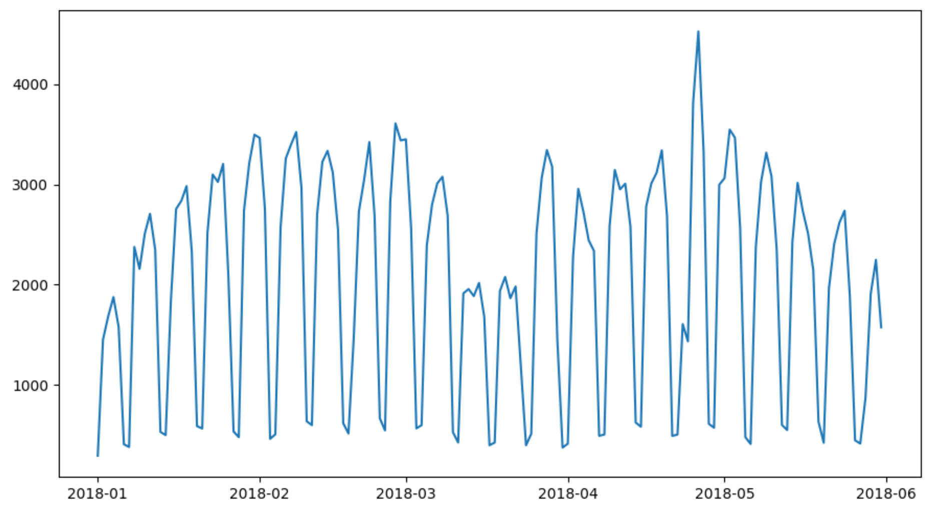

Figure 1: Daily publications

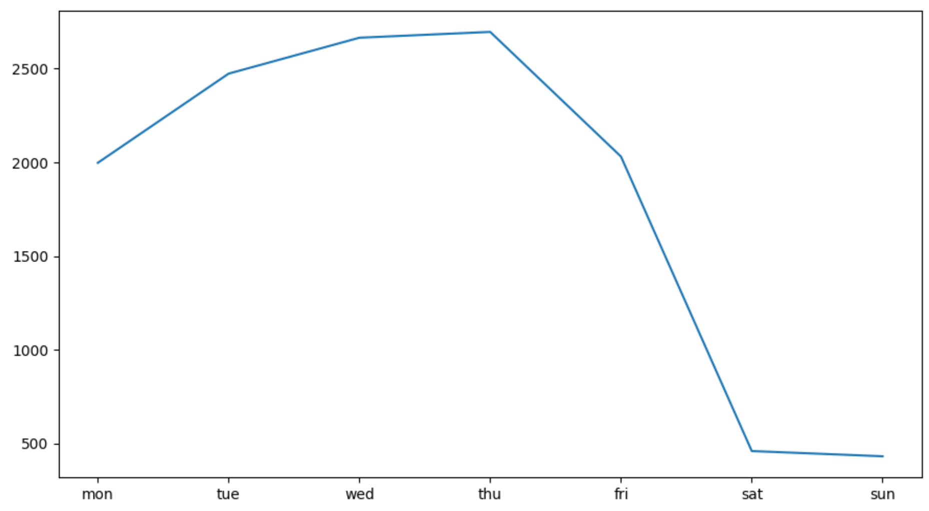

Figure 2: Average number of publications per day of the week.

But our data set also contains textual data, the textual contents of the articles. Figure 4 is a histogram of the log transformed amounts of words per article. We plotted a log transformed histogram because the amount of words per article is very skewed. The minimum amount per article is 0, the maximum 35218 and the mean 401.

We also have a data set containing the time series with daily closing price of the S&P 500. For this research we have picked out the time window which matches our news article data set. Figure 10 shows these daily close values. We can see that this period was fairly

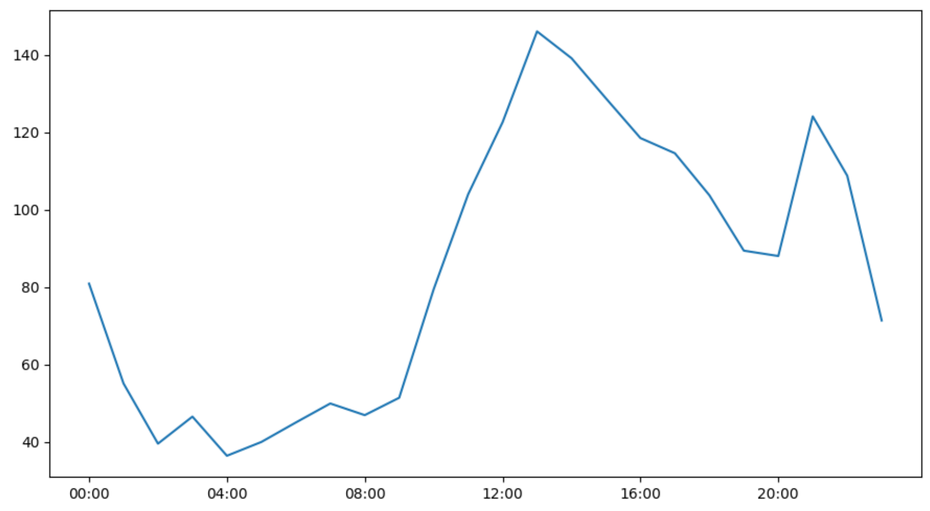

Figure 3: Average number of publications per hour of the day

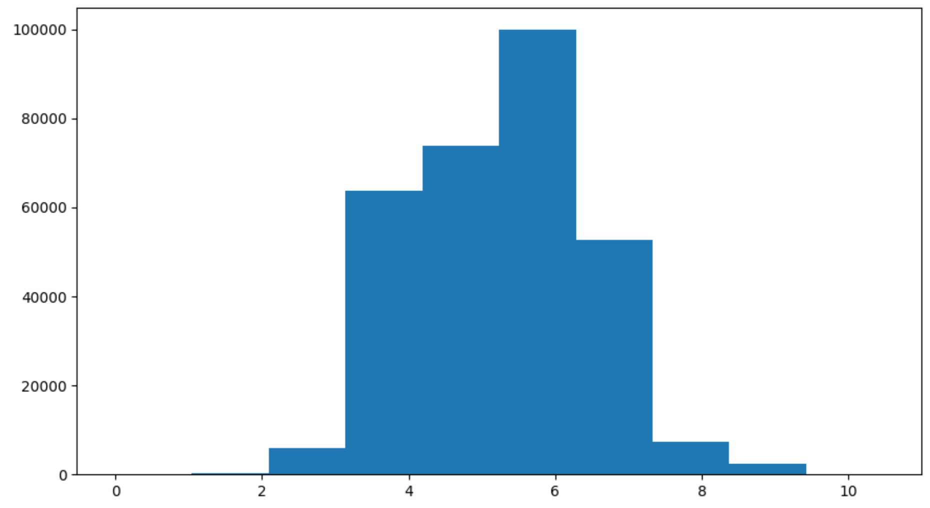

Figure 4: Log transformed words per article

stable, the lowest point is less than a 15% decrease from the highest point. But nevertheless there are two periods of fast decline which are surrounded by steady climb, which make for an interesting prediction problem.

Furthermore, we have a small data set which contains our sentiment lexicon. This sentiment lexicon is used to give a sentiment to an article. We use the sentiment of Hu and Liu (2004), which contains 2004 positive words, like: good, best, great, and 4781 negative words, like: bad, bankrupt and disaster.

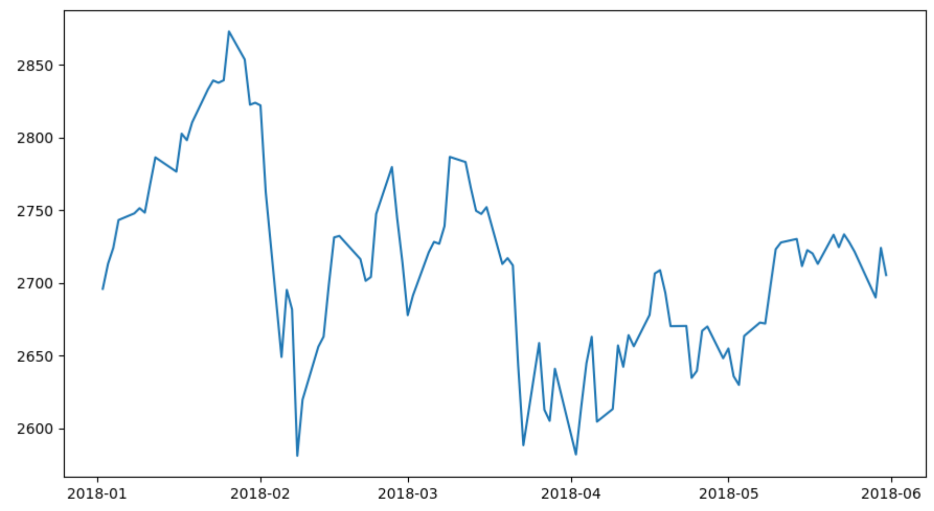

Figure 5: S&P 500 daily closing prices

2.2 Data Preprocessing

To be able to use the textual data in the Dirichlet-Hawkes process we preprocess it first. The main goal of this is to make the entire corpus, the set of unique words, as small as possible while still retaining the information in it, because the smaller the corpus is, the less parameters we will have in our model. We also try to minimize the amount of different conjugations of a word which contain the same or similar meaning. To achieve this we run every article through the following process. To illustrate what goes on in the processing we keep track of the following example sentence:

```
Apple is set to release the best iPhone yet, the iPhone 12. See more on www.apple.com.
```

Firstly, we make all letters lower case, to make sure there is no difference between upper and lower case words.

```
apple is set to release the best iphone yet, the iphone 12. see more on www.apple.com.
```

Secondly, we remove all links which are possibly present in the articles. We do not want these in there because they are very likely unique and thus provide no use for clustering. Neither do they contain sentiment which also makes them redundant.

```
apple is set to release the best iphone yet, the iphone 12. see more on.
```

Thirdly, we remove all special characters, which are also not useful for our clustering or sentiment analysis.

```
apple is set to release the best iphone yet the iphone see more on
```

After we have made these changes, we remove the words which are in our sentiment lexicon from the articles, and use them to calculate the sentiment score of the article. We do this by summing over all opinionated words, where positive words have a value of 1 and negative words have a value of -1. We remove them because we do not want them to interfere with the clustering. We do keep the sentiment score of the article for constructing the sentiment indices later on.

```
apple is set to release the iphone yet the iphone see more on
```

The next step is to stem each word. When we stem the words we chop of the end or the beginning of words to bring multiple conjugations or derivations back to a single word, an example of this is: ‘Playing’, ‘plays’, ‘played’ would all be stemmed to ‘play’. In our example iphone is stemmed to iphon, but iphones would also be stemmed to iphon. This allows for a single parameter in the clustering model which relates to both iphone and iphones. For stemming we use the NLTK python package.

```
appl is set to releas the iphon yet the iphon see more on
```

Finally, we do one more processing step to our article data set. In this processing step we remove all words which appear in less than 100 articles, because these words are too specific. We also remove words which appear in more than 50% of all the articles, because they are too general. At last, we only keep the 10,000 most used words, because we want to limit the amount of parameters in our model.

```
appl  releas  iphon  iphon
```

And in this final form the news information is used in the clustering.

3 Methodology

In this section the methods used in this research are described. Section 3.1 explains the Dirichlet-Hawkes process which is used as a prior for clustering the articles. Section 3.2 describes the Inference algorithm which is responsible for clustering the articles. Section 3.3 describes how the different sentiment index time series are computed. And finally, In Section 3.4 it is described how we forecast the movement of the S&P 500 using our sentiment indices with the VAR model.

3.1 Dirichlet-Hawkes process

To cluster the articles into topics, a Dirichlet-Hawkes process (DHP) based model is used (Du et al., 2015). The Dirichlet-Hawkes process is a combination of the Dirichlet Bayesian nonparametric Process and the Hawkes temporal point process. The resulting process can be used to model a stream of events based on the intensity, through the Hawkes process, as well as the diversity, through the Dirichlet process. Before explaining the Dirichlet-Hawkes process we will separately explain both fundamental building blocks of the DHP, the Dirichlet process and the Hawkes process, in Section 3.1.1 and 3.1.2 respectively.

3.1.1 Dirichlet process

The first building block of the DHP is the Dirichlet process (DP). Dirichlet processes are probability distributions with other probability distributions as its range. The DP is first proposed by Ferguson (1973) The process is parameterized by a concentration parameter $\alpha > 0$, and a base distribution $G_0(\boldsymbol{\theta})$. The expected value of the DP is the base distribution, which means samples from a DP will lay around the base distribution. Also, distributions drawn from the DP are almost always discrete, even if the base distribution is continuous. The amount of discretization is controlled by α. If $\alpha \to 0$, the samples are all concentrated on a single value. When $\alpha \to \infty$, the samples are continuous. In the middle of these two limits the samples are discrete with decreasing concentration when α increases. Because a DP sample, G, is a distribution itself, it allows us to draw a sample from it, $\boldsymbol{\theta}_{1:n}$. Drawing from a DP can be simulated with the Chinese Restaurant process:

1. Draw $\boldsymbol{\theta}_1$ from G_0 , for n = 1
2. For $n > 1$:
   (a) Draw $\boldsymbol{\theta}_n$ from G_0 with probability $\frac{\alpha}{\alpha+n-1}$.
   (b) Reuse $\boldsymbol{\theta}_k$ for $\boldsymbol{\theta}_n$ with probability $\frac{m_k}{\alpha+n-1}$, where m_k is the amount of past samples with values $\boldsymbol{\theta}_k$.

This process is called the Chinese Restaurant process because it follows the following analogy: We have a (Chinese) restaurant with infinitely many tables, each corresponding to a different value of $\boldsymbol{\theta}$. When a new customer walks into the restaurant he or she has two choices. Either picking a new table, or joining a table where there are already m_k people seated. The first option is executed with probability $\frac{\alpha}{\alpha+n-1}$, and matches drawing a new $\boldsymbol{\theta}$ from G_0 as $\boldsymbol{\theta}_n$ The second option is executed with probability $\frac{m_k}{\alpha+n-1}$, and matches reusing a previous $\boldsymbol{\theta}_k$ as $\boldsymbol{\theta}_n$. Because the probability of reusing a $\boldsymbol{\theta}_k$ is directly proportional to m_k, the Chinese restaurant process captures the 'rich-get-richer' phenomenon. To translate this to the analogy: The more people are sitting at a table, the higher the chance a new customer will join the table. This behaviour combined with the fact that there is always a small probability that a new table, or cluster, is filled or created makes the DP an often used prior in data clustering models.

3.1.2 Hawkes process

The second building block of the Dirichlet-Hawkes process is the Hawkes process. Hawkes processes are a special kind of temporal point process, first proposed by A.G Hawkes (Hawkes, 1971a) and (Hawkes, 1971b). Temporal point processes are processes which have a set of isolated event times t_n as outcome. Many real life events can be modelled as a temporal point process; an example of this is people visiting a website. Temporal point processes can be characterized by their conditional intensity function, which models the chance of a next arrival given all previous events. $\lambda(t)dt$ denotes the probability of the occurrence of a new event given all past events $\mathcal{T}$, in a small interval $[t, t + dt)$:

$$\lambda(t)dt = \mathbb{P}\{ \text{ event in } [t, t + dt) \mid \mathcal{T}\} \tag{1}$$

The most basic temporal point process is a homogeneous Poisson process. The intensity function of a Poison process is independent of all past events and constant over time: $\lambda(t) = \lambda_0 \geq 0$. An inhomogeneous Poisson process also has an intensity function which in independent of past event, but its intensity does vary over time: $\lambda(t) = gt \geq 0$. A Hawkes process has an intensity function which is both varying over time as well as dependent on previous events. Its intensity is defined as:

$$\lambda(t) = \gamma_0 + \alpha \sum_{t_i \in \mathcal{T}} \gamma\left(t, t_i\right), \tag{2}$$

where $\gamma_0 \geq 0$ is the base intensity, which is independent of past events, and $\gamma\left(t, t_i\right)$ is the triggering kernel function which holds the temporal dependency. When there are no previous events, the intensity function is equal to the base γ_0, but the intensity can increase because of previous events. This self-exciting behaviour is characteristic for Hawkes processes. This behaviour makes Hawkes processes suitable for modelling activities with self exciting behaviour, such as likes on social media: On social media platforms the intensity of the next like is dependent on the likes before, since posts gain traction when they get more likes. Hawkes processes are very adaptable because multiple types of functions can used as kernel functions. Functions can either be chosen in advance, for example, $\gamma\left(t, t_i\right) = \mathbb{I}\left[t > t_i\right]$ or $\gamma\left(t, t_i\right) = \exp\left(-\left|t - t_i\right|\right)$. Parameters can also be added to kernel functions. These parameters can then be estimated through the likelihood. In this way a kernel function can be fitted to the data (Aalen et al., 2008).

3.1.3 Dirichlet-Hawkes process

Before delving deeper, we first explain the general DHP, which is a process from which we can sample multiple arrival times with corresponding event types. This process is parameterized by the intensity parameter, $\lambda_0 > 0$, which controls the intensity of the creation of new events, a base distribution $G_0(\boldsymbol{\theta})$, defined over an event space $\boldsymbol{\theta} \in \Theta$, and multiple triggering kernel functions $\{\gamma_{\boldsymbol{\theta}}\left(t, t'\right)\}$, each associated to an event type $\boldsymbol{\theta}$. With these we can generate data as follows:

1. Draw t_1 from Poisson $\left(\lambda_0\right)$ and $\boldsymbol{\theta}_1$ from $G_0(\boldsymbol{\theta})$

2. For $n > 1$:

   (a) Draw $t_n > t_{n-1}$ from Poisson $\left(\lambda_0 + \sum_{i=1}^{n-1} \gamma_{\boldsymbol{\theta}_i}(t, t_i)\right)$

   (b) Draw $\boldsymbol{\theta}_n$ from $G_0(\boldsymbol{\theta})$ with probability :

   $$\frac{\lambda_0}{\lambda_0 + \sum_{i=1}^{n-1} \gamma_{\boldsymbol{\theta}_i}(t_n, t_i)},$$

   or reuse $\boldsymbol{\theta}_k$ for $\boldsymbol{\theta}_n$ with probability :

   $$\frac{\lambda_{\boldsymbol{\theta}_k}(t_n)}{\lambda_0 + \sum_{i=1}^{n-1} \gamma_{\boldsymbol{\theta}_i}(t_n, t_i)},$$

   where $\lambda_{\boldsymbol{\theta}_k}(t_n) := \sum_{i=1}^{n-1} \gamma_{\boldsymbol{\theta}_i}(t_n, t_i)\, \mathbb{I}\left[\boldsymbol{\theta}_i = \boldsymbol{\theta}_k\right]$ is the intensity of the Hawkes process for previous events corresponding to $\boldsymbol{\theta}_k$

This process can capture four desirable properties: Preferential attachment, the higher the intensity of a cluster, the higher the probability of a next event of that cluster. Adaptive number of clusters, because of the base intensity there is always a probability of creating a new cluster. Self-excitation, the intensity of a clusters' Hawkes process is dependent on previous draws of the same cluster. Temporal decays, the triggering kernels decay over time. Also, the temporal dynamics, controlled by the triggering kernel functions can be learned from data.

3.1.4 Generating text

This DHP process can be used as a prior for clustering a continuous-time stream of documents, where the goal is to find clusters based on both the contents and temporal dynamics. For this purpose we use an altered version of the process to generate text. This is done by using the $\boldsymbol{\theta}_i$ which we sample from the DHP as the parameters for the content model. Also, we let clusters have a distinct triggering kernel, by defining the triggering kernel as the weighted sum of K base kernels.

We use the sampled vector $\boldsymbol{\theta}_i$ as the parameters for the content model. In our case we choose a Dirichlet distribution, $\mathrm{Dir}(\boldsymbol{\theta}|\boldsymbol{\theta}_0)$, for $G(\boldsymbol{\theta})$. Then, we can sample the words for the n-th document from a multinomial distribution, $\boldsymbol{w}_n \sim \mathrm{Multi}\left(\boldsymbol{\theta}_{s_n}\right)$. Here $\boldsymbol{w}_n$ corresponds to an integer vector, where the entries account for the number of times a certain word appears in the document. $\boldsymbol{\theta}_{s_n}$ refers to the parameters which belong to the cluster of document n, since s_n refers to the cluster label of document n.

Furthermore, we let each cluster have their own temporal dynamics by constructing the Hawkes Process triggering kernel as a linear combination of K base kernels. Here, vector $\boldsymbol{\alpha}_{\boldsymbol{\theta}_{s_n}}$ contains the parameters for cluster s_n, which we abbreviate to $\boldsymbol{\alpha}_{\boldsymbol{\theta}}$:

$$\gamma_{\boldsymbol{\theta}}\left(t_i, t_j\right) = \sum_{l=1}^{K} \alpha_{\boldsymbol{\theta}}^{l} \cdot \kappa\left(\tau_l, t_i - t_j\right), \tag{3}$$

where $\kappa\left(\tau_l, t_i - t_j\right)$ is the kernel function, $t_j < t_i$, $\sum_l \alpha_{\boldsymbol{\theta}}^l = 1$, $\alpha_{\boldsymbol{\theta}}^l > 0$, and τ_i is the reference time point belonging to a base kernel. We use the Gaussian RBF kernel,

$$\kappa(\tau_l, t_i - t_j) = \exp(-(t_i - t_j - \tau_l)^2)/2\sigma_l^2)/\sqrt{2\pi\sigma_l^2} \tag{4}$$

Because every cluster has its own $\boldsymbol{\alpha}_{\boldsymbol{\theta}}$ we can track its unique evolving process. Given $\mathcal{T} = \{(t_n, \boldsymbol{\theta}_n)\}_{n=1}^{N}$, the intensity function of the cluster $\boldsymbol{\theta}$ is:

$$\lambda_{\boldsymbol{\theta}}(t) = \sum_{t_i < t} \gamma_{\boldsymbol{\theta}}\left(t, t_i\right) \mathbb{I}\left[\boldsymbol{\theta}_i = \boldsymbol{\theta}\right]. \tag{5}$$

Then, the likelihood of observing $\mathcal{T}$ before time T is

$$\exp\left(-\sum_{\boldsymbol{\theta}_i = \boldsymbol{\theta}} \boldsymbol{\alpha}_{\boldsymbol{\theta}}^{\top} \boldsymbol{g}_{\boldsymbol{\theta}} - \Lambda_0\right) \prod_{\boldsymbol{\theta}_i = \boldsymbol{\theta}} \sum_{t_j < t_i, \boldsymbol{\theta}_j = \boldsymbol{\theta}} \boldsymbol{\alpha}_{\boldsymbol{\theta}}^{\top} \boldsymbol{k}\left(\Delta_{ij}\right), \tag{6}$$

where $g_{\boldsymbol{\theta}}^l = \sum_{t_i < T, \boldsymbol{\theta}_i = \boldsymbol{\theta}} \int_{t_i}^{T} \kappa\left(\tau_l, t - t_i\right) dt$ and $\Lambda_0 = \int_0^T \lambda_0 dt$. Then the the integral $g_{\boldsymbol{\theta}}^l$ has the analytic from:

$$\sum_{t_i < T, \boldsymbol{\theta}_i = \boldsymbol{\theta}} \frac{1}{2}\left(\operatorname{erfc}\left(-\frac{\tau_l}{\sqrt{2\sigma_l^2}}\right) - \operatorname{erfc}\left(\frac{T - t_i - \tau_l}{\sqrt{2\sigma_l^2}}\right)\right). \tag{7}$$

We consider the reference time points, $\boldsymbol{\tau}_{\boldsymbol{\theta}}$, and standard deviations, $\boldsymbol{\sigma}_{\boldsymbol{\theta}}$, as fixed parameters so they do not need to be sampled. Then, the generative process looks like this (Du et al., 2015):

1. Draw t_1 from Poisson $\left(\lambda_0\right)$, $\boldsymbol{\theta}_1$ from Dir $\left(\boldsymbol{\theta} \mid \boldsymbol{\theta}_0\right)$, and α_1 from Dir $\left(\boldsymbol{\alpha} \mid \boldsymbol{\alpha}_0\right)$

2. For each word v in document $1: w_1^v \sim$ Multi $\left(\boldsymbol{\theta}_1\right)$

3. For $n > 1:$

(a) Draw $t_n > t_{n-1}$ from Poisson $\left(\lambda_0 + \sum_{i=1}^{n-1} \gamma_{\boldsymbol{\theta}_i}(t_n, t_i)\right)$

$$\text{where } \gamma_{\boldsymbol{\theta}_i}(t_n, t_i) = \sum_{l=1}^{K} \boldsymbol{\alpha}_{\boldsymbol{\theta}_i}^{l} \cdot \kappa(\tau_l, t_n - t_i)$$

(b) Draw $\boldsymbol{\theta}_n$ from Dir $(\boldsymbol{\theta} \mid \boldsymbol{\theta}_0)$ with probability

$$\frac{\lambda_0}{\lambda_0 + \sum_{i=1}^{n-1} \gamma_{\boldsymbol{\theta}_i}(t_n, t_i)}$$

and draw $\boldsymbol{\alpha}_n$ from Dir $(\boldsymbol{\alpha} \mid \boldsymbol{\alpha}_0)$, or reuse previous $\boldsymbol{\theta}_k$ for $\boldsymbol{\theta}_n$ with probability

$$\frac{\lambda_{\boldsymbol{\theta}_k}(t_n)}{\lambda_0 + \sum_{i=1}^{n-1} \gamma_{\boldsymbol{\theta}_i}(t_n, t_i)}$$

where $\lambda_{\boldsymbol{\theta}_k}(t_n) = \sum_{i=1}^{n-1} \gamma_{\boldsymbol{\theta}_i}(t_n, t_i)\, \mathbb{I}[\boldsymbol{\theta}_i = \boldsymbol{\theta}_k]$

(c) For each word v in document n :

$$w_n^v \sim \text{Multi}(\boldsymbol{\theta}_n)$$

3.2 Inference

We cluster the documents by applying the Sequential Monte Carlo (SMC), or particle filter method. We denote the latent cluster labels of all documents as $s_{1:n}$. In SMC, a particle $f \in \{1, ...F\}$ contains an estimation of the cluster memberships $s_{1:n-1}$ for all past documents based on all past documents and arrivals times, $d_{1:n}$ and $t_{1:n}$. When a new document, $d_{1:n}$, with arrival time $t_{1:n}$ arrives, we update each particle as follows.

To sample the cluster label s_n for document d_n, we use:

$$\mathbb{P}\{s_n = k \mid t_n, rest\} = \begin{cases} \frac{\lambda_{\boldsymbol{\theta}_k}(t_n)}{\lambda_0 + \sum_{i=1}^{n-1} \gamma_{\boldsymbol{\theta}_i}(t_n, t_i)} & \text{if } k \text{ occupied} \\ \frac{\lambda_0}{\lambda_0 + \sum_{i=1}^{n-1} \gamma_{\boldsymbol{\theta}_i}(t_n, t_i)} & \text{otherwise} \end{cases} \tag{8}$$

to calculate the probabilities of document d_n belonging to a already formed cluster as well as the probability that d_n creates a new cluster. These chances are based on the learned triggering kernels of the existing clusters and the base density λ_0.

Given the sampled cluster membership we can update the triggering kernel of the corresponding cluster. Updating the triggering kernel equates to updating the parameter $\boldsymbol{\alpha}_{\boldsymbol{\theta}_{s_n}}$. The Bayesian rule dictates that the posterior is:

$$P\left(\boldsymbol{\alpha}_{\boldsymbol{\theta}_{s_n}} \mid \mathcal{T}_{s_n}\right) \propto P\left(\mathcal{T}_{s_n} \mid \boldsymbol{\alpha}_{\boldsymbol{\theta}_{s_n}}\right) P\left(\boldsymbol{\alpha}_{\boldsymbol{\theta}_{s_n}} \mid \boldsymbol{\alpha}_0\right), \tag{9}$$

where $\mathcal{T}_{s_n} = \{(t_i, s_i) \mid s_i = s_n\}$ or all past events of the same cluster. To update $\boldsymbol{\alpha}_{\boldsymbol{\theta}_{s_n}}$ we can draw a set of K samples, $\left\{\boldsymbol{\alpha}^i_{s_n}\right\}_{i=1}^K$, from the prior $P\left(\boldsymbol{\alpha}_{\boldsymbol{\theta}_{s_n}} \mid \boldsymbol{\alpha}_0\right)$, which is the Dirichlet distribution with parameters $\boldsymbol{\alpha}_0$. With this sample we can update by taking a weighted average.

$$\hat{\boldsymbol{\alpha}}_{s_n} = \sum_{i=1}^{K} \omega_i \cdot \boldsymbol{\alpha}^i_{s_n}, \tag{10}$$

where the weights, ω_i are defined as:

$$\omega_i = \frac{P\left(\mathcal{T}_{s_n} \mid \boldsymbol{\alpha}^i_{\boldsymbol{\theta}_{s_n}}\right) P\left(\boldsymbol{\alpha}^i_{\boldsymbol{\theta}_{s_n}} \mid \boldsymbol{\alpha}_0\right)}{\sum_i P\left(\mathcal{T}_{s_n} \mid \boldsymbol{\alpha}^i_{\boldsymbol{\theta}_{s_n}}\right) P\left(\boldsymbol{\alpha}^i_{\boldsymbol{\theta}_{s_n}} \mid \boldsymbol{\alpha}_0\right)}. \tag{11}$$

After updating the triggering kernel we update the particle weight by,

$$w_n^f \propto w_{n-1}^f \cdot P\left(d_n \mid s_n^f, d_{1:n-1}\right) \tag{12}$$

Here, because of the Dirichlet-Multinomial conjugate relation,

$$P\left(d_n \mid s_n,\ rest\right) = \frac{\Gamma\left(C^{s_n\backslash d_n} + V\boldsymbol{\theta}_0\right) \prod_v^V \Gamma\left(C_v^{s_n\backslash d_n} + C_v^{d_n} + \boldsymbol{\theta}_0\right)}{\Gamma\left(C^{s_n\backslash d_n} + C^{d_n} + V\boldsymbol{\theta}_0\right) \prod_v^V \Gamma\left(C_v^{s_n\backslash d_n} + \boldsymbol{\theta}_0\right)}, \tag{13}$$

where $C^{s_n\backslash d_n}$ denotes the word count of cluster s_n without document d_n, C^{d_n} denotes the word count of document d_n. And where $C_v^{s_n\backslash d_n}$, and $C_v^{d_n}$ denote the count of the v-th words in the same sets respectively. Finally, V is the vocabulary size of the data.

After we have sampled the cluster label, updated the triggering kernel and updated the particle weight for each particle $f \in \{1, ...F\}$ we normalize the particle weights w^f by dividing each weight w^f by the sum of the weights, $\sum_{f\in\{1...F\}} w^f$.

After we have updated the weights of each particle we resample the particles if $||\boldsymbol{w}||_2^{-2} < \epsilon$, where ϵ is a predefined threshold. In SMC, resampling means throwing away particles with a low weight, or low probability of being correct, and replacing them with the particles with a higher weight. This is done to prevent a so called weight collapse, where one particle has a weight very close to 1 and the others a negligible weight. We coded the inference algorithm in python 3 and based it on previous work (JFChi, 2018).

3.3 Sentiment index

The goal of this research is to construct a sentiment index time series which exploits the clustered topics to predict stock market movement most accurately. The forecasting challenge is to predict the next-day movement of the S&P 500. This next-day forecast will be made with a sentiment index up to the current day.

We consider multiple strategies to construct an index. However, all of the strategies make use of the same opinion lexicon O, a list of positive and negative words, proposed by Hu and Liu (2004). This lexicon will be used in the following way: $l(w)$ equals 1 if w is a positive opinion word, -1 if w is a negative opinion word and 0 otherwise. This results in equation:

$$\delta_a = \frac{1}{W_a} \sum_{w \in W_a} l(w), \tag{14}$$

where δ_a is the sentiment score of article a, and W_a is the set of opinionated words in article a. It is good to note that we we do not take into account the opinionated words for the clustering because we assume that they do not contain information about topic of an article. When calculating the sentiment of a single article we do take into account the opinionated words.

3.3.1 Baseline strategy (BL)

The first strategy that we apply is a baseline strategy. In this baseline the overall sentiment of all articles published in a day is measured by taking the average sentiment score within the day,

$$S_t^{BL} = \frac{\sum_{a \in \boldsymbol{A}_t} \delta_a}{|\boldsymbol{A}_t|}, \tag{15}$$

where S_t is the sentiment index at time t, and $\boldsymbol{A}_t$ is the set of articles published in day t. $|\boldsymbol{A}_t|$ is the cardinality of this set, which means the amount of articles published on day t. With this baseline we can compare and evaluate the performance of the following measures.

3.3.2 Biggest cluster strategy (BC)

The second strategy that we apply is the biggest cluster strategy. In this baseline the overall sentiment of the articles which appear in the biggest cluster of the day is measured by taking the average sentiment score within the day,

$$S_t^{BC} = \frac{\sum_{a \in \boldsymbol{A}_t^{BC}} \delta_a}{|\boldsymbol{A}_t^{BC}|}, \tag{16}$$

where S_t is the sentiment index at time t, and $\boldsymbol{A}_t^{BC}$ is the set of articles published in day t in the biggest cluster of that day. $|\boldsymbol{A}_t^{BC}|$ is the cardinality of this set, which means the amount of articles within the cluster on day t. With this baseline we can compare and evaluate the performance of the following measures.

3.3.3 Weighted by cluster size based strategy (CS)

The third index is an index which uses the cluster sizes to create a weighted score. The main idea behind this index is that the sentiment of bigger topics has a bigger effect on the movement of the stock market. This and the following strategy both use the learned clusters. We have the data about the clusters in the form of s_a where a is the index of the article. We define the set of clusters C as the set of unique values in vector of cluster memberships, s_a.

Then, to create this index we evaluate the sentiment of each individual cluster:

$$S_t^c = \frac{\sum_{a \in \boldsymbol{A}_t^c} \delta_a}{|\boldsymbol{A}_t|}, \tag{17}$$

where S_t^c is the sentiment index at time t for cluster c and $\boldsymbol{A}_t^c$ is the set of documents in cluster c published in day t. This set can be constructed with the sampled cluster labels s_n of each article. With this topic specific score we create a weighted overall index:

$$S_t^{CS} = \sum_{c \in C_t} \frac{|\boldsymbol{A}_t^c|^2}{|\boldsymbol{A}_t|^2} S_t^c. \tag{18}$$

3.3.4 Kernel forecasting based strategy (FCS)

Lastly, an index is created which exploits the learned triggering kernels of the clusters. The idea behind this index is that topics which are fast rising today will have a big impact on tomorrow. The Dirichlet-Hawkes process learns the triggering kernel of each individual cluster. What this means in practice is that we have a vector $\boldsymbol{\alpha_\theta}$ for each cluster $\boldsymbol{\theta}$. This vector contains the weights for K base kernels. This learned triggering kernel can be used to predict arrivals of new events. This index exploits this by forecasting the amount of arrivals for the next day. These predicted arrivals will then be used to create an index in a similar weighted fashion as the cluster size weighted index, $\boldsymbol{S}^{CS}$. The index is created as follows:

$$S_t^{KF} = \sum_{c \in C_t} \frac{|\hat{\boldsymbol{A}}_{t+1}^c|^2}{|\hat{\boldsymbol{A}}_{t+1}|^2} S_t^c \tag{19}$$

where $|\hat{\boldsymbol{A}}^c_{t+1}|$ and $|\hat{\boldsymbol{A}}_{t+1}|$ are the predicted amount of articles 1 day ahead. We make these one day ahead predictions by sampling new arrivals from the learned triggering kernels for the next day. We do this multiple times and take the average.

3.3.5 Indices from the literature

Next to the indices we construct ourselves we also consider multiple indices from the literature. We compare our own indices with the indices from the literature by examining the values and comparing the forecast performance.

Firstly, we consider the VIX index, which is a measure for expected volatility. Secondly, we consider the Economic Policy Uncertainty (EPU) index by Baker et al. (2016), which combines three components to quantify economic uncertainty. Namely, one component is economic news coverage, the second component is disagreement among financial forecasters and the third is the amount of federal tax code provisions which are set to terminate in coming years. The third index we consider is the Daily News Sentiment (DNS) index by Buckman et al. (2020) which closely resembles our own indices by extracting sentiment from economic related articles.

3.4 S&P 500 Predictions

To predict the movement of the S&P 500 with our sentiment indices we use a Vector Autoregressive model (VAR) (Si et al., 2013). A VAR model with a lag of 1 is given by:

$$S_t = \phi_{11} S_{t-1} + \phi_{12} y_{t-1} + \epsilon_{S,t} \tag{20}$$

$$y_t = \phi_{21} S_{t-1} + \phi_{22} y_{t-1} + \epsilon_{y,t}. \tag{21}$$

In this model, S_t is the sentiment time series, y_t is the daily closing price of the S&P 500 and $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2]$ are the model parameters, which represent the influence of the lagged sentiment and price on the current sentiment and price. $\boldsymbol{\epsilon}$ are the error terms, which are assumed to be normally distributed with mean 0 and variance $\boldsymbol{\sigma}^2_{\boldsymbol{\epsilon}}$. We fit the model using least squares regression. For the fitting and predicting process we use a moving window, that is, we train in $[t, t+w]$ and predict $[t+w+1]$, where w is the amount of days in the training window. We do this because, firstly, we are interested in the short term effect of the sentiment index, and secondly, in this way we have more test and training points. In the prediction process we make one day ahead forecasts. For each one day ahead forecast we count the forecasts which predict the right direction, up or down. Using this count we calculate the percentage of times the sentiment index correctly predicts the next day movement. We try training windows of 1, 2, 3, 4, 5 and 6 weeks. Furthermore, there are only stock market trades on business-days, so not on weekends and holidays. To account for this we disregard the dates in the sentiment indices where the stock market is closed. We consider lags of 1, 2, 3 & 4 days.

Furthermore, since we are using a VAR model our time series need to be stationary. We test for this with the Augmented Dickey-Fuller (ADF) test, which has the null hypothesis that there is a unit root in the time series. The alternative hypothesis can differ based on the chosen equation. In our case the alternative hypothesis is stationarity (Fuller, 2009). The result section will contain the outcomes of applying the ADF test on the S&P 500 time series and the sentiment indices. We use the statsmodels package in python to implement the tests.

4 Results

This section contains the results obtained using our methods. Firstly, we discuss results of clustering the financial news articles with the Dirichlet Hawkes process in Section 4.1. Secondly, in Section 4.2 we discuss the different financial news sentiment indices. Section 4.3 contains the results from the VAR regression predictions on the S&P 500 with the sentiment indices.

4.1 Dirichlet-Hawkes clustering

In this section we discuss the results obtained from clustering the financial news article data set using the Dirichlet-Hawkes process. The main goal of the clustering is to group together articles with the same topic. In this research we have used a similar setup as in Du et al. (2015). We have defined the triggering kernels with reference time points, $\boldsymbol{\theta_0}$, at 0.5, 1, 8, 12, 24, 48, 72, 96, 120, 144 and 168 hours with respective bandwidths, $\boldsymbol{\sigma_0}$, at 1, 1, 8, 12, 12, 24, 24, 24, 24, 24, and 24 hours. Because of this setup we can capture both short term and long term dynamics. The concentration parameter is set to $\alpha = 0.1$ and the Dirichlet prior for language model is set to $\boldsymbol{\theta}_0 = 0.01 * \mathbf{1}$ for each value in the vector. For inference we used 8 particles. Since the base intensity parameter λ_0 determines the rate at which new clusters are created, it has great influence over the quality of the results. Because we do not infer the base intensity, we have run the clustering on five different values: 1, 0.5, 0.1, 0.05 and 0.01. Due to computational limitations we only used articles published in the first month of our data set, which is January 2018. This month contained 59073 articles. Table 1 shows how many clusters were created for each of the five base intensities.

λ_0	1	0.5	0.1	0.05	0.01
Amount of clusters	3574	1907	442	245	76

Table 1: Amount of clusters per base intensity

Looking at the results in Table 1, the influence of the base intensity becomes apparent. A base intensity of 1 results in 3574 clusters while a base intensity of 0.01 results in only 76 clusters.

Cluster Size / Base Intensity	$(\infty, 10000)$	[10000,1000)	[1000,100)	[100,10)	[10,1)	[1]
1	0	10	52	377	2261	874
0.5	0	12	55	294	1102	444
0.1	0	13	33	121	200	75
0.05	0	16	28	83	80	38
0.01	2	10	11	22	18	13

Table 2: Number of clusters per base intensity and cluster size range

Table 2 shows the amount of cluster created within six different size ranges for each base intensity. We consider these different size ranges to gain more insights into different types of clusters. Here it becomes apparent that the smallest base intensity is the only base intensity which creates clusters with more than 10000 articles, namely 2 of them. For the range $[10000, 1000)$ all of the base intensities create a similar amount of clusters with the smallest amount being 10 for the smallest and largest base intensity, and the largest amount being 16 for the base intensity 0.1. For the smaller ranges we see that a higher intensity leads to more clusters being created.

Cluster Size / Base Intensity	$(\infty, 10000)$	[10000,1000)	[1000,100)	[100,10)	[10,1)	[1]
1	0,00%	0.28%	1.45%	10.55%	63.26%	24.45%
0.5	0.00%	0.63%	2.88%	15.42%	57.79%	23.28%
0.1	0.00%	2.94%	7.47%	27.38%	45.25%	16.97%
0.05	0.00%	6.53%	11.43%	33.88%	32.65%	15.51%
0.01	2.63%	13.16%	14.47%	28.95%	23.68%	17.11%

Table 3: Percentile distribution of clusters over cluster size

Table 3 shows which percentage of clusters falls within each size range. Here the share of clusters within size ranges $(\infty, 10000)$, $[10000, 1000)$ and $[1000, 100)$ increases when the base intensity decreases. For the size range $[100, 10)$, the share increases as well for 1 till 0.05, only decreasing a bit for 0.01 compared to 0.05. For the size range $[10, 1)$, the complete opposite happens; here the share increases when the base intensity increases. The share of clusters with only one single article stays fairly stable for the different base intensities. While it does decrease when the base intensity decreases till 0.05, and only rising again a bit for 0.01, the differences are relatively small compared to the other ranges. Conclusively, a higher base intensity lead to more clusters being created but also favours small clusters, i.e., clusters with more than 1 and a maximum of 10 articles, more than lower base intensities, which create relatively more larger clusters, namely clusters with more than 10 documents.

While these figures provide us with insight about the distribution of the cluster sizes it still leaves questions about the quality of the clusters. Since we do not know the true clusters of the news articles, we need to examine the words contained within documents in the clusters to make statements about the quality of the clusters. By examining these textual contents, we can say something about the cluster uniqueness and coherence.

To quantify cluster uniqueness, we compare the word vectors of each cluster. The word vector of each cluster contains the count of each word of the documents contained within the cluster. By computing the cosine similarity of two of these vectors we can say something about the textual similarity of two clusters and thus how well they reflect two different topics. Cosine similarity between two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ is defined as: $\cos(\boldsymbol{a},\boldsymbol{b}) = \frac{\mathbf{a}\cdot\mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|}$. To examine the overall performance of a base intensity we compute the cosine similarity between all cluster pairs contained within a size range for each base intensity. By doing this we can say something about the uniqueness and similarity of the clusters contained within a size range.

What we consider desirable is that any two clusters have low similarity, which means that the topics contained within the clusters are textually different. Table 4 shows the average cosine similarity for each base intensity and cluster size range. To be able to compare clusters with different sizes fairly we divide the word distribution vector of each cluster by the sum of that vector.

Base Intensity \ Cluster Size	(,10,000)	[10,000,1000)	[1000,100)	[100,10)	[10,1)	[1]	All ranges
1	-	0.8	0.61	0.28	0.11	0.05	**0.11**
0.5	-	0.73	0.59	0.29	**0.1**	0.04	0.12
0.1	-	0.67	0.56	0.3	0.11	**0.03**	0.16
0.05	-	0.67	0.51	0.31	0.11	0.04	0.21
0.01	**0.63**	**0.56**	**0.43**	**0.24**	0.12	0.04	0.22

Table 4: Average between cluster pairwise similarities, in bold the lowest similarity per cluster size range.

Table 4 shows that the base intensity of 0.01 results in the least similar clusters for size ranges (∞,10000), [10000,1000), [1000,100) and [100,10). This is expected since this intensity creates by far the least amount of clusters. Because of this, the clusters created with this intensity have relatively more articles contained within them, which in turn makes it more likely that the clusters are more different but do contain articles which in reality have different topics. Furthermore, we conclude that for all base intensities the cosine similarities of the clusters within each size range are very similar, especially for the smaller clusters. Independent of the base intensities we also conclude that the larger the cluster, the larger the cosine similarity tends to be. Following this, we can conclude that the large clusters contain articles which are all very similar to one another topic wise. This can be explained by the fact that financial news articles often revolve around very generic topics, like investment related new or news about government decision making, where a lot of the same words are used. So, articles about these generic topics tend to cluster in a few very large clusters with similar topics. Furthermore, it can be seen that for all size ranges the cosine similarity between clusters decreases when the base intensity increases, meaning that a higher base intensity results in more similar clusters and thus less well separated topics.

To quantify the coherence, we compare the word vectors of individual articles contained within a cluster. To examine the overall performance of a base intensity, we compute the average pairwise similarity of 3000 randomly chosen articles within a cluster, using all of the articles if a cluster contains less than 3000 articles. Of these averages within cluster similarities we then compute a average again. Here we consider a high similarity desirable since that relates to articles containing similar words and thus similar topics. These averages are shown in Table 5.

Base Intensity \ Cluster Size	$(\infty, 10000)$	[10000,1000)	[1000,100)	[100,10)	[10,1)
1	-	0.0556	**0.0748**	0.0692	0.0719
0.5	-	0.0575	0.0742	0.0671	0.0699
0.1	-	0.0636	0.0746	0.0669	0.0683
0.05	-	0.06	0.0706	0.0673	**0.093**
0.01	**0.0536**	**0.0674**	0.0742	**0.0757**	0.0788

Table 5: Average within cluster pairwise similarities, in bold the highest per cluster size range.

Table 5 shows that, although there are small differences, within cluster similarity is relatively similar for each base intensity and cluster size range. Because these differences are so small, it is hard to make conclusions regarding the influence of the base intensity on the coherence of the topics.

To make the conclusion on which base intensity is best suited for creating our sentiment indices, we take into account both the similarity measures as well as the cluster size distributions. First of all, we want to have a base intensity which creates unique and coherent topics, which by looking at the similarities points us at the lowest tested base intensity, which is 0.01. But since we are creating a sentiment score which uses the size of the clusters to increase or decrease the influence of the cluster sentiment on the final sentiment index, we do not want the documents to pile up in a few big clusters, since this will diminish the effect of our weighted approach. And since this behaviour of articles piling up in a few large clusters is more present when the base intensity is lower, choosing the right base intensity is a tradeoff between cluster quality and cluster size distribution.

Our final choice is 0.1, because it is a compromise between good cluster quality and a good cluster size distribution.

4.2 Sentiment indices

This section shows the different sentiment indices and compares them to indices from the literature. As explained in the methodology, we have created four different sentiment index time series using the financial news articles. We have a baseline (BL) index which is the average daily sentiment which does not use the clusters. We have an index based on the average sentiment in the biggest cluster on each day, the biggest cluster (BC) index. We also have an index which is a weighted average of the clusters sentiments, weighted by the cluster sizes (CS). Furthermore, we have an index which is similarly weighted by size, but uses the next day forecasted cluster sizes (FCS). Both weighted indices give a greater weight to bigger clusters.

The following figures contain the time series plots of our own indices, the indices in Figure 6, the VIX index in Figure 7, the EPU index in Figure 8, the DNS index in Figure 9 and for reference the closing values of the S&P 500 in Figure 10.

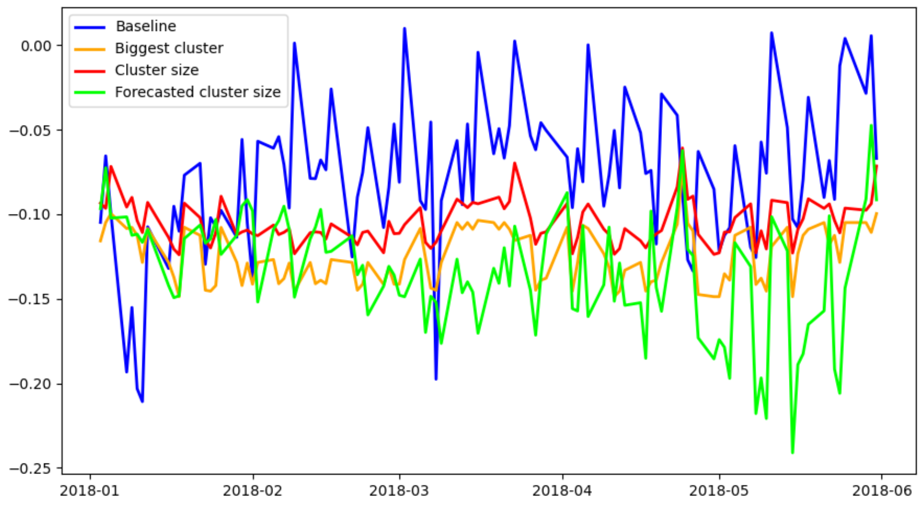

Figure 6: Sentiment index values

Figure 6 shows our own indices. Here, we can see that overall the baseline index is more positive, because of larger values, compared to the indices which use the clustering. We can also see that the indices which exploit the clusters are very similar, especially for the first three months. In the last two months the forecasted cluster size index has a few negative sentiment spikes which cannot be seen in the other indices. When comparing our indices with the prices of the S&P 500 in Figure 10 it is hard to recognize patterns.

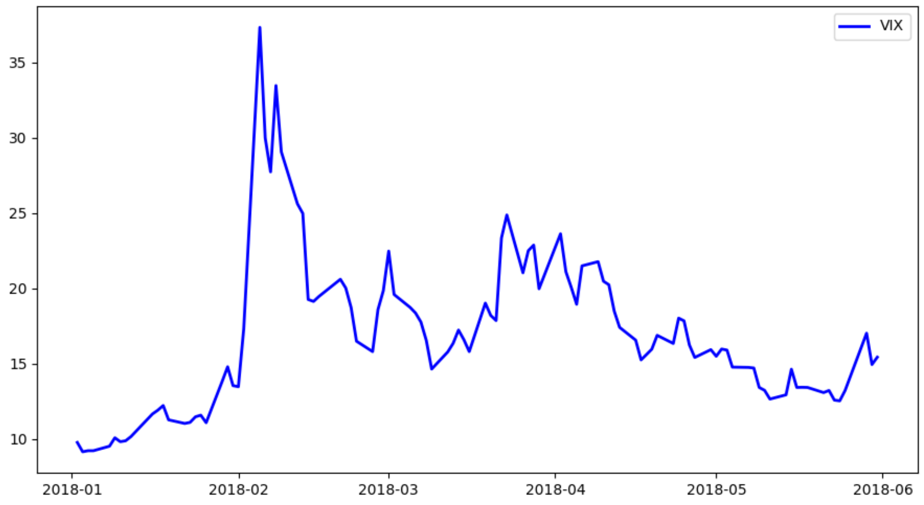

Figure 7: VIX index values

The VIX index represents the amount of expected volatility, and it is also called the fear index. The higher the value of the VIX the higher the fear and the higher the volatility is to be expected. In Figure 7 we can see a very large surge of the VIX in the beginning of February, meaning a sure in fear. When looking at the S&P 500 prices we can see that beginning of march was a period which saw a large dip of the price, so this matches what we see in the VIX.

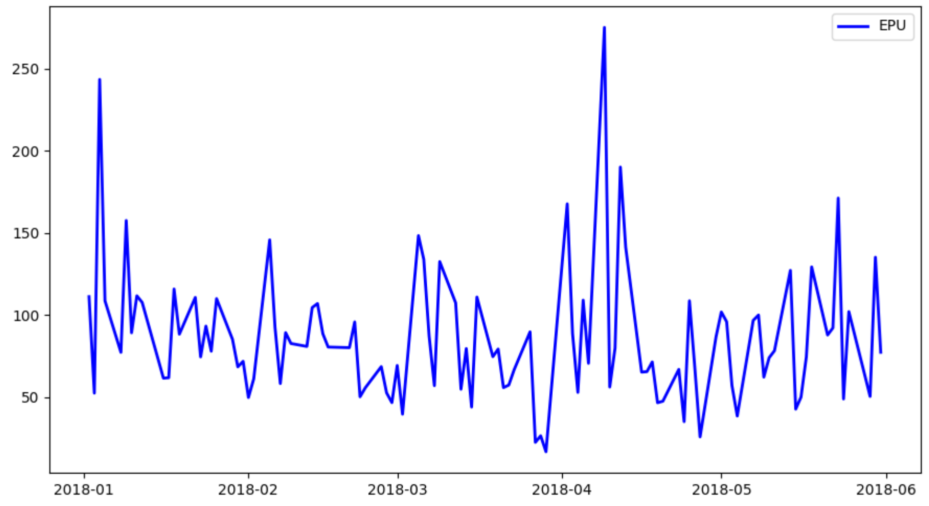

Figure 8: EPU index values

The EPU index tries to capture the overall uncertainty on economic policy. When comparing the values of this index in Figure 8 with the prices of the S&P 500 it is hard to recognize any matching patterns. We see two spikes of the EPU, one in the beginning of January, and one in the middle of April. But we cannot see any direct result of these spikes in the prices of the S&P 500.

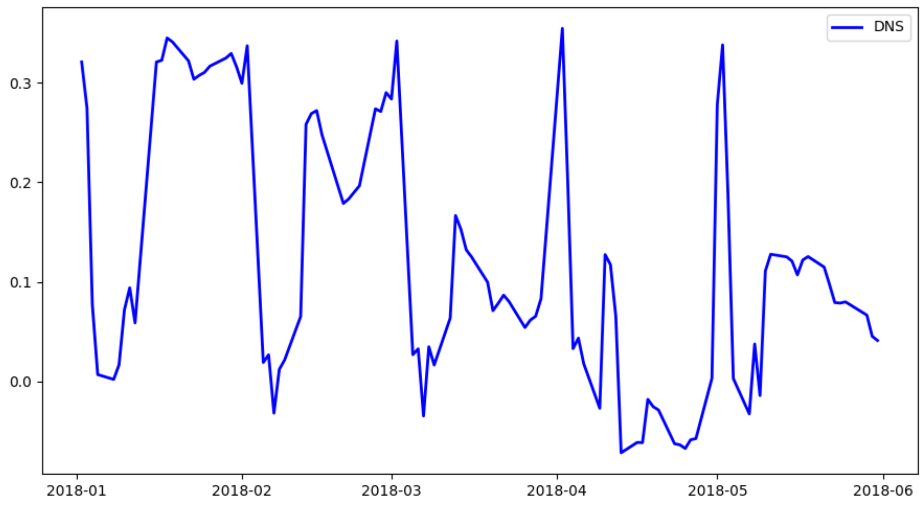

Figure 9: DNS index values

The DNS index is the most similar to ours, since it represents the sentiment contained in economic news articles. For this index we can also not really recognize matching patterns with its values compared to the S&P 500. It seems like there is almost some kind of reoccurring pattern, since around the first of each month there is a surge followed by a dip which cannot be seen in the prices of the S&P 500.

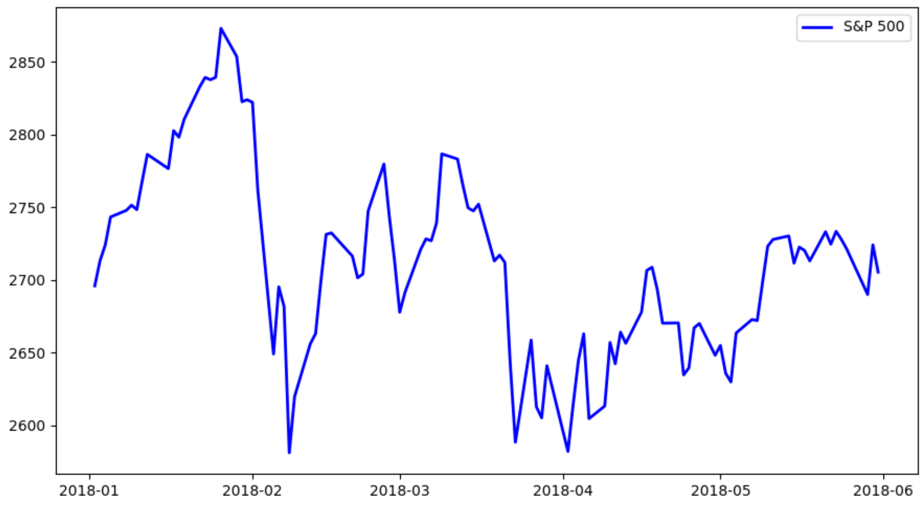

Figure 10: S&P 500 daily closing prices

4.3 S&P 500 prediction

This section contains the results obtained from predicting the S&P 500 with the sentiment indices in a VAR setting. As explained in the methodology, we have created four different sentiment index time series using the financial news articles. We have a baseline index which is the average daily sentiment which does not use the clusters. We have an index based on the average sentiment in the biggest cluster on each day. We also have an index which is a weighted average of the clusters sentiments, weighted by the cluster sizes. Furthermore, we have an index which is similarly weighted by size, but uses the next day forecasted cluster sizes. Both weighted indices give a greater weight to bigger clusters.

For modeling time series with a VAR model the time series need to be stationary. As explained in the methodology we use the Augmented Dickey-Fuller test with alternative hypothesis of stationarity to test for this. Table 6 shows the outcomes of this test for each of the sentiment indices, the part of the S&P 500 we use and the first differenced version of the S&P 500.

	S&P 500	S&P 500 FD	BL	BC	CS	FCS	VIX	EPU	DNS
p-value	**0.11**	0.00	0.04	0.00	0.00	0.00	0.05	0.00	0.00

Table 6: p-values of the ADF test. Accepting null hypothesis in bold.

The table shows that all sentiment indices reject the null hypothesis, suggesting stationarity. Regarding the S&P 500 index, we see that for the price dataset we cannot reject the null hypothesis of a unit root, meaning that it is not stationary. For the first differenced version of the index we can reject the null hypothesis. Since the VAR model requires that the time series are stationary we use the first differenced version of the S&P 500.

Following, we use each sentiment index to make predictions on the first differenced S&P 500 using rolling window regressions with the S&P 500 and the sentiment indices as variables in a VAR model we make one day ahead predictions. We have tried different training period lengths. Due to computational limitations, running the clustering on all five months takes more than a month. We only consider the clustering based sentiment indices which are created with base intensity, $\lambda_0 = 0.1$. To evaluate the predictive performance we convert each one day ahead price prediction into a binary value which equals 1 if the price prediction had the correct sign compared to the first differenced S&P 500 value. Then, we are left with a binary vector, of which we take the average to compute the ratio of correct one day ahead movement prediction. We call this ratio the prediction accuracy.

Tables 7, 8, 9 & 10, shows the resulting prediction accuracies for each sentiment index and training window period for lags of 1, 2, 3 and 4 respectively.

Sentiment index \ Training weeks	1	2	3	4	5	6
Base	0.45	0.38	0.45	0.51	0.44	0.49
Biggest Cluster	0.52	0.46	0.41	0.45	0.5	0.49
Weighted	0.51	0.47	0.38	0.48	0.47	0.49
Kernel forecasting	0.47	0.4	0.38	0.45	0.47	0.52
VIX	0.44	0.44	0.43	0.44	**0.59**	0.51
EPU	0.48	0.47	0.48	0.44	0.46	0.44
DNS	0.46	0.4	0.39	0.44	0.4	0.44

Table 7: S&P 500 fractions of correct next day price movement predictions with lag = 1. Best accuracy in bold.

In Table 7 we see the results for a lag of 1 in the VAR model. Looking at the resulting prediction accuracies we can see that the highest prediction accuracy is 0.59, and is reached with the VIX index with a training period of 5 weeks. Furthermore, we can see that this accuracy is the only one which convincingly passes 0.5, the rest of the accuracies where well below 0.5 or just about 0.5.

Sentiment index \ Training weeks	1	2	3	4	5	6
Base	0.44	0.39	0.41	0.45	0.54	**0.56**
Biggest Cluster	0.45	0.44	0.49	0.44	0.43	0.51
Weighted	0.51	0.43	0.45	0.48	0.49	0.49
Kernel forecasting	0.48	0.53	0.39	0.44	0.49	0.51
VIX	0.39	0.48	0.45	0.43	0.49	0.49
EPU	0.49	0.39	0.49	0.41	0.43	0.46
DNS	0.54	0.4	0.39	0.45	0.51	0.49

Table 8: S&P 500 fractions of correct next day price movement predictions with lag = 2. Best accuracy in bold.

In Table 8 we see the results for a lag of 2 in the VAR model. For this lag the best prediction accuracy was 0.56 and reached with our baseline index. There is once again not one index which always outperforms the other ones.

Sentiment index \ Training weeks	1	2	3	4	5	6
Base	0.43	0.49	0.41	0.35	0.56	0.52
Biggest Cluster	0.43	0.46	0.43	0.44	0.44	0.48
Weighted	0.43	0.54	0.46	0.43	0.44	0.49
Kernel forecasting	0.43	0.55	0.44	0.47	0.54	**0.59**
VIX	0.43	0.54	0.49	0.32	0.47	0.52
EPU	0.42	0.51	0.52	0.44	0.4	0.44
DNS	0.43	0.49	0.43	0.43	0.43	0.52

Table 9: S&P 500 fractions of correct next day price movement predictions with lag = 3. Best accuracy in bold.

In Table 9 we see the results for a lag of 3 in the VAR model. This time around it does seem that the kernel forecasted score slightly outperforms the other indices. But this does not happen for each training week period. The kernel forecasted score also reaches the highest accuracy, namely 0.59.

Sentiment index \ Training weeks	1	2	3	4	5	6
Base	0.44	0.4	0.48	0.43	0.5	0.54
Biggest Cluster	0.44	0.48	0.5	0.55	0.57	0.56
Weighted	0.44	0.46	0.59	0.61	**0.62**	0.61
Kernel forecasting	0.44	0.48	0.46	0.47	0.53	0.61
VIX	0.45	0.49	0.46	0.39	0.41	0.49
EPU	0.49	0.48	0.46	0.48	0.46	0.51
DNS	0.44	0.49	0.48	0.4	0.35	0.41

Table 10: S&P 500 fractions of correct next day price movement predictions with lag = 4. Best accuracy in bold.

The best prediction accuracy in Table 10 is reached for the cluster size weighted index, namely 0.62, on a training period of 5 weeks. Furthermore, we can see that here as well prediction accuracies look very similar for each sentiment index, fluctuating around ,and mostly under 0.5, depending on the amount of training weeks and the chosen lag.

5 Conclusion

This paper first of all focuses on clustering financial news articles with the Dirichlet-Hawkes process. Our data set contains around 300000 articles which contain their textual content and the date and time of publishing. The articles were published in the first five months of 2018. Due to computational limitations we have only used articles published within the first month of 2018 to evaluate the Dirichlet-Hawkes process. In our evaluation we have considered multiple values for the base intensity, which has a lot of influence on the amount of clusters which are created and the quality of those clusters. We have found that the base intensity parameter has a lot of influence on how the clustering results look. A relatively low base intensity results in very few clusters with very skewed document counts, of the few clusters that are created a subset contains almost all documents. A relatively high base intensity resulted in a lot of clusters which contained more similar topics between each other. Giving a definitive answer to which base intensity is best suited for our data set is hard to decide. But looking at the way we construct our weighted sentiment indices, we decided to use a base intensity which was in the middle of both extremes. Furthermore, we can answer our first research question: ”Is the Dirichlet-Hawkes process suitable for clustering financial news?” positively, because we did find clusters which had similarities within the words contained in the cluster and dissimilarities were also found between the individual clusters. We also discovered that the bigger clusters created with the DHP have a relatively high average pairwise cosine similarity, which led us to believe that these large clusters often contain very similar articles. A reason for this can be the repetitive word use which is found in financial news articles, which leads to very similar clusters, where a lot of articles can fit in.

Furthermore, we have created multiple different sentiment indices with using the financial news articles. The first index is the base index which does not exploit the cluster made with the DHP. Next to that there are three indices which do exploit the clusters. With these articles we made one day ahead forecasts with multiple rolling window sizes in a VAR model setting. Then, we evaluated if the one day ahead forecasts correctly predicted whether the price of the S&P 500 would rise or fall. Research questions 2 & 3 can be answered with the results of the predictions. Research question 2 was: ”Can S&P 500 prediction be improved by incorporating DHP cluster size into the sentiment index?”. To answer this question we consider two sentiment indices, the biggest cluster index and the weighted index. One of these two indices outperformed the base intensity for some lags and training week period, but not for all. So, we can confirm that exploiting topic clusters made with a Dirichlet-Hawkes process can, in some cases, improve the next day ahead predictions of the movement of the S&P 500. The last research question was: ”Can S&P 500 prediction be improved by incorporating DHP cluster size and dynamics into the sentiment index?” To answer this research question we look at the sentiment index which utilizes the ability to forecast new arrivals for the clustered topics, based on the learned topic dynamic. This is the kernel forecasted sentiment index. When looking at the resulting accuracies of this index we can give a similar conclusion, for some training periods and lags the kernel forecasted index outperforms the other ones, but not always. So incorporating DHP cluster size and dynamics can, in some cases, improve prediction accuracies of next day S&P 500 sign predictions in a VAR setting.

References

Aalen, O., Borgan, O., and Gjessing, H. (2008). *Survival And Event History Analysis: A Process Point of View*. Springer.

Baker, S. R., Bloom, N., and Davis, S. J. (2016). Measuring Economic Policy Uncertainty*. *The Quarterly Journal of Economics*, 131(4):1593–1636.

Bollen, J., Mao, H., and Zeng, X. (2011). Twitter Mood Predicts the Stock Market. *Journal of Computational Science*, 2(1):1–8.

Buckman, S. R., Shapiro, A. H., Sudhof, M., Wilson, D. J., et al. (2020). News Sentiment in the Time of COVID-19. *FRBSF Economic Letter*, 8:1–05.

Butler, K. C. and Malaikah, S. (1992). Efficiency and Inefficiency in Thinly Traded Stock Markets: Kuwait and Saudi Arabia. *Journal of Banking Finance*, 16(1):197–210.

Cowles 3rd, A. (1933). Can Stock Market Forecasters Forecast? *Econometrica*, 1(3):309.

Du, N., Farajtabar, M., Ahmed, A., Smola, A. J., and Song, L. (2015). Dirichlet-Hawkes Processes with Applications to Clustering Continuous-Time Document Streams. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 219–228.

Fama, E. F. (1995). Random Walks in Stock Market Prices. *Financial Analysts Journal*, 51(1):75–80.

Ferguson, T. S. (1973). A Bayesian Analysis of Some Nonparametric Problems. *Annals of Statistics*, 1(2):209–230.

Fuller, W. A. (2009). *Introduction to Statistical Time Series*, volume 428. John Wiley & Sons.

Hawkes, A. G. (1971a). Point Spectra of Some Mutually Exciting Point Processes. *Journal of the Royal Statistical Society: Series B (Methodological)*, 33(3):438–443.

Hawkes, A. G. (1971b). Spectra of Some Self-exciting and Mutually Exciting Point Processes. *Biometrika*, 58(1):83–90.

Hu, M. and Liu, B. (2004). Mining Opinion Features in Customer Reviews. In *AAAI*, pages 755–760.

Jeet.J (2018). Us Financial News Articles. https://www.kaggle.com/jeet2016/us-financial-news-articles.

JFChi (2018). Dirichlet-Hawkes GitHub Repository. https://github.com/JFChi/Dirichlet-Hawkes-Process.

Kavussanos, M. and Dockery, E. (2001). A Multivariate Test for Stock Market Efficiency: The Case of ASE. *Applied Financial Economics*, 11:573–79.

Malkiel, B. G. (1973). *A Random Walk Down Wall Street*. Norton, New York.

Malkiel, B. G. (2003). The Efficient Market Hypothesis and its Critics. *Journal of Economic Perspectives*, 17(1):59–82.

Nassirtoussi, A. K., Aghabozorgi, S., Wah, T. Y., and Ngo, D. C. L. (2014). Text Mining for Market Prediction: A Systematic Review. *Expert Systems with Applications*, 41(16):7653–7670.

Nofsinger, J. R. (2005). Social Mood and Financial Economics. *The Journal of Behavioral Finance*, 6(3):144–160.

Prechter Jr, R. R. and Parker, W. D. (2007). The Financial/Economic Dichotomy in Social Behavioral Dynamics: the Socionomic Perspective. *The Journal of Behavioral Finance*, 8(2):84–108.

Schumaker, R. P., Zhang, Y., Huang, C.-N., and Chen, H. (2012). Evaluating Sentiment in Financial News Articles. *Decision Support Systems*, 53(3):458 – 464.

Si, J., Mukherjee, A., Liu, B., Li, Q., Li, H., and Deng, X. (2013). Exploiting Topic Based Twitter Sentiment for Stock Prediction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 24–29.

Smith, V. L. (2003). Constructivist and Ecological Rationality in Economics. *American Economic Review*, 93(3):465–508.