Using Sentiment from Twitter optimized by Genetic Algorithms to Predict the Stock Market

Carlos Vieira Simões

Thesis to obtain the Master of Science Degree in

Electical and Computer Engineering

Supervisors: Prof. Rui Fuentecilla Maria Ferreira Neves

Prof. Nuno Cavaco Gomes Horta

Examination Committee

Chairperson: Prof. António Manuel Raminhos Cordeiro Grilo

Supervisor: Prof. Rui Fuentecilla Maria Ferreira Neves

Member of the Committee: Prof. João Miguel Duarte Ascenso

April 2017

Resumo

Actualmente, na rede social Twitter, são criados, em media, cerca de 6000 *tweets* por segundo. Isto corresponde a cerca de 500 milhões de *tweets* por dia. Com toda esta informação torna-se interessante saber que partido se pode tirar dela. Neste trabalho propomos usar o Twiter para encontrar empresas que tenham um grande potencial de crescimento e que, por isso, sejam boas oportunidades de investimento.

Para chegar a esse fim nós construímos um modelo de sentimento usando o texto contido em tweets. Para garantir que o nosso modelo de sentimento contém tweets que foram criados em contextos emocionais diferentes utilizámos *hashtags* do Twitter. Por exemplo, usámos as *hashtags* #sad e #happy para obter *tweets* que foram criados em ambientes emocionais distintos, ou seja, triste e feliz. Para nos certificarmos de que obtivemos *tweets* de um vasto leque de sentimentos usámos com permissão as emoções apresentadas pelo *Circumplex Model of Affect* e os seus sinónimos. São essas palavras que foram utilizadas como termos de pesquisa na *API* do Twitter. As emoções foram agrupadas em quatro classes. Por exemplo, emoções como feliz e orgulhoso foram incluídas na classe 1; calmo e sereno na classe 2; nervoso e ansioso na classe 3 e; triste e deprimido na classe 4. Posto isto, com o texto dos *tweets* recolhidos com as diferentes emoções construímos uma *document-term matrix* com *unigram features.* Pontos de interrogação, pontos de exclamação e *emoticons* também foram incluídos no conjunto das *features*. Para construir o modelo de sentimento usámos uma máquina de vetores de suporte que consegue prever a qual das quatro classes um determinado tweet pertence com uma precisão de 63.75%.

Para além de *tweets* com contexto emocional também fizemos o download de *tweets* relacionados com empresas. Esses *tweets* foram recolhidos usando o *cashtag* da empresa como parâmetro de pesquisa no Twitter (O *cashtag* de uma empresa é o seu ticker precedido pelo símbolo do dólar, e.g. $AAPL para a empresa Apple Inc.). Somente *tweets* submetidos enquanto o mercado bolsista esteve fechado foram considerados. Incluímos nas nossas simulações todas as empresas que compõem o Standard & Poor's 500 (S&P 500), NASDAQ 100 e Dow Jones Industrial Average (DJIA). Com os tweets que referem essas empresas e que foram submetidos enquanto o mercado esteve fechado previmos o respectivo contexto emocional usando o modelo de sentimento construído anteriormente.

Com as medidas de sentimento para as diferentes empresas criámos uma estratégia de investimento que foi optimizada por um algoritmo genético por forma a maximizar o lucro.

As nossas simulações demonstram que é possível criar uma estratégia de investimento lucrativa usando o sentimento recolhido no Twitter. Durante o nosso período de teste (7 de Novembro a 16 de Dezembro de 2016) conseguimos um retorno de investimento de 11% que é superior a qualquer um dos retornos atingidos pelo S&P 500, NASDAQ 100 e DJIA.

Palavras chave: análise de sentimento, twitter, recolha de opinião, mercado bolsista.

Abstract

At the time of writing around 6000 tweets are posted every second on the social network Twitter. This corresponds to about 500 million tweets a day. With access to that data it becomes interesting to study what we can learn from it. In this work we propose to use Twitter to find companies with a good growth potential that could be good investment options.

In order to achieve this we built a sentiment model using the text content of tweets. We make use of hashtags to collect Twitter posts from a broad range of emotions so that our sentiment model can reliably distinguish tweets containing different sentiment expressions. So, for example, we use the hastags #sad and #happy to collect with a sad and happy emotion respectively. To guarantee that no human sentiment is left behind we adopted emotions from the Circumplex Model of Affect (used with permission) and their synonyms and used them as search terms on the Twitter API. These emotions were grouped in four classes. As an example, happy and excited belong to class 1; calm and peaceful belong to class 2; mad and furious belong to class 3 and; sad and gloomy belong to class 4. Then, with the tweets collected using the emotion words, we created a document-term matrix with unigram features. Question marks, exclamation points and emoticons were also included in the feature set. Afterwards, we use a support vector machine classifier to build a sentiment model. This model achieves 63.75% accuracy in predicting to which of the four classes a tweet belongs to.

Alongside sentiment tweets we also collected company related tweets. Company tweets were collected using their cashtags (the company's ticker with a dollar sign in front of it, e.g.,$AAPL for Apple Inc.) as search terms and only the ones posted during off trading hours were considered. The companies whose related tweets were included in the simulations resulted from the intersection of companies present in the Standard & Poor's 500 (S&P 500), National Association of Securities Dealers Automated Quotations 100 (NASDAQ 100) and Dow Jones Industrial Average (DJIA). With those tweets, posted between market close time and market open time, we used the sentiment model to predict the predominant sentiment in them.

With the sentiment measures of tweets from different companies we created a trading rule that was optimized by a Genetic Algorithm so that we can maximize profit.

Our simulations show that it is possible to build a profitable strategy for trading in the stock market using Twitter with the rules we implemented. During our testing period (November 7, 2016 to December 16, 2016) we achieved an 11% return, outperforming the S&P 500, NASDAQ 100 and DJIA composites.

Key words: sentiment analysis, twitter, opinion mining, stock market.

Acknowledgements

I would like to express my gratitude to my supervisors, professors Rui Neves and Nuno Horta, for their support, patience, advice and feedback. Especially to my advisor Rui Neves, for the invaluable transmitted knowledge on the financial market domain, for his constant presence and availability, without which it would not be possible to conclude this work.
Last but not least, I would like to thank my family, friends and colleagues for supporting me throughout writing this thesis.

Table of Contents

List of Tables

List of Figures

List of Acronyms and Abbreviations

API – Application Programming Interface

ASX – Australian Securities Exchange

CEO – Chief executive officer

CNB – Complement Naive Bayes

CSV – Comma Separated Values

DAL – Dictionary od Affect in Language

DJIA – Dow Jones Industrial Average

DNA – Deoxyribonucleic Acid

DPM – Dirichlet Process Mixture

EDT – Eastern Daylight Time

EMH – Efficient Market Hypothesis

EST – Eastern Standard Time

GA – Genetic Algorithm

GPOMS – Google-Profile of Mood States

HTML – HyperText Markup Language

IEEE – Institute of Electrical and Electronics Engineers

IPO – Initial Public Offering

LDA – Latent Dirichlet Allocation

MA – Moving Average

ModVaR – Modified Value at Risk

NA – Not Available

NASDAQ – National Association of Securities Dealers Automated Quotations

NLP – Natural Language Processing

NLTK – Natural Language Toolkit

NYSE – New York Stock Exchange

POMS – Profile Of Mood States

ROI – Return On Investment

SMART – System for the Mechanical Analysis and Retrieval of Text

SOFNN – Self-Organizing Fuzzy Neural Networks

S&P 500 – Standard and Poor's 500

SVM – Support Vector Machine

TF – Term Frequency

TF-IDF – Term Frequency – Inverse Document Frequency

URL – Uniform Resource Locator

UTC – Coordinated Universal Time

VaR – Value at Risk

VIX – Chicago Board Options Exchange Volatility Index

Chapter 1. Introduction

The stock market is the congregation of buyers and sellers of stocks. Shares or stocks represent ownership in a company so, if we own 10% of the shares of a company it means we own 10 % of the company. The stock market is very important for companies and businesses because it allows them to finance their activities by selling small portions of themselves to the public. That way, a company can easily and rapidly obtain liquidity that can be used to grow production, or increase market share, or enter a new market, amongst others, which in turn, hopefully, increases their profit that can used to buy back the shares sold before. Of course, this also brings opportunities to the public. If one buys shares of a company and, in turn, that company is able to grow their sales and turnover, the shares the investor initially bought are going to be worth more because the underlying company is also worth more. Indeed there are many people involved in buying and selling stocks hoping to profit from it. These include small individual investors and large investors such as banks, hedge funds, pension funds or insurance companies. Some of them have accumulated billions over the years doing just that, buying and selling stocks. What all this participants have in common is that they are constantly looking for an edge and that is where Twitter can come in. Nowadays almost every news organization and every journalist has live twitter feeds, not only that but most companies have an extensive social media presence including their top managers and CEOs.

The number of social media network users has drastically increased in the last decade and continues to grow every day. This is also true for stock market participants. This means that large amounts of credible information can be spread out to thousands of users (followers) who can spread it themselves. With this enormous amount of data available it is necessary to examine its potential for financial forecasting. Although computational power, storage capacity, and bandwidth limits are still big obstacles, these limits are smaller than they were a while back and eventually they will pose an even smaller hurdle.

Some large hedge funds are spending millions on building computer software able to generate trade signals based upon data from social media[*]. Also there are several companies that provide sentiment analysis of social media for financial institutions (we will mention a few on the next chapter). Indeed, if useful economic data is available on twitter, then using sentiment analysis or opinion mining, it can be possible to merge that functionality with the already existing automated trading algorithms. Most of these algorithms use real-time data that comes from the stock market exchanges, but if they were also able to access and interpret all the information available, including news feeds and public conversations on social media, then they would be able to react instantly and make sound rational trading decisions.

Gathering sentiment from Twitter to predict the stock market has been the subject of previous studies. Some of them reach high accuracy rates in predicting if prices are going up or down. Bollen et al. [1] used more than 9 million tweets to claim an 87.6% accuracy rate predicting the up and down

[*] http://www.nytimes.com/2010/12/23/business/23trading.html?_r=0

movement of the DJIA index. Similarly, Mittal and Goel [2] claimed a 75.5% accuracy rate doing this, with more than 476 million tweets collected.
Xu and Keelj [3] built a system that classified the sentiment on a tweet as positive, negative or neutral in order to predict the price movement direction of stocks. They achieved an accuracy of 58.9%.
Si et al. [4] schemed a setup that used an unsupervised learning technique to group Twitter messages into topics by analyzing their text content. Then, with a sentiment dictionary, a daily sentiment score was calculated for each topic and those time series are used to predict the price of the Standard & Poor's 100. They claim 61% accuracy at predicting price movements.

Also, Porshnev et al. [5] identified emotional states of Twitter users in order to predict the future prices of the Standard & Poor's 500 and Dow Jones Industrial Average. Best achieved accuracy was 64.10%.

A more detailed explanation of these studies and others is presented on chapter 2.

1.1. Work Purpose

We propose to use tweets, posted on the social Network Twitter, to predict the future price movements of individual stocks listed on the S&P 500, NASDAQ 100 and DJIA Therefore in this work tweets with explicit sentiment were used to classify company related tweets. Tweets with explicit sentiment are identified by their hashtags and, afterwards, are used to build a sentiment model that classifies company's tweets. Then, with the sentiment of the tweets of each company we use a GA to create a trading rule that maximizes profit. The profit of our simulations will be compared to a buy and hold strategy.

1.2. Main contributions

The main contributions of this work are:

- A collection of procedures that filter out tweets without genuine emotive content;
- Selection of tweets using their retweet count and favorite count;
- The use of the Circumplex Model of Affect to help build a list that encompasses a broad range of human emotions;
- A sentiment model built with tweets collected using emotion words, in hashtag form, as search term;
- Application of the sentiment model on company related tweets;
- Procedures that filter out most of the collected company related twits (spam) and keep the ones that help understand the true market sentiment;
- A set of rules that uses sentiment to choose which companies to buy.

1.3. Document Structure

Besides the present first chapter that provides some context and presents the main contributions of this work this theses also contains:

Chapter 2 – State of the art and related work – Contains theory and key concepts necessary to understand not only the procedures proposed in this work but also similar work we talk about at the end of the chapter.

Chapter 3 – Architecture of the proposed solution – Presents all the procedures that were performed to build the proposed solution.

Chapter 4 – Proposed solution performance and case studies – Shows a few practical applications of the designed system on the stock market and describes their performance.

Chapter 5 – Conclusions and future work – Mentions limitations of the proposed solution and possible improvements and summarizes the effectiveness of the results obtained.

Lastly, at the end of the document, we list references and publications consulted during this work.

Chapter 2. Related Work

In this chapter we discuss some literature review related to our work. This chapter is divided in five sections. The first refers the theories concerning market analysis and the role Twitter can have to this purpose. Sections two and three provide some context and a few basic concepts and useful tools that are essential to understand sentiment analysis. The fourth section concerns sentiment analysis applications and it is divided into two subsections. In the first we present research on sentiment analysis applications that are not stock market related. In the second we present research cases specifically related with stock market analysis using sentiment analysis. The last section is a short conclusion of the chapter.

2.1. Market Analysis

There are several theories regarding the predictability of a stock price, chief among these is the Efficient Market Hypothesis (EMH) [6]. This theory states that it is impossible to profit from trying to "beat the market" because the actual stock prices already reflect all the information available and everybody involved has access to it, that is, the stock market is efficient and reacts instantly to new information by adjusting the share price. Another theory with similar marks is the Random Walk Theory [7], which suggests that, in the short-term, share prices move randomly and thus, consistently outperforming the market is impossible. The mainstream idea is that the random walk theory is explained by the EMH[*].

As consequence of these two theories two main philosophies of future price analysis have emerged: the fundamental and the technical analysis. A fundamentalist trader pays special attention to the financial numbers and ratios describing the economics of the quoted company like the P/E ratio or earnings per share, taking inflation and other external factors into consideration.

Technical traders, on the other hand, consider that price movements are not totally random and that patterns may be found. To achieve such stage, a time series of the past prices and a myriad of technical indicators are used to help predict the future prices. Moreover, there are those who argue that the success of technical analysis is largely due to a self-fulfilling prophecy, that is, if a very large number of people in the market believe and act the same way, then, perditions can easily become true.

With the proliferation of the social networks, a new way to perform market analysis has arisen. This new way relies on the sentiment collected from vast amount of texts posted on sites, such as Twitter to gauge the public mood and access if stock prices are going to rise or not.

[*] http://thismatter.com/

2.2. Twitter

Twitter is a social networking service, launched in 2006, that allows users to exchange short messages. It has grown exponentially since its inception and its users come from many different areas and backgrounds. Bill Gates, Barack Obama, Katy Perry, Twitter, Instagram, CNN, and Google are amongst the twitter accounts with the most followers[*].

Twitter has well over 1 000 million registered users. Throughout 2015 the network managed to average more than 300 million monthly and 100 million daily active users[†], a 10 fold increase since the first quarter of 2010.

Twitter is free and works on multiple platforms. It allows members to broadcast short massages called tweets. Unlike Facebook or LinkedIn Twitter members do not need to approve social connections, anyone can follow anyone. Users can keep track of a certain topic by using hash-tags, i.e. a cardinal immediately followed by the conversation theme, for example, #football. Also members can mention other members on conversations with the 'at' symbol, for example, @username. The default settings for Twitter are public, which means that anyone can search tweets on the network, even if he is not a member.

The advantage of twitter comes from its unique architecture. Each user chooses whom to follow and instead of pursuing certain topics like in forums and news sites, information is pushed to the user. This means that the content is moderated automatically. Users that provide helpful information get more attention drawn to them and users who produce useless information are easily forgotten and ignored. Also, major news organizations, such as Reuters and Bloomberg, have news feeds and broadcast them to subscribers and to third party services on twitter who help transmit the information to the end user through Twitter.

Twitter also verifies some of its users accounts. These accounts are marked by a blue badge on their profile page. A verified account means that Twitter has verified the true identity of the account holder by other means than just the nickname. Not all accounts are verified and, according to their website[‡], an account may be verified “if it is determined to be an account of public interest. Typically this includes accounts maintained by users in music, acting, fashion, government, politics, religion, journalism, media, sports, business, and other key interest areas.”

A tweet is a short text message containing a maximum of 140 characters plus some additional metadata such as creation date and time, author username, tweet ID, number of retweets, number of followers and language. On the social network, a tweet can be retweeted if a user, other than the original author, chooses to relay that tweet to his followers. A user's followers are the accounts that subscribe to the tweets posted by that user.

[*] Twittercounter - http://twittercounter.com/pages/100
[†] Statista - http://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/
[‡] Twitter - https://support.twitter.com/groups/31-twitter-basics/topics/111-features/articles/119135-about-verified-accounts

Alongside the already spoken hashtags and mentions, or at signs, twitter supports cashtags. These are used to identify a company's stock symbol. It is the dollar sign followed by the sock ticker as used in the stock market, e.g. $AAPL for the Apple Inc. company.

On Twitter a user can click on a username to see that user's profile page with the most recent posted tweets. To see the most recent tweets mentioning a hashtag or cashtag one just has to click on them or search them in the search box.

2.2.1. Twitter API services

An Application Programming Interface, or API, is a set of routines and protocols that allows developers to interact with the technology. More specifically, Twitter created their open API so that external software or applications could automatically access Twitter data and create new technology services with it.

The offering of a public API promotes innovation and the creation of new platforms, products and interfaces by developers outside the Twitter. The social network has capitalized on this by acquiring companies that, to some extent, were built to explore the twitter API. A few examples of such companies are Posterous, Summify, TweetDeck.

i) Twitter's Search API

The Twitter's search API allows access to tweets that have already occurred. One uses a keyword, username, hashtag, or other information, to query tweets matching those parameters and the results are returned on the spot, much like a user using a browser and navigating to the Twitter web page and performing a search there. Presently, 180 requests can be done in 15 minutes[*].

ii) Twitter's Streaming API

The Twitter streaming API works just as above, using a keyword, username, hashtag, or others, to query tweets, but things happen in real time, that is, only tweets that are being posted and match the search criteria are returned. This requires the developer to maintain an open connection with Twitter so that if someone posts a tweet that meets the search parameters it gets immediately pushed to the developer. Not all data, that matches the search parameters, gets to the developer though. The Twitter Streaming API documentation promises 1% of all data although Morstatter et al. [8] found that in their 28 day test period the number was 43.5%. Furthermore, they found that when twitter activity spiked, the amount of data returned by the Twitter Streaming API did not. Thus they aimed to investigate whether that affects the quality of the measures and metrics performed on the data or not.

[*] Twitter - https://dev.twitter.com/rest/public/rate-limiting

iii) Twitter's Firehose

Twitter Firehose is similar to the streaming API except that it guarantees 100% delivery of tweets that match the search parameters. At first Twitter delegated their firehose handling to third party companies, GNIP and DataSift. Access to that data was, and still is, very costly. Moreover, besides having to pay for the firehose access, one also has to consider the cost of retaining the firehose data, which requires servers, disk space and network bandwidth. Twitter, a company which has yet to make a profit, was not always aware of their firehose value. Apparently Twitter is exploring a new business model by selling access to their firehose directly to customers rather than transferring those services to Gnip and DataSift. Biggest evidence for this is that Twitter acquired Gnip for $134 million in 2014[*] and ended the partnership with DataSift on August 13, 2015[†]. This attempt to highly restrict firehose access has even resulted in legal action. In 2012, the tech company PeopleBrowsr sued Twitter for shutting down their access to the firehose. The legal case ended in an out of court agreement established in 2013[‡].

2.2.2. The Power of Twitter over the stock market

Two main points are brought up when investigating the influence of twitter on the stock market. One is that the information posted on Twitter is fresh and the stock prices, now, do not yet incorporate that new information but are expected to in the near future. Two, is that the value of a company in the stock market, not only is the value of the underlying business, but fluctuates with waves of optimism and pessimism that overflow the investors. Twitter can be a way to measure these sentiments.

There have been several demonstrations Twitter power over the stock market. On March 30, 2015, Elon Musk, CEO of Tesla, tweeted “Major new Tesla product line -- not a car -- will be unveiled at our Hawthorne Design Studio on Thurs 8pm, April 30”. This soared the company value on the stock market by approximately one billion US$, in just a few minutes. On a similar note, investor Carl Icahn, on August 13, 2013, tweeted “We currently have a large position in APPLE. We believe the company to be extremely undervalued. Spoke to Tim Cook today. More to come”. This boosted capitalization by more than ten billion US$. On April 23, 2013, a twitter account of the Association Press, an American news organization, got hacked and hackers posted “Breaking: Two Explosions in the White House and Barack Obama is injured”. In the next three minutes the S&P index fell nearly 1%, that means that investors took more than 130 billion US$ from the market in that time period[§].

The importance of Twitter is also demonstrated by the fact that companies, themselves, are starting to

[*] The Wall Street Journal - http://blogs.wsj.com/digits/2014/08/11/twitter-paid-134-million-for-data-partner-gnip/

[†] DataSift - http://blog.datasift.com/2015/04/11/twitter-ends-its-partnership-with-datasift-firehose-access-expires-on-august-13-2015/

[‡] Techcrunch - https://techcrunch.com/2013/04/25/twitter-settles-with-peoplebrowsr-gives-the-company-firehose-access-until-the-end-of-the-year/

[§] Telegraph - http://www.telegraph.co.uk/finance/markets/10013768/Bogus-AP-tweet-about-explosion-at-the-White-House-wipes-billions-off-US-markets.html

publish share price altering information, like earnings and reports, on twitter, even though, that information might be available elsewhere. On July 2012, Netflix's CEO Reed Hastings shared a post on Facebook claiming that, in June, the company had surpassed 1 billion monthly viewing hours for the first time in history. After that, the company's share price jumped sharply and the incident raised an important issue. The question was if this constituted a violation of disclosure laws when companies or their executives shared sensitive information on social media. At the time, the securities and exchange commission, SEC, considered bringing charges against Hastings but after months of investigation they decided not to peruse an enforcement action and that it was admissible for companies and their executives to use social media to, with some restrictions, disseminate corporate related information.

Investors are already using data from Twitter to help them pick better trades. StockTwits[*] is a pioneer example of this. StockTwits "listens" to the stock market related chatter and gauges if the market is bearish or bullish. This is called big data analysis - to comb through a massive amount of data to extract useful information. The main obstacle of big data analysis is, of course, the computational power, which needs to be immense. As processing capabilities roughly double every two years, it will eventually be possible to perform sentiment analysis on live data and instantly gauge the market sentiment.

Sentiment analysis on Twitter data might however be very different according to the total market capitalization of the company in question. In fact, a study concludes that Twitter sentiment has stronger effects on small cap companies [9]. This conclusion can have more than one cause. Firstly, large companies are widely discussed and thus the data related to them is very noisy and a rigorous analysis is very difficult. Secondly, these companies are widely discussed and thus a rigorous analysis is already made by many which makes their prices too efficiently priced. Also small companies have less coverage by stock market analysts and the chance that their prices are efficiently priced is smaller, thus they have more room to fluctuations.

2.3. Sentiment Analysis

Opinion mining or sentiment analysis is the computational interpretation of opinions, sentiments, affects, subjectivity, views or emotions, present in texts. Opinions are important influencers of our behavior. Whenever, either individuals or companies need to make an important decision they seek out the opinions of others.

The subject of sentiment analysis encompasses many different tasks, such as, opinion mining, opinion extraction, sentiment mining, subjectivity analysis, affect analysis, emotion analysis, review mining and others. However, now, they can be referred to as sentiment analysis or opinion mining [10].

Recently, there has been a significant increase in the number of studies published about public opinion and sentiment analysis. There are several reasons for this. One is the increasing use of

[*] http://stocktwits.com/

sentiment analysis for commercial applications, which functions as an allurement for researchers, because their work can result in financial gains. Two, the process of sentiment analysis unveils arduous problems that have never been conquered before, so, the challenge can be enticing. Three, thanks to the Internet and the proliferation of social media sites, there is a large volume opinionated data online and gaining access to it is, sometimes, quite easy. Obviously, research on this matter without this data would have been far more difficult. Lastly, the growth of computational power means that now a huge number of texts can be processed in a fraction of time it would take ten years ago. That increased computational ability can be used not only for preprocessing techniques in sentiment analysis but also for machine learning algorithms.

2.3.1. Sentiment Analysis Applications

Public opinion can be very important for decision making situations. Whenever someone decides to make a purchase they can go online and find out what people that bought the product think about it. The same can be said for services such as restaurants, stores, coffee shops and others. There are sites solely dedicated to user's reviews, for instance, Yelp where people can post their opinion about almost everything, from education to health to sports and shopping. To further add to this point, online shops, such as Amazon, have dedicated spaces for user's reviews, so, their users can rate the product and everyone that visits the product's page after that can see it and take that opinion into consideration. As a consequence of this, businesses and organizations always wish to ascertain what the consumers think about their services or products. In the past, the way to do that was to conduct surveys or opinion polls or through customer feedback from emails and call centers but now they can use online data on blogs, Twitter, Facebook and reviews sites. This doesn't mean that the information obtained the former way is no longer valid, however the data that can be gathered online is becoming abundant and its interpretation is a business necessity.

Online social media postings can also have a profound political impact. It can influence elections and mobilize mass protests like the ones in some Arab countries in the recent past. A recent literature review focused in a socio-political perspective pointed out the role of Twitter on a range of social movements, such as those in Iran in 2009 and in Egypt in 2011 [11].

As the amount of opinion sites on the web becomes increasingly ample, the extraction of relevant information turns into a biblical task. Having a human monitoring and unraveling all of the information contained in those sites is not practical so, the creation of automated systems to accomplish this is unavoidable. Consequently, business activities involving sentiment analysis have grown rapidly in recent years and many start-ups have been created. Those new companies implement sentiment analysis to a wide range of domains, such as political elections, financial services, healthcare and consumer products and services. Dataminr, Digimind, Infegy Atlas, Mention and Talkwater are just a few examples. Other, already established, companies have internally developed their sentiment analysis capabilities, e.g Microsoft, Apple, Google and HP, in the hopes of making better business decisions.

2.3.2. Common Terms used in Sentiment Analysis

Before digging deeper into sentiment analysis related texts, there are a few basic concepts one needs to be familiarized with to be able to understand the procedures performed on the related research.

Emoticon

Emoticons are used to express a feeling or a person's mood and consist of a text representation of facial expressions, or feelings, with punctuation, numbers and letters. A collection of emoticons is listed on Table 1.

Table 1 - Example of emoticons and their meaning.

Emoticon	Meaning
:-) :) :-] :] :-3 :3 :-> :> 8-) 8)	Happy face
:-D :D 8-D 8D x-D xD X-D XD	Laughing
:-(:(:-c :c :-< :< :-[:[>:(	Frown, sad and angry
:-))	Very happy
:'-(:'(	Crying
:'-) :')	Happiness Tears
:-O :O :-o :o :-0 8-0 >:O	Surprise, shock, yawn
D-': D:< D: D8 D; D= DX	Horror, sadness
;-) ;) *-) *) ;-] ;] ;^) :-, ;D	Smirk

Token

A token is a component of a text. It is going to be used as feature for the classifier. It can be a single word or two or even a question mark or an exclamation point. For example, in the tweet "This bread is stale :(.", the person performing sentiment analysis can choose single word tokens, in which case "This", "bread", "is", and "stale" are the four tokens. Alternatively, he can choose three word tokens, in which case "This bread is" and "bread is stale" are the only two tokens. Also, he can choose punctuation tokens, in which case ":(." is the only token. In many papers, to distinguish normal

punctuation, such as a single dot that marks the end of a sentence, from punctuation with emotional content, such as the emoticon, ":(", used in the tweet above, the emoticon is replaced by text, usually, with similar meaning. So, for sentiment analysis purposes and continuing with the tweet "This bread is stale :(.", the emoticon would be replaced and the text would be "This bread is stale SadEmoticon.". Obviously this would require some kind of emoticon dictionary but the advantage is that now the text version of the emoticon is a single word token and the single word tokens for the tweet become "This", "bread", "is", "stale" and "SadEmoticon" and regular punctuation, like the single dot, can be, usually, erased because they possess very little emotional weight.

Tokenization

Tokenization is the process of splitting a document into tokens. There is no single right way to do tokenization. The right technique is very application dependent. Sentences can be broken up on white spaces or a set of special characters or symbols or punctuation among others. Example tokenization strategies include the Whitespace tokenizer and the Treebank tokenizer.

N-grams

N-gram is the number of words to include in each token. Typically, if we want to split a sentence into n-grams, we take the first n words as a token, move one word forward and take that set of n words as another token. Finally, repeat these steps until the last word of the sentence is reached.

For example, if we want to split the sentence "This bread is stale" into bigrams or 2-grams, they would be "This bread", "bread is" and "is stale", so, the sentence has three bigrams. For a sentence with X words the number of n-grams in it is

$$X – (N – 1),$$

with N equal to the number of words in a single n-gram.

Features

In classification explanatory variables, independent variables or features is the set of input variables that are a measurable characteristic of the phenomenon being observed and, to some variable extent, explain the outcome or the class they belong to. For example, suppose we want to classify tweets as positive or negative and all we will use to decide whether they belong to one class or the other is the sad smiley emoticon, ":(", and the happy smiley emoticon, ":)". In this case, the number of happy smiley emoticons is a feature and so is the number of sad smiley emoticons. The number of happy smiley emoticons is a measurable characteristic of the data, or a feature, that explains the outcome, or the classes positive or negative. The same can be assumed for the number of sad smiley emoticons.

Document

A document in sentiment analysis is a single entity in a corpus. It is where the text to be processed

and classified is. It is a collection of terms that belong to a single class. After the training process the classifier computes the probability of any given document belonging to a certain class. A document can be, for example, a tweet, a review, a blog post, an e-mail, or an article.

Corpus

The corpus is the collection of documents to be used in the sentiment analysis process. It can be a set of tweets, or articles, or reviews, or forum posts.

Supervised/Unsupervised Learning Technique

An unsupervised learning technique is one where no labels are provided to each set of features, that is, the unsupervised learning algorithm tries to separate the data into groups solely based on the features' values. Essentially, they look for similarities between observations in order to, somehow, group them. Those groups are called clusters. On the other hand, a supervised learning algorithm requires the training dataset to be labeled or to have an output. These labels, or classes, belong to a finite set previously, arrived at by a human. K-means and Support Vector Machines are examples of unsupervised and supervised learning algorithms respectively.

Classifier

A classifier is the machine learning algorithm that performs the classification. In this work the classifier "interprets" the tweets and decides to which class they belong to. Before that is possible, they need to learn a model and for that a training dataset (observations whose class membership is known) is needed. After the model is trained, or learned, predictions can be made for new observations, whose class membership is not known. Some classifier algorithms include Naive Bayes. Support Vector Machines and Maximum Entropy.

Classification

Classification is the process to determine to which class or category a new observation belongs to, based on other observations with corresponding known classes or categories. The collection of data points whose category membership is known is called the training dataset and the classifier uses it to learn a model which can then be used for predictions. Classification is an instance of supervised learning. The analogous unsupervised learning technique is called clustering.

Training Dataset

The training dataset is the set of observations whose output or category membership is known. Each observation consists of at least one feature and the output class or number.

Testing Dataset

The testing dataset is the set of observations whose output or category membership we wish to know.

Each observation consists of at least one feature that has to be consistent with the features in the training dataset.

Training

Training refers to the process a learning algorithm must go through in order to be able to decipher to which class an observation belongs to. A set of labeled data, or a training dataset, must be provided so that the classifier can learn associations between features and classes and in the future make accurate predictions on unlabeled data.

Recall and **Precision**

Recall and precision measure a classifier performance. Recall tells us the percentage of relevant instances that were found while precision is the percentage of solutions that are relevant. These are best explained with an example. Assume we have a camera pointing at the sky. The image is fed to an algorithm that tries to discern planes from birds and birds from planes. After some time we know that five planes and some birds crossed the field of view of the camera. Now, the algorithm tells us that the camera saw four planes but, we know that of those four only three were, actually, planes, the other one was a bird. The classifier precision is said to be ¾, the correct number of identifications divided by the total number of identifications while the classifier recall is 3/5, the correct number of identifications divided by the total number of actual planes.

F-score

A classifier's F-score is the harmonic mean between it's precision and recall and is calculated as

$$2\frac{precision \times recall}{precision + recall}$$

Document-term matrix

A document-term matrix is merely a table that tells us how frequently each term, or token, appears in a corpus. It is a very common way of representing text for further computation [12], [13]. In a document-term matrix one has documents as rows, for example, tweets, and term frequencies as columns which are merely the number of times a term appears in that particular tweet. For example, a document-term matrix for the tweets "This cat hates me too much, too much" and "This stupid cat just woke me up" would have as columns the terms present in all tweets without repetitions and, as rows, the tweets IDs, in this case let us assume the ID for the first tweet is 1 and for the second tweet 2 (Table 2).

Table 2 - Document-term matrix for the tweets “This cat hates me too much, too much” and “This stupid cat just woke me up".

	“cat”	“hates”	“just”	“me”	“much”	“stupid”	“This”	“too”	“up”	“woke”
Tweet 1	1	1	0	1	2	0	1	2	0	0
Tweet 2	1	0	1	1	0	1	1	0	1	1

Sparse terms

In text mining, a sparse term is a term or a token that is used very infrequently. A term's sparsity refers to the frequency that it is used, not within each document, but on the entire corpus. Its numerical value can be easily obtained, even more so if we already have a document-term matrix. A term's sparsity is the total number of documents that contain that term divided by the total number of documents, so, if a term has sparsity 1.0 it means that that term appears on all documents, i.e. it has a very small sparsity.

The goal of this text mining technique, i.e. ignoring terms that appear only in very few documents, is to prevent overfitting and help generalization. To explain this further, let’s say we have ten tweets that are divided into two classes, 1 and 2. Now, the word “cat” appears in one and only one of those tweets and it belongs to class 1. If we train a classifier with those tweets it is probably going to create a very strong link between the word “cat” and the class 1 where, in fact, it could be an ungrounded association. If we had ignored the sparse term “cat”, we minimize the risk of classifying every tweet containing the word “cat” as a class 1, just because it has the word “cat” in it, thus helping generalization.

In this work, where we have a very large number of documents containing very short texts, each one being a very distinct from the next, we can assume the there is going to be a large number of very sparse terms.

Stemming

Stemming, in linguistics, is the process of reducing a word to its stem or root. That stem or root doesn't need to be a valid word, as defined in the English dictionary, it just has to map or be a representation of a collection of related words. For example, a stemming algorithm should identify the words “fisher”, “fishing”, “fished”, “fisherman” and replace them with the word “fish”. Stemming the words "argue", "argued", "argues", "arguing", and "argus" results in the stem "argu", which is not a valid word but is a valid stem. This is a common technique that reduces complexity, without any significant loss in entropy, because it performs a dimension reduction (the number of columns of the document-term matrix gets smaller). Word stemming has been subject to many studies. One of the best known

stemming algorithms is the Porter's stemmer [14].

Lemmatization

Lemmatization process, just like stemming, groups related words but, contrary to stemming, it tries to understand the word's context. The reason for this is because the same word can have multiple meanings, for example, on the sentences "My favorite flower is a rose." and "He quickly rose from his seat", the word "rose" has two different meanings and what lemmatization does is, in principle, distinguish the verb from the noun. However, this added precision comes at a cost. Lemmatization is a complicated and a computationally expensive process, thus stemmers are usually preferred.

Part of speech tagging

Part of speech tagging is the process of labeling, or tagging, the words in a sentence according to their part of speech (noun, verb, adverb, adjective). The tagging is based on the word's definition as well as it's relation to adjacent words or context. The Penn Treebank tagger is one of the most commonly used part of speech taggers. Using this algorithm, adverbs will be tagged "RB", adjectives "JJ", singular proper nouns "NNP" and interjections "UH". With 36 different words tags, the Penn Treebank is one of the most finer-grained taggers.

Stop words

The term stop word is used to refer the most commonly used words in a language. Because they are so common, it is assumed they have zero emotional weight and hence are removed. Eliminating words such as "and", "the", "this", "to", "a", "that", "too", "so" and "or" can bring a significant increase in precision in that, more information can be gained considering just the remaining words in the sentence rather than considering all the original words. The adequate stop words list for removal can be very application dependent. There are lists with just a few words and others with hundreds of words.

The reader has most likely dealt with stop words removal because, most online search engines use this procedure alongside other techniques described in this work, for instance, stemming.

Lexicon based versus Statistical based Sentiment Analysis

There are three possible approaches to sentiment analysis. One is the lexicon or knowledge based techniques. Suppose we want to classify tweets with either positive or negative sentiment. Using a lexicon based approach; a list with positive words and a list with negative words are required. To classify each tweet, we can count how many words are in the positive word list and how many are in the negative word list and arrive at a final score of the sentiment. This method doesn't require a training dataset, we just have to compile the word lists. On the other hand, if we choose a statistical based approach to sentiment analysis and using the same classifying problem, we need a training dataset, i.e. tweets labeled positive or negative. Put simply, this dataset will be used, by machine learning models, to learn what words are more likely to be on the text of the positive tweets and what

words are more likely to be on the text of the negative tweets. After that is done, the model can be used to classify new tweets based on the words present in them.

The third and final approach is called an hybrid approach. This technique uses the former two methods simultaneously.

WordNet

WordNet[*] is an English lexical database compiled by Princeton University. Put simply, it is a dictionary were one can find nouns, verbs, adjectives and adverbs and their synonyms grouped by their cognitive meaning. There synonyms that have a similar emotional meaning are called synsets.

2.4. Support Vector Machines

Support Vector Machines (SVM) is a very popular machine learning algorithm [15] [16] [17], [18], [19], [20]. It has been widely used in various domains with multiple applications, such as health care [21], stock market prediction [22] [23], face recognition [24], handwritten character recognition [25] and others. Its popularity comes from its solid mathematical background, high resistance to overfitting and computational efficiency. Furthermore, there are excellent libraries with the SVM algorithm implementation for several programming languages so, one can easily implement the algorithm on a dataset.

A picture in how SVM can function is demonstrated in the following example based in a subset of the Iris data set [26]. Each sample in this subset is composed 3 values: 2 features, petal length and petal width (measured in centimeters) and the class that sample belongs to, which is merely the species of that Iris flower, i.e. either Iris setosa or Iris versicolor. In this subset there are 50 samples of each species or 100 in total. Table 3 shows a little peek into the data.

What the SVM will do is to create a surface that is a boundary that separates the data points based on their feature values. That boundary, called a hyperplane, has to separate the classes as well as possible, that is, data points of one class should be on one side of the surface and the data points of the other class should be on the other side. Figure 1 illustrates the SVM applied to the subset of the Iris data with a linear kernel.

As we can see from the plot, the data is easily separable but it is not always the case. In Figure 1 the hyperplane, or a line in this case, is the boundary between red and blue and the grading scale in tones of blue and red measures the distance of the points to the boundary. The deeper the color the farthest it is from the hyperplane. The SVM algorithm tries to find the maximum margin hyperplane on a linear space, in other words, among the possible hyperplanes the one that presents the maximum distance from the closest data points (this is the margin) is selected and those data points are called support

[*] https://wordnet.princeton.edu/

vectors. The support vectors in the Figure 1 are the two black triangles and the two black circles.

Table 3 - Sample of the Iris data set.

Petal Length	Petal Width	Species
1.5	0.2	Iris Setosa
1.4	0.2	Iris Setosa
4.7	1.4	Iris Versicolor
4.5	1.5	Iris Versicolor

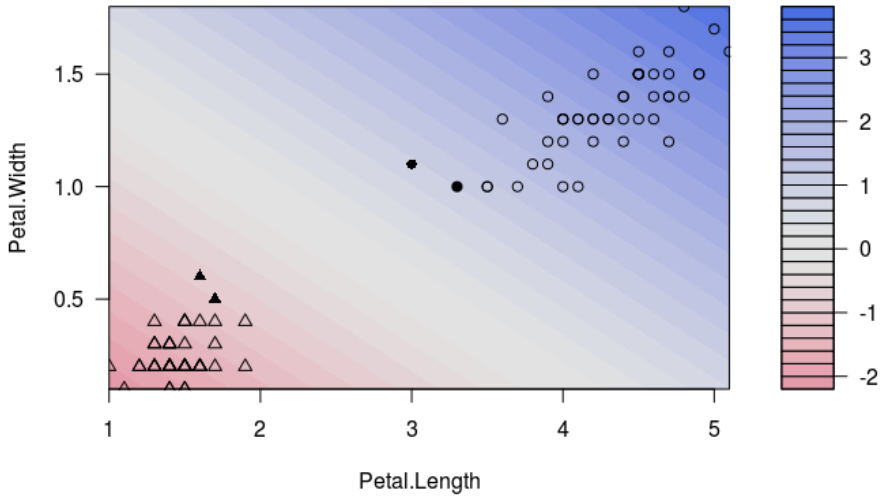

Figure 1 - Application of SVM to the subset Iris data with a linear kernel.

In the case we are analyzing, we have only two features (petal width and length) so, it is easy to visualize the problem in a two dimensional space. Some applications of SVM models may have thousands of features thus, thousands of dimensions, ergo it becomes impossible to visualize a hyperplane that separates the data. That is where different kernel functions come in. Sigmoid, Polynomial and Radial Basis are examples of commonly used kernel functions. Sometimes feature values are not linearly separable so, one can try and use a kernel function in order to map the features values into a new space where the data points become linearly separable. This is called the Kernel Trick [27] [28].

2.5. Genetic Algorithms

Genetic algorithms (GA) are part of the evolutionary computing. Evolutionary computation is a particular field of artificial intelligence and it encompasses a number of algorithms, called evolutionary algorithms that use Darwinian principles to solve problems, that is, they use a natural selection logic where the strongest and fittest individuals from a population are more likely to survive and seed the next generation. Essentially these algorithms emulate biological mechanisms of evolution to find optimal solutions to a given problem. Besides genetic algorithms there are many other techniques in evolutionary computing. Ant colony optimization, artificial bee colony algorithm, artificial immune systems and cultural algorithms are examples of those. Artificial evolution algorithms became widely recognized, as an optimization technique, by the work of Ingo Rechenberg and Hans-Paul Schwefel, in the 1960s, when they used evolution strategies to solve engineering problems. The initial idea was then developed by John Holland and his students. That work resulted in the book the adaptation in natural and artificial systems [29].

On a biological sense, organisms are made up of cells and every cell has the same set of chromosomes. Chromosomes consist of blocks of DNA or genes and each gene encodes a specific protein or a trait, such as eye color, and occupies a particular position in the chromosome. All possible values for the trait are called alleles, that is, green, blue and brown for the eye color trait and the position of a gene in the chromosome is called locus.

For a given problem the set of all possible solutions is called search space, or state space. Every point in the search space represents a solution to the problem and has its fitness value. When looking for a solution one merely needs to look for a maximum or a minimum in the search space. When the search space has a considerable size the search for a solution can be a very complex task due to a limited computational power, so, one needs a way to find where to start looking for a solution in that space. Methods such as genetic algorithms, hill climbing, simulated annealing and tabu search can achieve this. Although they do not necessarily find the best solution, they help find a suitable solution that is close to the best one.

A genetic algorithm starts with a set of solutions, represented by chromosomes, called population. These individuals can reproduce, that is, solutions from one population are used to form a new chromosome through recombination or crossover. This new chromosome, or offspring, can undergo mutation which is the change of its DNA elements. The success that that new organism has in life is called fitness value. Solutions are selected based on their fitness score and the higher the score the higher the chances they reproduce. The process is cycled until a stop criterion is reached.

One way to encoding a chromosome is in a binary string format (this format is the one used on the examples below, Table 4, Table 5 and Table 6) where each bit represents some characteristic of the solution. To give a quick example of a chromosome and a practical application of a GA, let's imagine we want to optimize a trading strategy that uses 2 moving averages (MA) to buy or sell stocks. In this example the chromosome has 3 genes. One for the company selection, one for the first moving

average period and another for the second moving average period, as illustrated in Table 4, Table 5 and Table 6. A long signal is given when MA 1 is greater than MA 2 and a short signal is given when MA 1 is less than MA 2.

Table 4 - Company's gene configuration.

Gene	Company
00	Apple
01	Google
10	Netflix
11	eBay

Table 5 - First moving average gene configuration.

Gene	Period Length
00	5
01	10
10	15
11	20

Table 6 - Second moving average gene configuration.

Gene	Period Length
00	25
01	30
10	35
11	40

A possible chromosome would be 00, 10, 01, which corresponds to the chromosome [Apple Company with a 15 and 30 for MA 1 and MA 2 periods, respectively]. The fitness value can be the profit obtained over a period of time trading the Apple Company following the rule mentioned above with the MA periods of the chromosome. Then, using operations mentioned below the GA will search for an optimum solution.

The **Selection Operator** favors the better individuals giving them a better chance of passing on their genes to the next generation. The better individuals are elected based on their fitness value which is evaluated by the fitness function.

The **Crossover Operator** takes two individuals from the population, using the Selection Operator, and exchanges the values between them. An example of crossover between two particular individuals is shown in Table 7.

Table 7 - Crossover operator example. Offspring recombines information from the two parents.

Parent 1							
0	0	0	0	0	0	0	0
Parent 2							
1	1	1	1	1	1	1	1
Offspring							
0	0	0	0	1	1	1	1

There the first half of parent 1 is combined with the second half of parent 2 to form a new offspring. The idea is to recombine portions of good individuals hoping to create fitter offspring individuals which are then included in the next generation of the population.

The **Mutation Operator** picks, with some probability, a portion of a new individual and flips their values. on Table 8 we show an example of this. There the value of four middle bits are flipped and together with the original other four, two at each end, they form a new individual.

Table 8 - Mutation operator example.

Individual before mutation							
0	1	0	1	0	1	0	1
Individual after mutation							
0	1	1	0	1	0	0	1

The operations of a genetic algorithm are outlined in Figure 2 and described below:

1 – Randomly generate an initial population of n chromosomes (possible solutions for the problem) (t);

2 – Evaluate the fitness value for each of the chromosomes in the population (t);

3 – Repeat: Create a new population using the following operations:

3.1 – **Selection** of parents from population (t);

3.2 – **Crossover** on parents to create population (t+1);

3.3 – **Mutation** of population (t+1);

3.4 – Evaluate fitness of population (t+1);

4 – Stop when a *criterion* is reached.

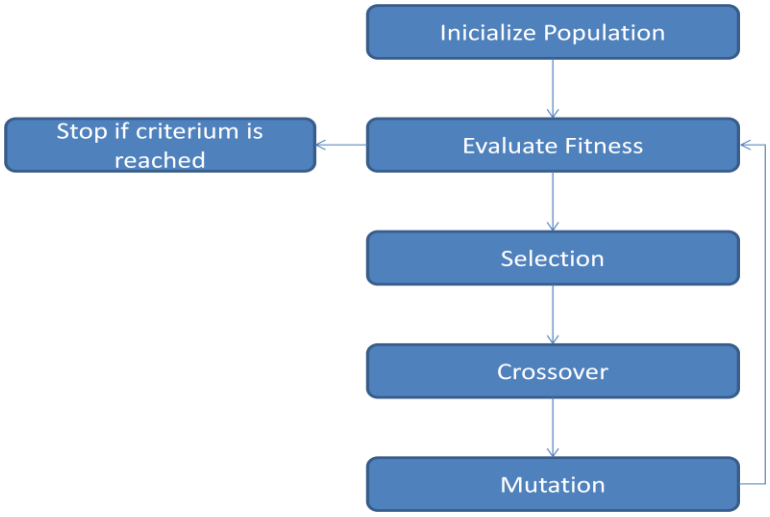

Figure 2 – Genetic algorithm operation.

2.6. Existing Solutions

In this section we present a few examples of existing research on sentiment analysis alone and also, stock market prediction using sentiment analysis. Some interesting research papers on sentiment analysis are mentioned just to give the reader an idea of what has been done or, how useful research on the subject can be, while research papers that present techniques similar to the ones used in this work are more extensively explained.

2.6.1. Sentiment Analysis Practical Applications Examples

Sentiment Analysis for User Reviews

Zhang et al. [30] built a ranking score for products based on their reviews on online stores, such as Amazon. McGlohon et al. [31] attempted to put together a model that measures the true quality of the products or merchants based on the reviews from different sites with different rating scales. Asur and Huberman [32], Joshi et al [33], and Sadikov et al. [34] used tweets to predict box-office revenues for movies. Liu et al. [35] attempted to extract sentiment from blogs and use that information to predict product sales performance. Neethu and Rajasree [36] and Gautam and Yadav [37] used machine learning techniques to performed sentiment analysis on online customer reviews.

Sentiment Analysis for Sports Betting

Hong and Skiena [38] investigated the relationship between the NFL betting line and public sentiment extracted from blogs and microblogs such as Twitter. They demonstrated that a betting strategy using sentiment is profitable, identifying the winner about 60% of the time.

Sentiment Analysis for Political Applications

In another approach, the Twitter was used to investigate political sentiment and deliberation and to test whether political sentiment displayed on the social network is similar to offline political sentiment [39]. Still in a political dimension, they examined the correlation between the analysis results of the 2010 US congressional elections predictions and the corresponding electoral outcomes [40]. Authors argued that predictions about events using social media should not be accepted unless the predictive power of the model is explained. In another scientific paper the election results of the UK 2015 General Election was forecasted [41]. On the 2012 U.S. Presidential Election Cycle a system that gauges real-time sentiment expressed on Twitter towards presidential candidates was proposed [42]. Yano and Smith [43] studied the correlation between the text content of political blog posts and the amount of response comments. Yano et al. [44] tested if campaign contributors affected what politicians say by analyzing tweets posted by members of the US congress. Finally, Vaziripour et al [45] analyzed the political sentiment of tweets in Farsi.

Sentiment Analysis for TV Audiences

Tiara et al. assessed the quality of television programs by analyzing comments posted on twitter using sentiment analysis. Their study, performed using Support Vector Machines together with a lexicon based approach, achieved up to 80% accuracy in predicting if the sentiment towards a television program was positive or negative.

Sentiment Analysis for Social Mechanics

De Choudhury et al. [46] investigated emotional states and identified the different moods of Twitter users, according to their posts. Email texts were also used to produce sentiment classification and to identify and categorize social relations in another study [47]. Cha et al. [48] gathered near 1.7 billion tweets to study what makes a user's influence. Tweets were also used in another study [49] with sentiment analysis in order to build six time series, each representing a specific mood dimension e.g. tension, depression, anger, vigor, fatigue and confusion. They then analyzed these time series during several events such as fluctuations in the stock market and crude oil prices, 2008 US Presidential Election and Thanksgiving Day. The difference between public opinion measured from traditional polls and surveys and public opinion measured from contemporary Twitter messages was also studied [50].

Sentiment Analysis for Public Health

Park et al. [51] examined tweets' text content to detect people suffering from depression. Paul et al.

[52] proposed a method that uses Twitter to track illnesses, measure behavioral risk factors, group illnesses by geographic region and analyze symptoms and medication usage was proposed. Mislove et al. [53] analyzed whether or not Twitter users are a true representative of the overall population along three measures: geography; gender and; race/ethnicity.

2.6.2. Sentiment Analysis Procedures and Techniques

Sentiment Analysis using Lexical Dictionaries

Agarwal et al. [54] studied sentiment analysis on Twitter by building models that classified tweets into positive, negative and neutral sentiment. They experimented three types of models: unigram model, a feature based model and a tree kernel based model. They acquired 11 875 tweets from a real time stream without any location restrictions or language restrictions, instead, in case of a non English tweet, they used Google translate to convert it into English. Every tweet was sorted by a human and it could belong to one of four classes: positive, negative, neutral and junk. Junk class meaning the tweet was not understood by the human thus it was removed from the sample.

They used a 170 emoticons list from Wikipedia and assigned each of them a label. Additionally they compiled an acronym dictionary from the Noslang website[*] and, for example, “lol” was translated into laughing out loud. Next, for the tweet's preprocessing, all emoticons were replaced by their respective label, URLs were replaced by “||U||”, all targets, e.g. “@John”, were replaced by “||T||”, negations, e.g. “not”, “no”, “never”, “n’t” and “cannot”, were replaced by “NOT” and sequences of repeated characters were replaced by the same character repeated three times. e.g. “cooooooooool” becomes “coool”. This allows for the differentiation between regular and emphasized usage of the word.

To score the polarity of the words they used the Dictionary of Affect in Language (DAL) [55] and extended it based in WordNet. That dictionary gave every word a pleasantness score between one and three, which was then normalized dividing by the scale, so that words with a score of less than 0.5, between 0.5 and 0.8 and above 0.8 were assigned negative, neutral and positive polarity, respectively. If a particular word was not in DAL then a synonym was found using WordNet. If the a synonym was in the DAL, its pleasantness score was attributed like if it were the original word, on the other hand, if no synonyms to the original word was found in the DAL, no polarity was assigned to it.

Furthermore, they designed a Partial Tree kernel model that computes the similarity between two trees by comparing all sub-trees. The Partial Tree kernel was proposed by [56] and is based on convolution Kernels [57] to represent tweets in order to combine many categories of features in one succinct representation. They proposed also what they called the 100 Senti-features model. This model consisted of several numerical variables, including number of positive words, number of negative emoticons, number of hashtags and boolean variables such as the presence of exclamation points or capitalized text. 75.39% was the best accuracy they obtained and was achieved using the

[*] www.noslang.com

unigram model plus some features in the Senti-features model.

Barbosa and Feng [58] proposed to explore the way tweets are written and the words that compose them, to detect sentiment. Twitter data was collected using the word “of” as search term. The use of the common stopword “of” was explained by the fact that authors were interested in generic data. That data, representing three weeks time, was obtained from three websites (Twendz, Twitter Sentiment and TweetFeel) that provided sentiment detection engine for tweets. Those tweets, containing the keyword “of” and the corresponding label, given by the website, were stored for future analysis. The websites Twendz and Twitter Sentiment labeled each tweet as positive, neutral or negative while TweetFeel labeled each tweet either positive or negative.

Two sets of features were built: The meta-features and syntax features. Meta features include part-of-speech tags and word's polarity and subjectivity labels. Polarity labels were positive, negative and neutral whereas subjectivity labels were weak and strong. The subjectivity and polarity dictionary used came from a previous published work [59]. The subjectivity lexicon is available in the web[*]. Additionally they added some extra popular words found on online discussions.

Syntax features were the occurrences, in a tweet, of retweet, hashtags, reply, links, exclamation marks, question marks, emoticons and upper case words.

Different classifiers from the Weka framework [60] were tried. The best results were achieved using Support Vector Machines with the features sets described above (error rate of approximately 20% either for subjectivity or polarity detection). Interestingly, they noted that training sample size has a heavy influence on the classifier's error rate respecting unigram features.

Sentiment Analysis using Statistical Methods

In EMOTEX [61] it is attempted to classify text messages of individuals to ascertain their emotional states. To model sentiment they used the well-known Circumplex Model of Affect [62] that maps emotions in a two-dimensional circular space defined by arousal, as the vertical axis, and valence, as the horizontal axis. This Circumplex model suggests that all emotional states can be represented by a linear combination of arousal and valence. The 28 words from the Circumplex Model of Affect were used and the list extended with synonyms from WordNet. Those keywords were used as hashtags for a search parameter to collect data from Twitter. Then a number of preprocessing steps were taken. First, usernames and URLs were replaced by “USERID” and “URL” respectively. Next, letters occurring more than two times consecutively were replaced by a single occurrence. Then, tweets containing hashtags from different classes or emoticons from different classes, for example “(:” and “):”, or hashtags and emoticons from different classes were removed from the dataset. Finally, hashtags that were not part of the sentence (usually hashtags that are at the end of a tweet) were also removed. The sets of features explored were unigram features, emoticon features, punctuation features and negation features.

[*] http://mpqa.cs.pitt.edu/

Best achieved precision values were 90.3% and 90.4% using Support Vector Machines with unigram features and unigram plus punctuation features respectively. Best achieved recall values were 90.1%, 89.9% and 89.7% using K-nearest neighbors, Decision Trees and Support Vector Machines with any set of features, all sets of features and just unigram features respectively.

Davidov et al. [63] proposed a supervised sentiment classification framework for Twitter using tags and smileys. As classification features, they used single words features, n-grams features, pattern features and punctuation features which were then combined into a single feature vector.

Single word and n-gram-based features had a binary value with weight equal to the inverted count of the word in the corpus. N-grams contained from 2 to 5 words and features that appeared in less than 0.5% on the training data were removed. Smileys and sequences of two or more punctuation characters were used as single word features. Additionally URLs, references and hashtags were replaced by “URL”, “REF” and “HASHTAG” respectively.

They then differentiated high-frequency words from content words, according to the frequency they appear in the corpus and coded them high-frequency words (HFW) or content words (CW), which was an approach proposed before [64]. Then a valid pattern would take the form [HFW] [CW slot] [HFW], for example. Finally, each pattern was scored and then it could be included in the feature vector.

Additionally, tweet length and the number of exclamation points, question marks, quotes, capitalized words in a tweet were also used as features. Their value was normalized by dividing the maximum observed value times the average of the maximum values of the other features.

From a Twitter dataset of more than 2.5 million different hashtags they selected 3 852 most frequently used. These hashtags were manually separated into the following five categories: 1 - strong sentiment; 2 - most likely sentiment; 3 - context- dependent sentiment; 4 - focused sentiment; and 5 - no sentiment.

The classification algorithm they used was k-nearest neighbors (kNN). Moreover they test with multi-class classification, binary classification, evaluation with human judges, exploration of feature dependencies, tag co-occurrence and feature overlap. To provide an example, with multi-class classification, an accuracy level of 64% and 31% was achieved using smileys and hashtags features respectively.

An automatic Twitter message classification framework was proposed Go et al. [65]. Here it was assayed three different machine learning classifiers: Naive Bayes, maximum entropy and support vector machines. Features sets were unigrams, bigrams, unigrams and bigrams and unigrams with part of speech tags. Their framework allowed authors to effortlessly combine a different classifier with a different set of features. Only tweets in English were collected using two query terms,“:)” and “:(“, in the Twitter API from April 6, 2009 to June 25, 2009. In their Twitter data, the search term in each tweet was replaced by “QUERRY_TERM”, so that the search term by itself wouldn't bias the results.

Retweets and tweets with both negative and positive emoticons were removed from the dataset. The emoticons ":)", ":-)", ":)", ":D", "=)", ":(", ":-(" and ": (" were removed from the training data. They found that emoticons hurt accuracy, so they consider them as noisy labels as they were faulty at defining the actual sentiment of a tweet.

To further reduce the feature space, all words that start with the symbol "@" (or usernames on Twitter) were replaced by the token "USERNAME". All links were replaced by "URL" and, finally, every sequence of three repeated letters or more was replaced with only two occurrences. Altogether the previous steps reduced the number of features to 45.85% of the original size. The total number of tweets used for training was 1,6 million, halve containing positive emoticons, and the other halve containing negative emoticons.

To collect test data, the search queries used were different consumer goods, companies or persons, such as "kindle2", "mcdonalds" and "warren buffet". The best achieved accuracies, at predicting positive or negative feeling in a tweet, were 82.7% and 83.0% with Naive Bayes and Maximum Entropy, respectively, using unigram plus bigram features and 82.2% with support vector machines using only unigram features. Bigram features alone and unigram plus part of speech features gave the poorest results.

The accuracy of two machine learning algorithms, logistic regression and neural networks, on classifying sentiment on stock-related tweets as positive, negative or neutral was studied in another investigation [66]. Classifier's features included bigram term frequency and unigram term frequency - inverse document frequency. Computational tasks were performed using Microsoft Azure ML.

Every tweet was represented as a vector of weights, each weight is the term frequency (number of times a term appears in the tweet) and term frequency - inverse document frequency. To normalize these values, since every tweet may have a different length, the term frequency was divided by the total number of terms in the tweet. Inverse document frequency measures how many tweets, in the whole dataset, contain a term. It is calculated as the logarithm of the total number of tweets divided by the number of tweets containing the term. The term frequency – inverse document frequency is the multiplication between the term frequency and the inverse document frequency.

Authors built a training dataset of 42 000 tweets, using the Twitter API, which were automatically labeled according to the following rules: if a tweet contained either positive or negative emoticons, it is labeled accordingly; if both positive and negative emoticons were present, the tweet is removed and; if no emoticons or keywords (*e.g.* "happy", "sad", "good", "rise" and "down") that display polarity were present, the tweet was labeled as neutral.

Text preprocessing includes replacement of special characters and duplicate letters, filtering stop words and word stemming. For dimensionality reduction, the number of features was limited to 5 000. This selection was made using Chi-squared values. 70% of the dataset was reserved for training and 30% for testing.

Investigators concluded that unigram features with term frequency – inverse document frequency weighting outperforms, with an overall accuracy of 58%, bigram with term frequency weighting. However their model showed a high conflict between classes positive and neutral, meaning, a high portion of positive tweets were identified by their classifier as neutral and vice-versa.

Kanakaraj and Guddeti [67] introduced a way to improve sentiment classification using a Natural Language approach. The idea was to include semantic and context based variables in the feature vectors.

The proposed system performs the following tasks: Collect data using Twitter's API using a specific word as search term. After that, a few preprocessing techniques were performed. First, stop words were removed. Secondly, letters repeated more than two times consecutively were replaced by only two occurrences. Thirdly, URLs and hash tags symbols were removed, so, "#football" becomes "football". Also, usernames were replaced by a placeholder and words with both alpha and numeric characters were removed. Moreover, part of speech tagging was performed. This task was designed to identify the role of a word in a sentence, i.e. noun, verb, adverb. Afterwards, word stemming was performed and, for every word in a tweet, its synset was looked up in WordNet and it was added to the tweet. A word's synset is a set of synonyms of that word. The following step involved the use of *WordNet Sence Relate* to discriminate the meaning of words that have multiple definitions in the dictionary. Finally, a feature vector was built. It consisted of key terms, the key terms sense value and the synsets.

The classification algorithms used included the Naive Bayes, Support Vector Machines, Maximum Entropy and several ensemble methods such as Random Forests, Decision Trees with and without Ada Boosting and Extremely Randomized Trees. Best results were achieved using ensemble methods, with an F-score of approximately 88%. Support Vector machines achieved an accuracy of approximately 86%.

2.6.3. Sentiment Analysis for Stock Market Prediction

In this section we present applications of sentiment analysis for stock market prediction. The section is divided in three parts, **A**, **B** and **C**.

Part **A** cites research on sentiment analysis for individual stocks price prediction. For example, Vu et al. [68] captured public mood from Twitter to forecast the up and down stock price movement of Apple, Google, Microsoft and Amazon. Chung and Liu [72] delved into stock market prediction by studying the sentiment score of tweets of the top five gainers and the top five losers of companies in the technology sector. Feldman et al. [77] created Stock Sonar, which is an application designed to collect articles (related to the stock market) from thousands of sources, analyze them and help investors make better trading decisions. Xu and Keelj [3] tested whether StockTwits is useful in a stock market forecasting and, finally, Bouktif and Awad [79] proposed an ant colony based approach model for

stock market movement prediction using public mood gathered from Twitter.

Part **B** presents research on sentiment analysis, but this time, for composites price prediction. For instance, Bollen et al. [1] and Mittal and Goel [2] attempted to predict the price movement of the DJIA using Twitter. Si et al. [4] proposed a new technique that uses a continuous Dirichlet Process Mixture model to estimate the number of topics on Twitter streaming messages. Porshnev et al. [5] proposed to identify eight basic emotional states of Twitter users and improve accuracy of stock market predictions of the DJIA and S&P500 composites.

Part **C** cites research on sentiment analysis used to understand investor sentiment. For example, Chua et al. [84] designed a new sentiment detection engine for a very popular Australian investment forum, HotCopper.

A - Sentiment Analysis for Individual Stock Price Prediction

Vu et al. [68] captured public mood from Twitter to forecast the up and down stock price movement of four tech companies. They achieved an accuracy of 82.93%, 80.49%, 75.61%, and 75.00% for Apple, Google, Microsoft and Amazon, respectively, for a period of 41 days. In another study [69] it was tried through text analysis to better identify which news, such as those published in the Wall Street Journal and Dow Jones News Service, had significant effect on the stock's price and which had little or no effect.

Zhang and Skiena [70] performed text analysis on blogs and news and related the results with a company's stock trading volumes and financial returns while Baik et al. [71] examined the influence of Twitter on the stock market. Furthermore, they studied the influence of the tweet depending on the geographic location of the tweet's poster.

Chung and Liu [72] delved into stock market prediction by studying the sentiment score of tweets of the top five gainers and the top five losers of companies in the technology sector. Another case study intended to investigate if there is a type of company whose stock price is more easily predictable analyzing public sentiment using Twitter messages [73]. Concretely, from a set of 30 stocks listed in the NASDAQ and the NYSE, they tried to find which were more suitable for price prediction, using Twitter sentiment analysis. The proposed algorithm achieved 76.12% accuracy in the price movement prediction.

Zhang [74] studied the correlation between public sentiment on Twitter and the stock price fluctuations. The author looked for specific words that were more commonly present in tweets of the best, or the worst, performers on the stock market. Another example [75] targeted the performance of IPOs (Initial Public Offering is the first time a company is sold publicly, on the stock market). This study investigates the relation between the first day returns of 325 IPO's and sentiment score collected from Twitter.

Hu et al. [76] proposed to study the correlation between one important user's tweets, the CEO of Tesla

– a famous automobile company – Elon Musk and the price behavior of that company's stock using a Support Vector Machine algorithm. They collected 816 tweets of that user from June 4, 2010 to November 27, 2013 and Yahoo! Finance data, which consisted of daily open and close prices of the stock. The tweets posted in the same day were grouped together and the average number of favorites and retweets were computed. Tweets posted when the market was closed were discarded. With this data three different sets of features were tried in a Support Vector Machine model.

The first set includes a label vector, number of favorites, number of retweets and a vector of the "Indication Factor". "Indication Factor" is the number of times a word appears when price changes more than 3% divided by the number of times a word appears when price changes less than 3%. Values of medium frequency words (very frequent and very sparse words were removed from the vector). The label vector could have two values. Price changes of more than 3% were labeled 1, otherwise label was -1.

The second set included the label vector from the first set plus the sentiment scores from the tweets. To get the sentiment score of a tweet, first, all links, URLs and non-alpha characters were removed from it and all letters were converted to lowercase, second, they used the Natural Language Toolkit to label every tweet as positive, negative or neutral.

The third and last set was the second set plus the number of favorites and the number of retweets considering the time series effect, that is, the features for this set were tweet's sentiment score, number of retweets and number of favorites of the past N days.

Best results were achieved using the third set of features with an accuracy of 60% considering N equal to three.

Feldman et al. [77] created Stock Sonar, which is an application designed to collect articles (related to the stock market) from thousands of sources, analyze them and help investors make better trading decisions. The main textual content from the HTML pages was extracted using a supervised learning machine technique and a visual training module described in [78]. This module removes non-essential content from the articles, such as ads, links and other stories, and keeps only plain text and its associated stock. Based on this, a sentiment score is calculated for each article and each stock ticker get its score combining the scores from every article.

Stock Sonar is a new hybrid approach to sentiment analysis, that is, a combination of three techniques: dictionary-based sentiment, compositional phrase-level patterns and semantic events. The dictionary-based sentiment consists of an expanded combination of McDonalds's Lexicons[*] and the Harvard Inquirer[†]. It contains around 2000 words or expressions each of them can portrait either a positive or negative sentiment. The phrase-level pattern evaluation is capable of deciphering multiple patterns in the same sentence, each one with its own sentiment. For example, when the sentence

[*] http://www3.nd.edu/~mcdonald/Word_Lists.html
[†] http://www.wjh.harvard.edu/~inquirer/

“Sales rose 10% although share price is at record low and expenses were worse than expected.” is fed into their engine, it gets separated into three parts: “Sales rose 10%”, “share price is at record low” and “expenses were worse than expected”, then, the sentiment is obtained combining the resulting analysis from each of those three pieces. Finally, articles on business events are also monitored. These include the announcements of mergers and acquisitions, partnerships, new deals or products, stock price changes amongst others.

All this measures are used to compute a company's score, regarding sentiment, every day. Sentiment scores and the most positive and negative articles can be accessed at The Stock Sonar website. Users can also build their own portfolio and monitor it. Also, they have an API, so that algorithmic trading companies can incorporate sentiment scores into their trading signals.

Xu and Keelj [3] tested whether social media sentiment is useful in a stock market forecasting. Authors proposed a two-stage method for sentiment analysis and stock market price movement prediction. Firstly they collected tweets from 16 frequently discussed stocks on StockTwits from March 13, 2012 to May 25, 2012. Secondly, tweets published on weekends and public holidays and duplicated ones are removed. Text was converted to lowercase and the “@” sign plus username, the $TICKER for each company, and links were replaced by “atreplace”, “stocksignreplace” and “linkreplace” respectively.

After these, preprocessing steps were executed. 2 380 tweets were hand labeled, according to a specific set of guidelines, as negative, positive or neutral. Of those, 2 000 tweets were used for training and 380 for testing. On a first stage a machine learning classifier would discriminate neutral from polarized tweets, and in the second stage, the polarized tweets would be separated into positive or negative. An overall accuracy of 71.84% and 74.03% was achieved on the test data, on the neutral vs. polarized detection and positive vs. negative detection, respectively, using support vector machines.

The Pearson product-moment correlation coefficient measures the linear correlation between two variables. It assumes values from -1 to +1. If the correlation value is -1 the two variables are totally negatively correlated, if the value is +1 the two variables are totally positively correlated and if the value is 0 there is no correlation. This coefficient was used to measure the dependence between the volume of tweets and the stock's trading volume. Investigators observed that user activity on StockTwits, between 4:00pm and 9:30am, is positively correlated with traded volume on the next trading day. Furthermore, a Granger Causality test was used to verify if there is a predictive ability between sentiment collected from tweets and the stock's price and they concluded that, indeed, collective sentiment, considering tweets posted in the same time frame as above, has predictive power over the next day's price on 9 of the 16 stocks they studied. They were able to predict up and down stock price movements with an accuracy of 58.9%.

Bouktif and Awad [79] proposed an ant colony based approach model for stock market movement prediction using public mood gathered from Twitter. This model uses an ensemble technique, which is,

in essence, a combination of predicting models, and it is intended to be more interpretable than existing ones. Exactly, this model is a combination of Bayesian classifiers optimized using an Ant Colony based approach.

The authors argued that although some machine learning algorithms have acceptable accuracies on this domain – predicting stock market movement using Twitter mood – they provide very little interpretability. A machine learning model interpretability or explicability is the ability it has to explain the predictions. Models that provide very little interpretability are called back box machine learning algorithms. Support Vector Machines, Random Forests and Neural Networks are examples of such models. Although they can be very accurate, they do not provide a causality relationship between prediction and input features. On the other hand Bayesian classifiers and fuzzy rules are examples of interpretable techniques because one can see the impact of an input feature in the final prediction. Moreover, although common ensemble methods, such as Bagging and Boosting, can achieve better accuracies, they can even deteriorate the initial model explicability.

To achieve a better interpretability, a set of Bayesian classifiers was built and data mentioning a particular company was collected from Twitter. In total, 18 mood sentiment attributes (such as depressed) were monitored. Each classifier represented a mood state attribute and the optimization of the Bayesian classifiers and the final prediction was made using an ant colony based approach [79]. Best achieved precision and recall values, on predicting next day's price movement (up or down), were 62.65% and 70.32%, respectively.

B - Sentiment Analysis for Composites Price Prediction

Bollen et al. [1] attempted to predict the price movement of the DJIA using daily Twitter feeds. More than 9 million tweets were collected from February 28th to December 19th, 2008. Punctuation and stop-word were removed from the tweets and only those which contain expressions such as “i feel”, “i am feeling”, “i'm feeling”, “i dont feel”, “I'm”, “Im”, “I am”, and “makes me” were kept. In order to control the amount of spam present in the dataset, tweets containing a link or URL were excluded.

The text content from Twitter feeds with OpinionFinder[*] and Google-Profile of Mood States (GPOMS) was analyzed. The OpinionFinder is a tool that measures the polarity (negative/positive) of mood on text documents while GPOMS measures mood on text documents in 6 dimensions (calm, alert, sure, vital, kind, and happy). Together, these two tools, output 7 public mood time series, one from OpinionFinder and the other 6 from GPOMS. Subsequently a Granger causality test was used to verify if public mood, according to OpinionFinder and GPOMS, was predictive of the DJIA closing price. As the Granger causality test focuses on linear relationships, they employ a Self-Organizing Fuzzy Neural Network (SOFNN) to further explore the non-linear relationship between public mood and stock prices.

Best DJIA daily prediction accuracies were achieved using DJIA close prices of the three previous days together with the Calm mood dimension, for the same period, as inputs for the SOFNN. They

[*] http://mpqa.cs.pitt.edu/opinionfinder/

claimed an 87.6% accuracy predicting the closing price movement direction, although their test period starts at December 1, 2008 and ends on December 19, 2008.

Another example evaluates the correlation between Twitter posts and stock market composites such as the Dow Jones, NASDAQ and S&P 500 [80]. In this work it was observed that the percentage of tweets with emotional content was negatively correlated with the three stock market indicators mentioned above, but positively correlated with the VIX (Chicago Board Options Exchange Volatility Index).

Using the words “Hope”, “Happy”, “Fear”, “Worry”, “Nervous”, “Anxious”, “Upset”, “Positive” and “Negative” as search terms they downloaded the tweets from March 30, 2009 to September 7, 2009 averaging 29758 tweets per day. They found that people use more emotional words, regardless of their positive or negative context, in times of economic uncertainty, thus suggesting a negative correlation with Dow Jones NASDAQ and S&P 500. Additionally, and using the word “worry” as an example, they took the number of “worry” tweets in a day and divided by the total number of tweets collected in that same day. The same procedure was repeated for the following days and a time series was built. Then, the correlation coefficient between that time series and the price movement on the next day was calculated. They found that the use of the words “hope”, “fear”, “worry”, “anxious” and “negative” was significantly correlated with the price movements of the stock indexes.

Applying the same technique as the one above, a time series was built for the number of retweets of the nine emotion words. Only the number of retweets of the word “fear” was found to be significantly correlated with the price movements of the stock indexes. Finally, they concluded that a much higher correlation coefficient was obtained if the word utilization time series was built using the previous three days instead of just the single previous day.

Mittal and Goel [2] tried to find a correlation between public sentiment and market sentiment particularly they use Twitter to predict the DJIA price movements. They collected more than 476 million tweets from June to December 2009. A sentiment analysis algorithm was then used to classify the text into 4 different mood classes: calm, happy, alert and kind.

They built their own sentiment dictionary using the notorious Profile of Mood States (POMS) questionnaire[*]. POMS is a psychometric questionnaire used to evaluate a person’s current mood. The questionnaire is composed of 65 feelings and the participants have to answer how much of each feeling, on a level from 1 to 5, is being felt at the moment. Example questions are “how sad”, “how relaxed”, or “how bitter do you feel?”. These 65 feelings were then mapped on to 6 POMS moods, namely, anger, confusion, depression, fatigue, tension and vigor. That 65 word list was extended adding commonly used synonyms of those 65 words found using SentiWordNet[†] and Thesaurus[‡]. To reduce the amount of data to be processed, only used tweets with the expressions “feel”, “makes me”,

[*] https://www.brianmac.co.uk/poms.htm
[†] http://sentiwordnet.isti.cnr.it/
[‡] http://www.thesaurus.com/

“I'm” and “I am” were used.

The normalized daily score of every word in the list was calculated and mapped to one of the 6 POMS moods states. Then, the 6 POMS states were mapped to their four mood states (calm, happy, alert and kind). As an example, happy was considered the subtraction of depression from vigor. The use of Stanford core Natural Language Processing software for word tagging and to find a word’s importance based on its position on a sentence was also attempted but authors concluded that the process was too slow and of little help.

With these 4 mood time series and the DJIA values of the previous three days as inputs to a SOFNN they achieved 75.56% accuracy in predicting the direction of price movement.

Si et al. [4] proposed a new technique that uses a continuous Dirichlet Process Mixture (DPM) model to estimate the number of topics on Twitter streaming messages. The DPM Topic model is a statistical method for discovering topics from a large collection of text documents. One of such models is the Latent Dirichlet Allocation (LDA) [81]. The DPM is an unsupervised learning technique produced by the extension of the LDA [82]. Essentially, a topic model, such as the DPM, tries to separate tweets into topics by building clusters of the words contained in them. So each cluster represents a topic and a topic can be represented as a probability distribution of words.

With the application of this model, a sentiment score was calculated for each topic and a sentiment time series was built. The sentiment time series together with the stock market price series S&P100 (top 100 constituents of the Standard & Poor's index) were regressed to predict future price movements.

With the aid of an opinion lexicon every word in a tweet was categorized as an opinion word or a non-opinion word, then each opinion word was given a score: “+1” for positive words and “-1” for negative words. The non-opinion part of the tweet was used, with the DPM, to separate the daily Twitter data into topics and that topic became the tweet's class. Finally, a sentiment score was calculated for each topic using the polarity scores of the words from the opinion part of the tweets from that topic, so, the daily scores for each topic became points in a time series. Along with the sentiment based time series, they built an autoregressive model of the price time series. They used that model in a sliding window manner, i.e. train with the data in [t, t+n] and predict tomorrow's price [t+n+1].

Several window sizes and lag settings were tried and best average achieved accuracies (over all window sizes), using the sentiment based DPM time series together with the autoregressive model, were 60% for lags 1 and 2, and 61% for lag 3, i.e. the previous three days’ sentiment correctly predicts price movement on the next day, 61% of the time.

Ranco et al. [83] investigated the relation between the volume and sentiment of tweets and the stock market performance of the 30 companies on the Dow Jones Industrial Average over a period of 15 months. During this time, the correlation and Granger causality between sentiment and stock price

was found to be very low, except during peaks of Twitter activity. In fact, when Twitter activity surged a significant dependence between sentiment and stock returns was found.

Porshnev et al. [5] proposed a model that used a lexicon-based approach to identify eight basic emotional states of Twitter users and improve accuracy of stock market predictions using Support Vector Machines and Neural Network algorithms on DJIA and S&P500 composites.

Three features sets were built. The first was the Basic set which was "the characteristics of the stock market in previous days". The second set was called Basic&WHF and it was the first set plus the normalized number of tweets, per day, containing the words "Worry", "Hope" and "Fear" and, lastly, the third was the Basic&8EMO set which was a combination between the first Basic set and the normalized number of daily tweets with the words "happy", "loving", "calm", "energetic", "fearful", "angry", "tired" and "sad".

Using Twitter API, 755 000 101 tweets were collected from February 13, 2013 to September 29, 2013. The corresponding financial data, related to the DJIA and the S&P500 was downloaded from Yahoo Finance.

Best achieved accuracies were 64.10% and 61.84% using the Basic&8EMO and Basic&WHF feature sets, respectively, with Support Vector Machines for the DJIA and 62.03% and 60.53% using the Basic and Basic&WHF feature sets, respectively, with Support Vector Machines for the S&P500.

C - Sentiment Analysis to understand Investor Sentiment

Chua et al. [84] designed a new sentiment detection engine for HotCopper[*]. HotCopper is a very popular Australian investment forum that allows users to post stock related messages. The new engine was designed to classify investor sentiment using posts on this forum. Every post has a sentiment label, which can be "Buy", "Sell", or "Hold". In the analysis they used data posted on HotCopper for the first six months of 2004. All the collected posts were related to the stock on the ASX (Australian Securities Exchange). They obtained 8 307 labeled posts, averaging 28 words per post, related to 469 stocks. Every message belongs to a thread and every message within a thread is related to the same stock.

A set of preprocessing steps were performed. Stop words taken from the Natural Language Toolkit (NLTK) stop word list and non-informative tokens, such as "haaaaa" and "oohhh", were removed. Then a word stemming algorithm was run on the data applying spell correction with the PyEnchant[†] package, which is a spell checking package for the programming language Python.

As a single thread could have multiple posts and, therefore, multiple opinions (which could be distinct) investigators came up with a volatility measure that assigned a value to the level of disagreement

[*] http://hotcopper.com.au/
[†] http://pythonhosted.org/pyenchant/

between posts within a thread. This means that threads with posts labeled “buy and sell” were given a higher volatility figure than threads with posts labeled “buy and hold”. Threads with high volatility were then removed from the training data.

The classification algorithm used was the Naive Bayes that incorporated the term frequency – inverse document frequency or, instead, Boolean values representing term presence rather than term count. When, instead of the term frequency count, we have just a boolean value that represents the presence or absence of the term, it is said we have a Bernoulli model of Naive Bayes.

To account for the unbalanced data and avoid a skewed data bias a Complement Naive Bayes (CNB) was implement [85]. All features were then ranked by frequency and an optimal set of features was selected. This group was further reduced using the InfoGaing algorithm [86]. Additional features incorporated were sums of negative and positive bigrams or trigrams including financial terms, such as EPS, dividends and profit, and stock price alerts, which are a comparison between the current stock price and its past price measured in standard deviation.

The best achieved accuracies were 78.72% and 78.45%, using the CNB classifier with InfoGain and Bernoulli NB classifier with InfoGain respectively, in predicting the sentiment on texts for a particular stock.

2.7. Chapter conclusions

In this chapter we started by briefly explaining what are the existing market analysis techniques and how sentiment analysis fits in this group. Next, we talk about what is Twitter, when it was created, who uses it, how popular it is, how it works and what are the key terms and functionalities. Additionally, we mentioned the different API services Twitter provides and how much data a developer can access according to the API used. We also talked about the power and influence Twitter messages can have on the stock market. A few examples of that influence were provided with the corresponding price movement amplitude. Next we explained what sentiment analysis is and what the possible applications are. Then, some essential key terms were disclosed. These terms are essential to understand the procedures that come next on the literature review. On the literature review or related work, we presented many examples of research of practical applications for sentiment analysis. Methods, applications and results of most relevant related works are summarized in Table 9 and Table 10.

By the different applications that are proposed in the literature, one can see that sentiment analysis is here to stay and that we can gain a deeper understanding of what is going on around us and companies, organizations or governments can make better decisions by analyzing public mood on social networks.

Table 9 - Achieved accuracies in similar works.

Reference	Method	Period	Application	Results
[1]	Use Twitter mood to predict the stock market	February 28, 2008 to December 19, 2008	Prediction of up/down movement of the DJIA index	87.6%
[80]	Measure investor sentiment by tracking key words on Twitter	March 30, 2009 to Sept 7, 2009	Measure correlation coefficient between key words usage and Dow Jones, S&P 500, and NASDAQ indexes	-0.728 with NASDAQ
[2]	Use sentiment analysis on Twitter data and machine learning to correlate public sentiment and market sentiment	June 2009 to December 2009	Prediction of up/down movement of the DJIA index	75.5%
[76]	Use tweets of Elon Musk, CEO of Tesla, to predict the impact on Tesla stock price	June 4, 2010 to November 27, 2013	Prediction of up/down price movement of the Tesla stock	52.6%
[68]	Use Twitter messages to capture public mood	April 1, 2011 to May 31, 2011	Prediction of the daily up and down price movements of Apple, Google, Microsoft and Amazon stocks	82.93%,80.49%, 75.61% and 75.00% for Apple, Google, Microsoft and Amazon

Table 10 - Achieved accuracies in similar works (continuation).

[5]	Use Twitter sentiment analysis to predict the stock market	February 13, 2013 to September 29, 2013	Prediction of price movement direction on DJIA and S&P500	64.1% on DJIA
[3]	Measure sentiment on microblogs	March 13, 2012 to May 31, 2012	Prediction of the up and down price movement of stocks	58.9%
[4]	Topic based Twitter sentiment	November. 2, 2012 to February 7, 2013	Prediction of up/down movement of the S&P 100 index	61%
[73]	Measure public sentiment analysis in Twitter data to predict stock price movements	October 2011 to March 2012	Prediction of the price movements of 30 companies listed in NASDAQ and the New York Stock Exchange	76.12%

Chapter 3. Proposed Architecture

3.1. System's Overview

An overview of the idea behind the proposed system is described in the following paragraphs. More detailed explanations are reserved for the next sections. There, whenever we mention processing time or how long a task takes to complete, that duration was measured on a machine with a 2 core 2.5 GHz processor with 4Gb of RAM memory.

The main idea was to use Twitter to predict future price movements of companies quoted on the stock market. More concretely, we intended to build a system that is able to automatically interpret the emotional content expressed on tweets in order to deduce what would be the future price movements of individual stocks on the financial market. A practical example of the proposed solution is stated below.

We started by searching and downloading tweets with emotion words. Here, emotion words and what we use as search terms on Twitter are for example the hashtags "#happy" and "#sad" and, for explanation purposes, we will stick with these two for now. Resuming, we search and download tweets that contain the term "#happy" or "#sad". These tweets have nothing to do with the stock market or, if they do, it is purely coincidental. In fact, the contexts in which the term "#happy" appears can be very different from one tweet to another. The same applies to the term "#sad". Examples of tweets downloaded using the term "#happy" are "#Happy #Friday! We hope you all have a beautiful day and a lovely weekend! #DavidJones #SayCheese" and "#popcorn makes me happy lol #movietime #happy". Examples of tweets downloaded using the term "#sad" are "I love being #sad" and "December 2nd and it won't stop raining... #NoSnow #sad #december #foggy".

Once we have these tweets and their corresponding labels we can build a sentiment model, using a classifier that performs statistical analysis. Here the labels could be simply happy or sad, but, in our work, we grouped different emotion words into the same class.

The objective is to differentiate the characteristics of sad tweets from the characteristics of happy tweets, or, in other words, what words are present in the happy tweets that are not present in the sad tweets and *vice versa*. After this, we take tweets mentioning a company, e.g. Google, and use the built model to tell us if the Google related tweets are more similar to the happy tweets class or the sad tweets class. We than do the same for a group of companies and assess their performance on the stock market in order to understand if the companies whose tweets were classified as happy perform better or worse than the companies whose tweets were classified as sad.

3.2. Architecture

In order to perform the different tasks required we created modules to handle each step. A diagram of the proposed solution and the inputs and outputs of each module are outlined in Figure 3.

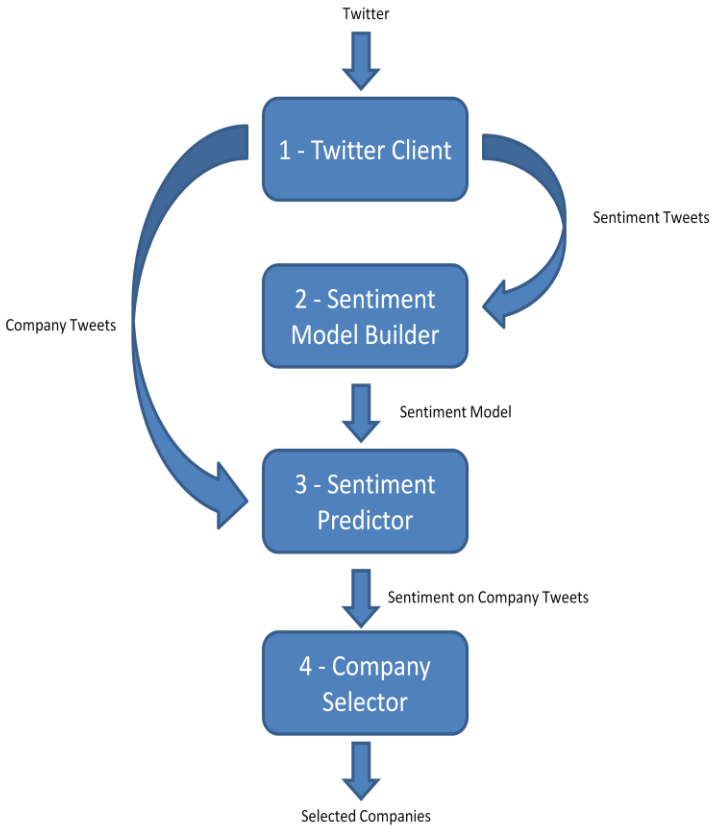

Figure 3 – Proposed system architecture

The first module downloads tweets, using the Twitter API, and stores them on files. Basically, there were two types of tweets that were downloaded and stored. These types can be differentiated according to the type of search term used to query the tweet which was either a sentiment word or a publicly traded company's ticker.

The second module reads collected tweets using the sentiment words from the corresponding files, preprocesses them and learns a sentiment model from them. The learned model can be saved and does not need to be created every time we want to gauge the sentiment on a company's related tweets. Assuming that we have a big enough number of tweets and that words used on tweets of each emotion do not change over time, we can use a sentiment model today that was created a year ago. Simplifying, if we assume that the words used on "happy" tweets and words used on "sad" tweets do not change much over time, there is no reason to build a new sentiment model each time we want to make a stock price prediction, in as much as, creating a sentiment model, is a very r*esource intensive task*. As it is, the sentiment model is stored in a file and it can be loaded very quickly when it is needed.

The third module reads company related tweets from the corresponding files, preprocesses them and uses the model, built in the second module, to measure the sentiment in them. The companies we chose to follow on Twitter are the ones that belong to the S&P 500, NASDAQ 100 and Dow Jones 30.

The fourth and final module takes the sentiment from the tweets of every company in the list and, using custom made trading rule together with a genetic algorithm, outputs which are the best to buy.

3.2.1. 1st Module: Query and Store Tweets

This module is required for downloading tweets using the Twitter Search API. The goal here was to download tweets with a single specific emotion so that the classifier could correlate words with emotion. For this we made use of the Twitter hashtags. Hashtags are words preceded by the character "#". They are very common in Twitter messages. A study in 2011 concludes that among 0.6 million tweets, 14.6% of them had at least one hashtag [87]. Hashtags are a useful feature for sentiment and emotion classification because they can help associate messages and emotions as just the mere presence of a hashtag in a message classifies it emotionally. For example, the tweet "Thankful for unexpected day off #happy" ends with the hashtag #happy that perfectly describes how the user is feeling.

i) Circumplex Model of Affect

In order to map a broad range of the human sentiment we picked 80 emotion words that were used in hashtag form as search terms on Twitter. Essentially, they have the same purpose as the terms "#happy" and "#sad" referred before on the System's Overview section. These words were chosen based both on the Circumplex Model of Affect [62], Figure 4, and on EMOTEX [61].

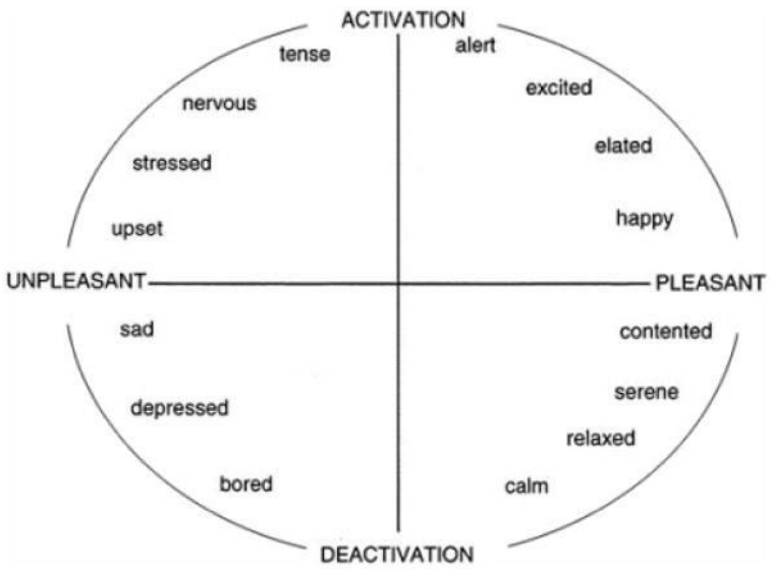

Figure 4 - Circumplex Model of Affect.

The Circumplex Model of Affect, Figure 4 [88], (Used with copyright clearance) is an integrative approach to affective neuroscience, cognitive development, and psychopathology. This model proposes that all affective states can be expressed as a combination of two fundamental

neurophysiological systems, pleasure and activity or arousal, so one can map all emotions in a two-dimensional plot where each human emotion is a linear combination of different degrees of both dimensions.

Each of the 80 chosen words belongs to one of four classes: Activity+ / Pleasure+; Activity- / Pleasure+; Activity+ / Pleasure- and; Activity- / Pleasure- (Table 11). In essence, each of these four classes represents a quadrant as shown in the Figure 4. Emotions located very close to the quadrants boundaries were ignored so that the four classes are as distinct from each other as possible. Identical procedures were proposed in [61].

Table 11 - Identification of hashtags used to search Twitter in this work.

Class	Hashtags
1 - Activity+ / Pleasure+	#elated, #overjoyed, #enjoy, #excited, #proud, #joyful, #feelhappy, #sohappy, #veryhappy, #happy, #superhappy, #happytweet, #feelblessed, #blessed, #amazing, #wonderful, #excellent, #delighted, #enthusiastic
2 - Activity- / Pleasure+	#calm, #calming, #peaceful, #quiet, #silent, #serene, #convinced, #consent, #contented, #contentment, #satisfied, #relax, #relaxed, #relaxing, #sleepy, #sleepyhead, #asleep, #resting, #restful, #placid
3 - Activity+ / Pleasure-	#nervous, #anxious, #tension, #afraid, #fearful, #angry, #annoyed, #annoying, #stress, #distressed, #distress, #stressful, #stressed, #worried, #tense, #bothered, #disturbed, #irritated, #mad, #furious
4 - Activity- / Pleasure-	#sad, #ifeelsad, #feelsad, #sosad, #verysad, #sorrow, #disappointed, #supersad, #miserable, #hopeless, #depress, #depressed, #depression, #fatigued, #gloomy, #nothappy, #unhappy, #suicidal, #downhearted, #hapless, #dispirited

Every emotion hashtag has its own file and every tweet collected using that emotion was appended at the end of the file so that each line corresponds to one tweet. The file was stored in a CSV (Comma-Separated Values) format. All files were stored in the same directory and their names take the form #emotion.csv, so, for example, all tweets collected using the hashtag "#happy" were stored in the file #happy.csv.

ii) Twitter Data Composition

Every tweet had 16 data fields: *text*; *favorited*; *favoriteCount*; *replyToSN*; *created*; *truncated*; *replyToSID*; *id*; *replyToUID*; *statusSource*; *screenName*; *retweetCount*; *isRetweet*; *retweeted*; *longitude* and; *latitude*. These variables and their description are enumerated in Table 12.

Table 12 – Variables available for each tweet.

Field	Description
text	Contains the tweet's text.
favorited	Assumes true or false values and tells us if someone has marked the tweet as favorite or not.
favoriteCount	Assumes non-negative integers representing the number of times the tweet was marked as favorite.
replyToSN	Is the screen name of the user the tweet is a reply to. If the tweet is not a reply it assumes the value NA (Not Available).
created	Is the date and time, with seconds resolution, the tweet was posted.
truncated	Assumes true or false values. This variable tells us whether or not the tweet's text was cut short or truncated.
replyToSID	If the tweet is a reply, this variable is the ID of the tweet this is a reply to, otherwise it takes the value NA.
id	Is an 18 digit long number unique to every tweet.
replyToUID	Is the id of the user this tweet was a reply to. If this tweet is not a reply it takes the value NA.
statusSource	Describes the utility used to post the tweet. It can be a specific website or an application.
screenName	Is the username of the tweet's poster.
retweetCount	Is the number of times the tweet was retweeted. It can assume non-negative integers values.
isRetweet	Tells whether or not the tweet is a retweet. It takes true or false values.
retweeted	Assumes true or false values and it tell if the tweet has been retweeted or not.
longitude	Geographic location longitude of the tweet's poster. If not available it takes the value NA.
latitude	Geographic location latitude of the tweet's poster. If not available it takes the value NA.

iii) Twitter Client

As for sentiment tweets, the same procedure is applied for company tweets, as can be seen in Figure 5. The data flow is quite intuitive. Essentially, our Twitter client makes requests to the Twitter API in

order to access tweets with the required search terms, which, in turn, are stored in the appropriate files.

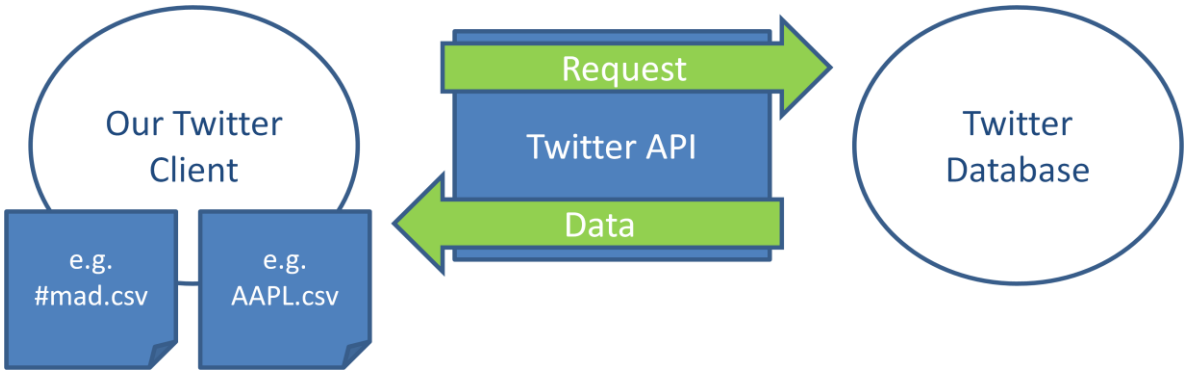

Figure 5 - Twitter data flow.

We collected tweets from a group of companies using their ticker on the stock market prefixed by a dollar sign, “$”. The company's ticker is its symbol, usually three or four capital letters long, on the stock market. For instance, the ticker of the Apple Inc. is AAPL and to download tweets which subject is the Apple company, we used the search term “$AAPL”. On Twitter this is called a cashtag.

We opted to download tweets related to all companies present one the S&P 500, NASDAQ 100, and Dow Jones Industrial Average 30. As most of the companies present in the NASDAQ 100 and the Dow Jones 30 are also present the S&P 500, the list contains 504 unique names (on 2016-05-18).

Analogously to the tweets downloaded with the sentiment words, the company related tweets were stored in a file. Each company’s tweets were stored in its own *csv* file whose name is in the format ticker.csv. For instance, all tweets collected using the Apple company cashtag were stored in the file AAPL.csv in chronological order from beginning to end, i.e. older tweets are at the beginning of the file and the most recent ones are at the end.

The procedures described above, including the Twitter client, were preformed with the R programming language with the aid of the *twitterR* package [89].

3.2.2. 2nd Module: The sentiment model

A text mining task requires many challenging processes to be executed because natural language texts, such as tweets, from a computer perspective, are a highly unstructured collections of words.

On the following paragraphs, we describe a few text preprocessing techniques we tested in this work.

To test the influence of each technique over the classification results, we measure the increase, or decrease, in accuracy on predicting to which of the four classes our test data tweets belong to. In other words, suppose that, after preprocessing, we have 30 000 tweets. Each of these tweets belongs to one of four classes mentioned on Table 11. We use 80% of these tweets, training data, to train an

SVM classifier and test its prediction accuracy on the remaining 20%, testing data. Therefore, the trained model is going to make an educated guess in order to predict the class of each tweet on the test sample and it is this accuracy, or correct predictions, on the test sample, that is mentioned in the text below. We also tried other classifiers such as neural networks, maximum entropy and random forests. However the SVM consistently achieved the best results whilst being reasonably fast. Worth to be noted that each of the four classes should be present in the same proportion on the training sample, if not, the classifier prediction tends to be biased towards the majority class [90], [91].

i) Setup

Figure 6 presents the diagram of the module. The learning algorithm, in this case an SVM classifier, takes feature vectors, the document-term matrix, built from preprocessed tweets from all emotion words with the corresponding labels. The label given to the tweets of each emotion follows the logic presented in Table 11. After the learning process we end up with a model, the Predictive Model, capable of making predictions on unseen tweets. As one can imagine, with thousands of tweets, this process can take a bit of time. For this reason, the model was saved on file and can be loaded and used per request without the need to start the process all over again. Assuming that the words present in one emotion's tweets and absent from others do not change over time and that the model we originally built incorporated tweets that were a good sample of all tweets population for every emotion, it is not necessary to rebuild the model.

ii) Preprocessing Tasks

Because the tweets' text content is highly unstructured we need to process it before it is turned into feature vectors. The following paragraphs describe all the preprocessing techniques we performed as well as the reasoning behind them. These explanations are in fine detail and can be a monotonous read but they are crucial for the inner workings of the proposed system. Among the techniques described is the removal of tweets that are duplicated or tweets posted by users from whom we collected more than a certain number of tweets. Also, we explain how we performed a tweet to tweet comparison, how we removed hashtags that were at the end of the tweet and how we dealt with repeated letters and emoticons. Lastly, we clarify how we built the document-term matrix and reveal the specifics of the custom java function we built to perform the preprocessing techniques.

Duplicated Tweets and Many Hashtags

First, we checked for and removed duplicated tweets. To identify duplicated tweets we used the tweet ID number and simply removed those that were not unique. We also implemented a feature to be able to remove tweets with a number of hashtags above a certain value. The reason behind this is that the main concern of users who use a large number of hashtags is to reach a vast number of people rather than show a genuine emotional state. We found that tweets with plenty of hashtags tend to have more of a marketing function. On the contrary, tweets with few hashtags are more easily spontaneous and

sincere sentiment expressions. Best accuracies however were seen when this restriction was not implemented, although, we still restrict the number of hashtags using a different formula (continue reading).

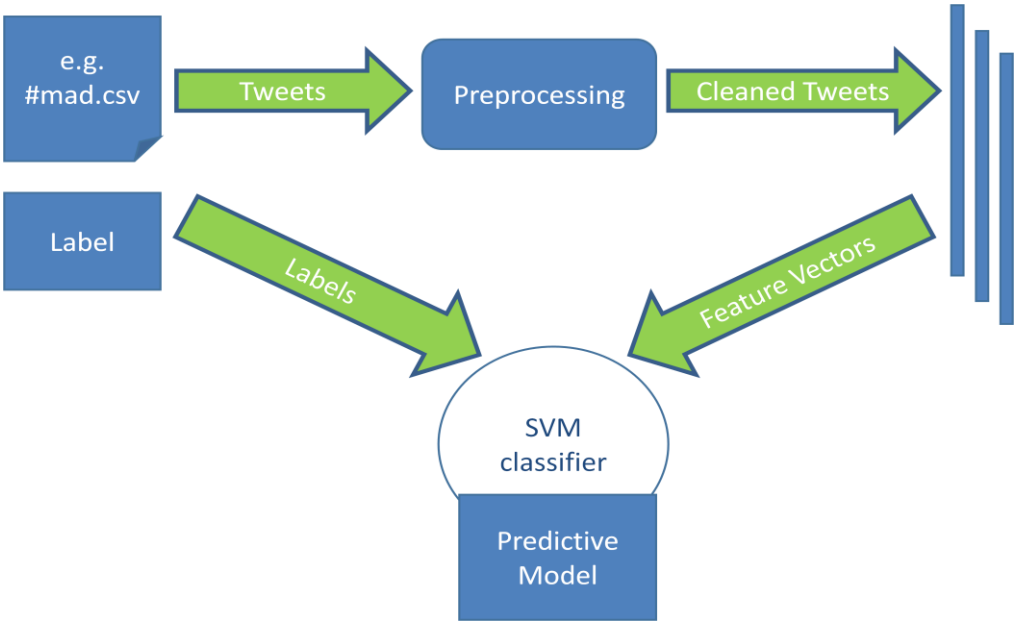

Figure 6 - Diagram of the Sentiment Model.

Harmonization of User Influence

Next, tweets were grouped by the poster's username. On the downloaded tweets we observed that many Twitter users posted dozens, even thousands, of tweets that have very little difference between them. For instance, one user posted dozens of tweets similar to "I am intelligent. #I_AM #positive #affirmation #happy" and "I am thankful for being able to choose. #thankful #positive #happy". Variations of the former tweet can be replicated by changing the word "intelligent" by a number of other adjectives, such as, "harmonious", "thankful", "genuine", "generous", "ok", "independent", "imaginative" and others. Variations of the latter tweet imply replacing the words "being able to choose" by "air fresheners", "vacation days", "being able to see", "coffee", "electricity", "fall leaves" and others. Moreover, after that user runs through all variations he or she posts the same ones again and again, in a loop fashion. The point here is that if we keep these tweets and include them in our classifier, the results of the classification are going to be severely biased by the words used by such users.

In order to equalize the power between Twitter users over the classification results we implemented a rule that allows for the exclusion of tweets posted by users from whom we collected more than a certain number of tweets. Several values between 1 and 15 were tried but accuracy was higher when tweets posted by users from whom we collected more than 3 tweets were discarded. On the other hand, tweets posted by users from whom we collected 3 or less tweets had to go through the following

selection process. We took that set with 1 to 3 tweets, and compared the words present in them. For example, say we collected 3 tweets (A, B and C) posted by user 1. We compare all the words from tweet A to all the words from tweet B and repeat the process for all possible two way combinations, that is, all the words from tweet A to all the words from tweet B, and C and all the words from tweet B to all the words from tweet C. If at least 50% of the words in the shorter tweet were present on the longer, a counter is incremented by one. If that counter ever reaches 2, comparisons are stopped and every tweet posted by that user gets removed from the dataset.

The 50% threshold of the words had to do with accuracy. Other values were tried, namely multiples of 10%, but highest accuracy was achieved with the specified percentage.

We found that this method helped preventing a single user, from whom we collected a large number of tweets, having too much influence on the classification results and if he or she still had an above average number of tweets on our dataset, they must have a somewhat diverse content.

Usernames, Exclamation Points and Question Marks

Next, a number of patterns were identified and removed or replaced from the text of tweets. On Twitter one can identify another person using the character '@' immediately before the username, e.g. @James. All the words prefixed by the char '@' were replaced by “UsernameHere”. This was necessary to avoid a specific username from being, itself, a predictor of sentiment. Likewise, all exclamation points and question marks were replaced by the tags “ExclamationPointHere”, “QuestionMarkHere” respectively. The text replacements used here, were built in such a way that they were both as unique and intuitive as possible. This means that, after preprocessing, when “ExclamationPointHere” was found in a tweet it should be because the pre-processing method put it there and not because the tweet’s poster wrote it. Because all non-alpha-numeric characters, just like exclamation points, question marks and other punctuation, were going to be removed later, such entities needed to be replaced by a text equivalent.

Hastags at the End of the Tweet

Subsequently, hashtags that are at the end of a tweet and are not part on the sentence were striped. So, for example, in the tweet "Wonderful place for an early evening dog walk #peaceful #unspoilt", the hashtags “#peaceful” and “#unspoilt” were removed but in the tweet "#Saakalya wishes you all a #peaceful and #happy long weekend :)" the hashtags “#peaceful” and “#happy” were kept. This helps the classifier not to put too much weight on the hashtags because wanted the classifying algorithm to be able to correctly identify sentiment on company related tweets even if they don’t have hashtags such as “#happy” or “#peaceful”. In other words, we wanted to improve accuracy and generalization. If the hashtags are in the middle of the tweet, they are part of the sentence and essential to understand its meaning, hence, they were not removed.

Links

Many tweets contain links. We considered, like we did with usernames, replacing link occurrences by “urlHere” but this did not improve accuracy, thus, we simply remove them.

Repeated Letters

It is common on Twitter to repeat one letter in a word to accentuate it's meaning, e.g. “joyyyyy”. In this work a letter repeated more than two times was replaced by three occurrences so “joyyyyy” becomes “joyyy”. We chose to replace multiple letter repetitions by three occurrences, instead of just one, so we could differentiate a word’s normal usage from its accentuated usage, hence “joyyyyy” becomes “joyyy” and not “joy”.

Search Term

As proposed in current literature, we tried replacing the search term used to pull the tweet by “QueryTermHere”. So, for example, in the tweet "#Saakalya wishes you all a #peaceful and #happy long weekend :)", used above, if the tweet was downloaded using the search term “#happy”, the same term would be replaced by the mentioned tag, e.i. "#Saakalya wishes you all a #peaceful and QueryTermHere long weekend :)". This way, the classifier wouldn’t use the hashtag itself, used to query the tweet, to classify it and thus helping generalization on company related tweets. However, we found this operation hurt prediction accuracy by around 15 percent points thus suggesting that the search term used to pull the tweet must play a very important role on the classification results.

Special Characters

We also noticed the occurrence of terms such as “&”, “<” and “&gt”, in some tweets. Those terms denote the characters “&”, “<” and “>” and as far as we can tell, nobody knows exactly why the transformation happens, just that it happens sometimes. Every occurrence of the terms “&” and “&”, on the tweets, were replaced by the word “and” while the terms “<” and “&gt” were replaced by the characters “<” and “>”. Replacing the terms “<” and “&gt” is of little importance unless we want to include emoticon features. An example of this is the heart emoticon, “<3”, which sometimes appears as “<3” hence the importance of replacing “<” by “<”. The less than and greater than characters are present in many emoticons, so we must make them explicit.

Emoticons

We compiled an emoticon dictionary from the respective page of Wikipedia[*]. We grouped emoticon into sets or categories. Initially we chose to identify 12 categories with a little over 90 emoticons. The

[*] https://en.wikipedia.org/wiki/List_of_emoticons

categories were happy, laughing, sad, sadness with sarcasm, angry, crying, happy with tears, horror, surprise, wink, cheeky and skeptical. Such categories added no improvement on accuracy and further analysis showed that, on our data, they were very scarcely used. Therefore only the two categories more represented were chosen. The analysis allowed us to conclude that emoticons of categories happy and sad were the most commonly occurring in our dataset, despite with low frequency. Additionally, we found some occurrences of the heart emoticon, "<3", thus, we added it to the search. Table 13 lists emoticons used and the corresponding categories. All the emoticons listed were replaced by "HappyEmoticon", "SadEmoticon" and "HeartEmoticon", according to their category.

Table 13 - List of emoticons used for each category in this work.

Category	Emoticons
Happy	:)) :) :) :D :o) :] :3 :c) :> =] 8) =) :} :^)
Sad	>:[:-(:(:-c :c :-< :< :-[:[:{
Heart	<3

Non-alpha-numeric Characters

To conclude the pre-processing steps, all non-alpha-numeric characters, sequences of two or more white spaces and white spaces at the end or beginning of the tweet were trimmed. Once all tweets were cleaned we had to come up with a criterion to filter most of them out. This was necessary because we had a limited amount of processing power and including all the millions of tweets we collected in our classifier would require a much larger amount of time to train and test models, in order to search for the one that provides best results, while trying different parameter combinations.

To perform the sentiment classification task we used the R package *RTextTools* [92] that can efficiently handle a maximum of 30 000 tweets. Our corpus, or all selected tweets, was used to create a document–term matrix which can be perceived as a table which rows represent the documents, or tweets, and columns represent the terms or single words in this work:

$$D = \begin{pmatrix} tf(t_1, d_1) & \cdots & tf(t_N, d_1) \\ \vdots & \ddots & \vdots \\ tf(t_1, d_l) & \cdots & tf(t_N, d_l) \end{pmatrix} \quad (1)$$

In equation 1 the element $(i,j)^{th}$ of the document-term matrix is the frequency of the term t_j in the document d_i. For example, a document-term matrix for the tweets "This cat hates me too much, too much" and "This stupid cat just woke me up" would have as columns the terms present in all tweets without repetitions and, as rows, the tweets IDs, in this case let us assume the ID for the first tweet is

1 and for the second tweet 2 (Table 14).

Table 14 - Example of a document-term matrix for the tweets “This cat hates me too much, too much” and “This stupid cat just woke me up".

	“cat”	**“hates”**	**“just”**	**“me”**	**“much”**	**“stupid”**	**“This”**	**“too”**	**“up”**	**“woke”**
Tweet 1	1	1	0	1	2	0	1	2	0	0
Tweet 2	1	0	1	1	0	1	1	0	1	1

The document-term matrix used in this project, by the package *RTextTools* [92] is a sparse matrix with a simple triplet format implemented by the package *slam* [93]. This package stores the terms’ frequency in a very efficient and not very memory intensive way. It makes use of the matrix high sparsity (very few non-zero elements) and stores them in a Simple Triplet Matrix format where only the triplets, (i, j, value), i,e, line, column and value, are stored when value is non-zero.

To limit our corpus to 30 000 tweets we set up a ceiling for how many tweets, collected using each one of the 80 search terms, could be included in the model. The following example illustrates how that upper ceiling works. Let’s say we choose 300 as the limit, so, our model can have at most 300 tweets collected using the term “#happy”, at most 300 tweets collected using the term “#sad”, at most 300 tweets collected using the term “#mad” and so on, for all the 80 search terms. Thus, our model will include a maximum of 80 times 300 (24000) tweets.

To choose which 300 from all tweets collected for a single search term would be included in the model we implemented two methods. In the first, we select the 300 tweets with the highest number of retweets plus favorites and in the second, the 300 tweets are selected randomly. We verified that the former method provides better results, with a raise of 1 point percent in accuracy. We noticed the accuracy reaches a plateau at the upper limit 220, so, that was the number we used.

All the previous preprocessing and tweet selection tasks were implemented in a Java method named *readTweetsExtractSpammersR*, which was executed from R through the package *rJava* [94]. The method *readTweetsExtractSpammersR* was designed with 13 arguments that are enumerated in Table 15 with a detailed explanation below.

Table 15 - Arguments used in readTweetsExtractSpammersR for sentiment related tweets.

Variable	Type	Value Used on Sentiment Tweets
word	String	A sentiment word
limitCollectionSize	int	220
maxHashTags	int	0
maxTweetsPerUser	int	3
equalWordsThresh	double	0.5
partiallyEqualTweetsMax	int	2
removeLinks	int	0
replaceSearchTerm	int	0
removeFinalHTfirstAddLinkLater	int	0
replaceQMandEP	int	1
letterRepetitions	int	3
replaceUsername	int	1
replaceEmoticonsSadHappy	int	0
pickTopRetweetsAndFavorites	int	1

Arguments used in our work in the Java method, readTweetsExtractSpammersR.

- **word** expresses the sentiment whose related tweets we want. If the variable **word** is, for example, “happy” then the Java method will open and read the file containing all tweets downloaded using, as search term, the hashtag “#happy”.

- **limitCollectionSize** is the maximum number of tweets, per search term, that are going to be included in the model. The way those tweets are chosen is specified by the variable **pickTopRetweetsAndFavorites** (explained below).

- **maxHashTags** is the maximum allowed number of hashtags per tweet. Tweets with

more than **maxHashTags** hashtags are excluded. As we said before, on our dataset, tweets with an above average number of hashtags appear to serve a marketing purpose rather than expressing a genuine and spontaneous emotional expression. This variable allows us to control the presence of that phenomenon.

- **maxTweetsPerUser** in the maximum allowed number of tweets per user. Tweets posted by users from whom we collected more than **maxTweetsPerUser** tweets are excluded. This variable plays a similar role to **maxHashTags**. Users from whom we collected an above average number of tweets also appear to serve a marketing purpose rather than expressing a genuine and spontaneous emotional expression. The first intention of those users is, usually, to promote a website, a product or themselves. Moreover, these two variables, **maxTweetsPerUser** and **maxHashTags**, are probably somewhat redundant because users from whom we collected an above average number of tweets also use more hashtags, so, removing tweets from users who post many tweets also removes tweets with the most hashtags. Also, the variable **maxTweetsPerUser** is important because it limits the influence of a single user on the sentiment classification results, which ideally should have a very low user bias.

- **equalWordsThresh** is a value used when comparing the similarity between tweets from a single user. Tweets are grouped by user and every possible combination of two tweets are compared. If the ratio (Number of common words in the two tweets) / (Number of words in the shorter tweet) is greater than **equalWordsThresh** a counter is incremented. If that counter ever reaches **partiallyEqualTweetsMax**, all tweets collected from that user are excluded. This variable helps to control the amount of tweets posted by users who write very similar tweets which normally, are intended to promote something.

- **partiallyEqualTweetsMax** is used to count how many very similar tweets are amongst all tweets posted by a user. See **equalWordsThresh** above.

- **removeLinks** controls how the links on tweets are treated. If the value of this variable is 0, links are simply removed, else, if the value is not zero, two things can happen depending on the value of the variable **removeFinalHTfirstAddLinkLater**. If **removeFinalHTfirstAddLinkLater** is zero, links are replaced with the tag “urlHere”; if it is not zero, links are removed temporarily and the tag is concatenated to the sentence later. Links are usually at the end of the tweet and with the link or links at the end it becomes harder to remove the hashtags that are at the end of the sentence, that is, in the tweet "My channel is going well :) #consistency #happy https://t.co/BowQq8pEZO" if the link, or its tag, are there, the algorithm doesn’t properly remove the hashtags “#consistency” and “#happy” at the end because it doesn’t perceive them to be at the end of the sentence due to the link, or links, being there. To resolve this, if the variable **removeFinalHTfirstAddLinkLater** is not zero,

links are removed first, hashtags get the desired treatment and afterwards the link tags are added at the end.

- **replaceSearchTerm** manages how the term used to query the tweet is treated. If the value of the variable is 0, the term is left as it is; if it is 1, the term is removed; if not 0 or 1 the term is replaced by the tag “queryTermHere”.

- **replaceQMandEP** controls the way to handle question marks and exclamation points on tweets. If the value of this variable is 0, they are removed; if the value is not 0, all question marks and exclamation points are replaced by the tags “exclamationPointHere” and “questionMarkHere” respectively.

- **letterRepetitions** control the number of letter repetitions. This is used when, for example, a tweet contains the term “happyyyyyy”. If this variable has any value but 3, the term is replaced by “happy” which is the correct use of the word as defined in the dictionary. If the variable is given the value 3, the term is replaced by “happyyy” with the letter “y” repeated 3 times. This allows to us differentiate the normal use of the word from its emphatic use.

- **replaceUsername** manages how Twitter usernames are handled. If the variable is equal to 1, usernames are replaced by the tag “usernameHere”, else, they are simply removed.

- **replaceEmoticonsSadHappy** controls how emoticons are treated. If the variable is 0, all emoticons are removed; if not 0, they are replaced according to a dictionary (see Table 13) of happy, sad and heart emoticons by “HappyEmoticon”, “SadEmoticon” and “HeartEmoticon”, respectively.

- **pickTopRetweetsAndFavorites** controls the way to limit the number of tweets to be included in the model, per search term. If the variable is 1, tweets with most favorites plus retweets are chosen, else, tweets are chosen randomly.

iii) Feature Vectors

Before we fed Twitter data into the learning algorithm we performed a couple of extra tasks. Here, by Twitter data, we mean our corpus, which contains all sentiment tweets we selected among those downloaded, and their respective labels.

The first task was to build the feature vectors, in this case, a document-term matrix, as shown in Table 14. In this matrix, the rows represent documents and the columns represent terms. The value on the i^{th} row and j^{th} column indicates the number of times the j^{th} term appears in the i^{th} document, so, every element on the matrix can assume non negative integer values. When this is the case, it is said we have a term frequency weighing. The elements on the matrix are numeric counters of terms in documents. However, there are other techniques, or weighing functions. The most common, and the ones we tried, are Binary weight, weight by Term Frequency - Inverse Document Frequency and

weight by SMART notation [95]. SMART (System for the Mechanical Analysis and Retrieval of Text) is a system for information retrieval for text created by Cornell University. In this work, the best results were achieved using Term Frequency - Inverse Document Frequency weighing (TF-IDF) which, for explanation purposes, can be broken up in two parts as seen in formula 2.

$$TF - IDF = f_{t,d} \times log \frac{N}{n_t} \quad (2)$$

The term frequency (TF) $f_{t,d}$ is merely a measure of how frequently a term, or feature, occurs in a tweet. The inverse document frequency (IDF) $\log \frac{N}{n_t}$ measures how important a term is across multiple tweets. It is calculated as the logarithm of the total number of tweets divided by the number of tweets containing the term. Thus if a term occurs many times in a tweet, it gets a high TF score, but, if that term appears in most tweets, it gets a low IDF score. In the specific case where a term appears in every tweet, it's IDF score is zero, or, in other words, if every tweet has a specific term, no information can be gained by its presence, so, the term gets a zero IDF score and it is the same as if the term was not present at all. The TF-IDF is the multiplication of TF by IDF therefore, even if a term has a high TF weight, it can become less discriminative after the multiplication by the IDF weight.

To create a document-term matrix we used the function *create_matrix* of the *RtextTools* package [92]. This function, besides documents (tweets' text) and the corresponding labels, takes in account a few other arguments listed below. The values used for each of them, also included in the list, were chosen with the goal of maximizing accuracy.

Arguments used in the function *create_matrix*.
- language is, intuitively, the language to be used in the text mining process. This is only necessary if we want to stem the text data (see **stemWords** below). We used English language. **- minDocFreq** and **maxDocFreq** is the minimum and maximum number of times a word must appear in a document to be included in the document term matrix, respectively. We set the first equal to 1 and the latter to Inf (Infinite). **- minWordLength** and **maxWordLength** are the minimum and maximum number of letters a term, or word, must contain to be included in the document term matrix, respectively. We set the first equal to 3 and the latter to Inf. **- ngramLength** is the number of words a n-gram should have. We set this variable equal to 1. **- removeNumbers** is a boolean variable. When set to true, the value we used, all numbers are removed. **- removePunctuation** is a boolean variable. When set to true all punctuation characters are removed. We set this variable to false not because we want to keep the

punctuation but because all punctuation features were dealt with before on the preprocessing steps. They were either removed or replaced by an explicit tag.

- removeSparseTerms is a logical variable specifying whether sparse terms should be removed. Sparse terms are tokens or terms that are used very infrequently. The sparsity of a term refers to the frequency that it is used, not within each document, but in the entire corpus. Its numerical value can be easily obtained, even more so if we already have a document-term matrix. The sparsity of a term is the total number of documents that contain that term divided by the total number of documents, so, if a term has sparsity 1.0 it means the term appears on all documents, i.e. it has a very small sparsity.

The goal of this text mining technique, i.e. ignoring terms that appear only in very few documents, is to prevent overfitting and help generalization. To explain this further, let's say we have ten tweets that are divided into two classes, 1 and 2. Now, the word "cat" appears in one and only one of those tweets and it belongs to class 1. If we train a classifier with those tweets it is probably going to create a very strong link between the word "cat" and the class 1 where in fact, it could be an ungrounded association. If we had ignored the sparse term "cat", we minimize the risk of classifying every tweet containing the word "cat" as a class 1, just because it has the word "cat" in it, thus helping generalization.

In this work, because we have a very large number of documents containing very short texts, very distinct among them, we can assume that we will have a large number of very sparse terms. Indeed, best results were achieved when this variable is set to zero, i.e., there is no term removal based on sparsity.

- removeStopwords is also a boolean variable. When set to true very common words, of the specified language, are removed from the text data. The term stop word is used to refer the most commonly used words in a language. Because they are so common, it is assumed they have zero emotional weight and hence are removed. Eliminating words such as "and", "the", "this", "to", "a", "that", "too", "so" and "or" can bring a significant increase in precision in that, more information can be gained considering just the remaining words in the sentence rather than considering all the original words. The adequate stop words list for removal can be very application dependent. There are lists with just a few words and others with hundreds. As we already mentioned, the stop words list we used is the one by SMART [95], which comprises 571 words. It is a longer than normal list for this purpose, but it is the one that provides best results. It is probably fruitless to write all 571 words here but, in the SMART stop word list, alongside the nine words mentioned above, are words such as "up", "two", "see", "new", "none", "like", "my", "gone" and "etc".

- stemWords is a logical variable specifying whether to stem words, according to the

chosen language in the variable with the same name above. Stemming, in linguistics, is the process of reducing a word to its stem or root. That stem or root doesn't need to be a valid word, as defined in the English dictionary, it just has to map or be a representation of a collection of related words, for example a stemming algorithm should identify the words “fisher”, “fishing”, “fished”, “fisherman” and replace them with the word “fish”. Stemming the words "argue", "argued", "argues", "arguing", and "argus" results in the stem "argu", which is not a valid word but is a valid stem. This is a common technique that reduces complexity, without any significant loss in entropy, because it performs a dimension reduction (the number of columns of the document-term matrix gets smaller). Word stemming has been subject to many studies. One of the best known stemming algorithm is the Porter's stemmer [14] used in this work.

- stripWhitespace is a boolean variable. When set to true, extra white spaces are removed. We don’t need this because the procedure is done on the previous preprocessing steps in Java.

- toLower is a logical parameter to specify whether to make all text lowercase. Also, this is done on the previous preprocessing steps in Java and it is a necessary step so that the same words don’t become two different features. For example, if we kept the text as it is, one tweet with the word “Twitter” and another with the word “twitter” would be considered different features when, in fact, we want them to be the same.

- weighting specifies the function we want to use for term weighting. In this case we use the Term Frequency - Inverse Document Frequency weighting function from package tm [96].

iv) Training The Sentiment Model

After all the previous steps were taken building and training the sentiment model is the simplest part. We used the function *train_model* of the *RtextTools* package with the feature vectors, or, the document-term matrix, and the learning algorithm as parameters. The learning algorithm used was the SVM implemented on the package *e1071* [97]. Lastly, the model was saved in a file.

The goal was, after training the classifier with the first set of tweets (sentiment tweets), to use it on tweets from a specific company, to assert which of those four sentiment classes prevails or assumes a dominant presence. The same was done for all companies and their performance on the stock market was compared.

3.2.3. 3rd Module: Company Tweets Classification

This module uses the model built and learned, from the tweets collected using the 80 sentiment words, to classify tweets from each company.

i) Preprocessing

As shown in Figure 7, company's tweets were read from the corresponding files and preprocessed. These preprocessing steps were the same as those applied on sentiment tweets that are performed by our custom Java function, with different parameters, and the *create_matrix* function from package *RTextTools*. The parameters used on company's tweets with the function *readTweetsExtractSpammersR* are listed on Table 16.

Performing all preprocessing steps with the proposed Java method takes about 2 seconds for 3000 tweets and 12 seconds for 5000 tweets. To create the document-term matrix with the R function takes about 20 seconds for 2000 tweets and 2 minutes for 3000 tweets.

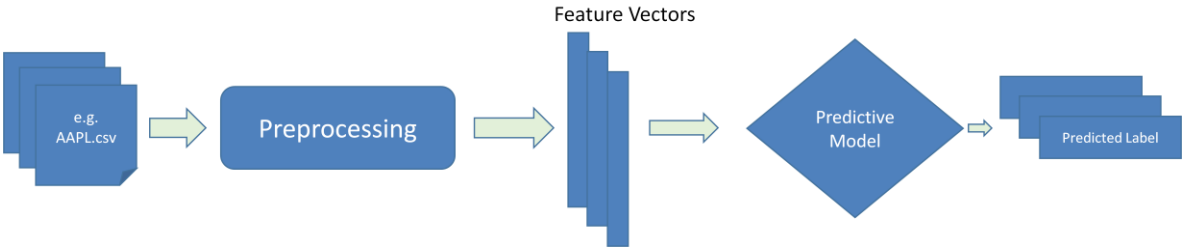

Figure 7 - Sentiment prediction on company's tweets.

Table 16 - Arguments used in readTweetsExtractSpammersR for company related tweets.

Variable	Type	Value Used on Company Tweets
word	String	A company ticker
limitCollectionSize	int	0
maxHashTags	int	3
maxTweetsPerUser	int	15
equalWordsThresh	double	0.5
partiallyEqualTweetsMax	int	1
removeLinks	int	0
replaceSearchTerm	int	0
removeFinalHTfirstAddLinkLater	int	0
replaceQMandEP	int	1
letterRepetitions	int	3
replaceUsername	int	1
replaceEmoticonsSadHappy	int	0
pickTopRetweetsAndFavorites	int	1

ii) Tweet Selection

As said before, each downloaded tweet has a data field called *created*. That field is the UTC (Coordinated Universal Time) time at which the tweet was posted. We also know that the companies we selected are traded on the NYSE (New York Stock Exchange) and NASDAQ. Trading hours in those exchanges happen between 9:30 and 16:00 at New York time. The timezone in that city is EST (Eastern Standard Time) in the winter and EDT (Eastern Daylight Time) in the summer. During our training period the time zone in effect was EDT which corresponds to UTC -4.

To predict the price movement on any business day, we gauged the sentiment from tweets posted in the previous non trading hours. Suppose we want to predict the price movement (up or down) of the Google stock or, put simply, we want to know if the closing price at 16:00 EDT is bigger or smaller than the open price at 9:30 EDT. For that, we measure the sentiment on Google tweets posted between the previous market close time, i.e. 16:00 EDT on the previous day, and the current day market open time, i.e. 9:30 EDT which corresponds to tweets posted between 20:00 UTC and 13:30 UTC respectively. This means that only tweets posted outside trading period were used for prediction, those others posted during trading hours were excluded. Furthermore, in order to predict price movement on a Monday the same method is applied, that is, we used tweets posted between 16:00 EDT of the previous Sunday and 09:30 EDT Monday.

Our simulations suggested we should reduce the off trading period even further. Although the NYSE and NASDAQ exchanges close at 16:00 EDT, we do not include tweets posted during the hour immediately after that because our simulations show that excluding them leads to better results. Also, the exchanges open at 09:30 EDT, but we stop at 09:27 to allow 3 minutes time for processing and submitting the eventual purchasing orders to the stock market.

Concluding, tweets used for price prediction in our simulations were all posted between 17:00 EDT and 9:27 EDT.

iii) Output

With the selected tweets we built a document-term matrix with the function *create_matrix*, just as the one used on sentiment tweets with the same parameters. Then the document-term matrix was fed into the classifier model we built. For each tweet fed into the sentiment model we got two output values. One is the class attributed by the classifier to the tweet and the second is a number between zero and one that measures the certitude of the classifier on the first value. This implies that if the second value is 1 the classifier is 100% sure the tweet belongs to the class specified by the first value. The first value was named SVM_CLASS and the second SVM_PROB.

As for performance metrics, it takes 5 seconds to classify 2000 tweets and about 12 seconds for 5000 tweets.

3.2.4. 4th Module: Stock Selection

So far we have a collection of companies, their related tweets and corresponding predictions, thus a method is necessary to transform this into buy signals for the stock market.

i) Remove Neutral Tweets

Table 17 is an example of how the data was available to us from module 3. Notice that, at this stage, we only have tweets that were posted during off trading hours. Also, an important point is that there will be tweets whose sentiment is hard to discern, even if they were labeled by a human. However, independently of how hard it may be to understand the sentiment shown on a tweet, our classifier will most certainly have a favorite class to attribute to any given tweet. For example, given a tweet, the classifier can decide that there is 25.1%, 25.0%, 25.0%, 24.9% chance it belongs to classes 1, 2, 3 and 4 respectively, so, the tweet will be given the values 1 and 0,251 to SVM_CLASS and SVM_PROB respectively, when, in fact, for practical reasons, this situation should be called a draw. Moreover, there are, indeed, tweets whose sentiment is neutral and a rule needs to be created to account for and exclude tweets whose sentiment is hardly obvious. To achieve this goal, tweets whose SVM_PROB is less than a specific threshold are excluded. That threshold is going to be mentioned again later but, for now, suppose that it is equal to 0.5. This means that a tweet's sentiment is only going to be used if the classifier is at least 50% sure about its sentiment class, so, on the example in Table 17, tweet 1 and tweet 4 would be eliminated.

Table 17 - Typical data table produced in module 3 in the used architecture.

Tweet	SVM_CLASS	SVM_PROB
Tweet 1	1	0.37
Tweet 2	4	0.66
Tweet 3	3	0.91
Tweet 4	3	0.44
Tweet 5	1	0.86
Tweet 6	3	0.52

ii) Daily Data Congregation

Each day, when we want to pick the best companies to buy, we take all tweets posted on the previous off trading hours (between 17:00 EDT on the day before and 09:27 EDT on the current day), as explained in the 3rd module, and summarize them in 13 values: 4 for the number of tweets belonging

to each SVM_CLASS; 1 for the total number of tweets; 4 for the average SVM_PROB values for each class and; 4 for the percentage of tweets from each SVM_CLASS. Those 13 values refer to a single period from 17:00 EDT to 09:27 EDT and were calculated for each company in our set. Using Table 17 as an example, we constructed Table 18 with the 13 variables.

Table 18 - Daily Summary Table of the companies.

Day, e.g., 2016-06-23, Company, e.g., Apple Inc.	
Number of tweets with SVM_CLASS 1	1
Number of tweets with SVM_CLASS 2	0
Number of tweets with SVM_CLASS 3	2
Number of tweets with SVM_CLASS 4	1
Number of total tweets	4
Average SVM_PROB of tweets with SVM_CLASS 1	0.86
Average SVM_PROB of tweets with SVM_CLASS 2	0.0
Average SVM_PROB of tweets with SVM_CLASS 3	(0.52+0.91)/2 = 0.715
Average SVM_PROB of tweets with SVM_CLASS 4	0.66
Percentage SVM_CLASS 1 tweets over total number of tweets	$\frac{1}{4} = 0.25$
Percentage SVM_CLASS 2 tweets over total number of tweets	0.0
Percentage SVM_CLASS 3 tweets over total number of tweets	$\frac{2}{4} = 0.5$
Percentage SVM_CLASS 4 tweets over total number of tweets	$\frac{1}{4} = 0.25$

Table 18 summarizes how the data was prepared. The values given to each variable were calculated based on the values on Table 17. The proposed setup is to have a table, just like Table 18, for each and every company we are looking into. The decision of which ones to buy is going to be made according to the values on those tables. The same procedure is taken every day we want to choose

what the best companies to buy are.

As we can observe, there is a date at the top of the Table 18. Although the table summarizes tweets posted from a 17:00 to 09:27 period, that is, tweets posted on two days, one being from 16:00 to 23:59 of a day and the other from 00:00 to 09:27 of the next day. The date at the top of the table always refers to the first one, so, if we want to predict the price movement on a particular date, the summary table will have the date of the previous day, even if no tweets were posted during that time.

iii) The Trading Rule optimized by a Genetic Algorithm

We arrive here with a summary table (see Table 18) for each of the 504 companies. Now we need a way to choose which of those companies to buy and which to leave well alone on any one day.

Keep in mind that, through all the steps described above, such as limiting the number of hashtags per tweet or limiting the number of tweets per user or removing tweets with a low SVM_PROB or a low retweet count, amongst others, we have seriously reduced our tweets sample size. Depending on the company, the number of tweets used to make the prediction may be, and usually is, more than fifty folds smaller than the total number of collected tweets for the period for that company. This is not that surprising if we consider that Twitter data can be very noisy, there is little information in it or, more precisely, there is a lot of contradictory information from a large number of sources. Additionally, the same company can have the summary table related to 100 tweets on one day and 0 tweets on the day after (in such case all values on the summary table would be 0). The point is that the summary table can be quite different from company to company and from one day to the next, so, we need a flexible rule that is able to pick the better companies to buy using very divergent tables.

To design a trading rule we created 5 variables (***Minimum_Number_of_Tweets***, ***Minimum_Number_of_Companies***, ***Number_of_Companies_to_Buy***, ***Variable_on_Summary_Table*** **and** ***Minimum_Probability***). ***Minimum_Number_of_Tweets*** and ***Minimum_Number_of_Companies*** are integer numbers representing a number of tweets and a number of companies respectively. ***Number_of_Companies_to_Buy*** and ***Variable_on_Summary_Table*** are also integer numbers that refer to the number of companies to buy and one of the 13 variables in the summary table respectively. Lastly, ***Minimum_Probability*** is a decimal number that represents a probability. Table 19 lists all values these variables can take.

The values of these variables are essential to the functionality of the system. The trading idea is explained below.

Suppose that on a particular day we want to decide which companies to buy and what we have from each of them is the summary table. Then, we take the tables of all the companies and subject them to a set of criterions:

> **Criterion 1 -** All tweets with an SVM_PROB that is less than ***Minimum_Probability*** are excluded and only the ones that remain enter in the following calculations.
>
> **Criterion 2 -** If a company has on its summary table ***Minimum_Number_of_Tweets*** or less

tweets, the company is excluded.

Criterion 3 - If, for a given day, the total number of companies that passed **Criterion 2** is less than or equal to ***Minimum_Number_of_Companies*** no company is selected, that is, no buying signal is produced for the day.

Criterion 4 - From the list of companies that passed **Criterion 1**, **Criterion 2** and **Criterion 3** we select the ***Number_of_Companies_to_Buy*** with the highest values of ***Variable_on_Summary_Table***.

To calculate the best values for ***Minimum_Number_of_Tweets***, ***Minimum_Number_of_Companies***, ***Number_of_Companies_to_Buy***, ***Variable_on_Summary_Table*** and ***Minimum_Probability*** we used a genetic algorithm implementation. The chromosome contained the values for ***Minimum_Number_of_Tweets***, ***Minimum_Number_of_Companies***, ***Number_of_Companies_to_Buy***, ***Variable_on_Summary_Table*** and ***Minimum_Probability*** where each of the 5 variables is a gene with the values that are to be optimized.

In order to implement a genetic algorithm optimization we used the *rgenoud* R package [98]. The only built in function in this package is the *genoud* function. This function takes in over 40 arguments and explaining all of them here would make this text unnecessarily long, instead, we explain just those few arguments that are more relevant to the problem under analysis (pick which stocks to buy).

Arguments used in the *genoud* function.
-**fn** specifies the fitness function. In our case, the fitness function returns a scalar representing the total profit/loss over the period.
-**nvars** is the number of parameters to be optimized (***Minimum_Number_of_Tweets***, ***Minimum_Number_of_Companies***, ***Number_of_Companies_to_Buy***, ***Variable_on_Summary_Table*** and ***Minimum_Probability*** in this case).
-**max** is a logical variable indicating whether the optimization procedure should maximize or minimize the fitness value. In this instance it is set to true, that is, maximize profit.
-**pop.size** is the number of individuals that is used to solve the optimization problem at each iteration. We used 30 individuals.
-**max.generations** is the maximum number of generations that the function will run to optimize the problem. We set this variable equal to 100.
-**wait.generations** tells the *genoud* function the solution was found if a better solution is not reached after the number of generations specified by this variable. We used 15 generations for this variable.
-**starting.values** gives the *genoud* function the parameter combination it will use at

startup. This is not strictly necessary as the function is able to randomly create individuals as needed. The starting individual we used was [1, 1, 1, 2, 0.5].

-**MemoryMatrix** is a logical variable that, if true, assures the *genoud* function will request the fitness value of a particular combination of parameters only once. This allows it to save processing time but, on the other hand, it consumes more memory (necessary to save the matrix). We used true for this variable.

-**Domains** is a matrix with **nvars** rows and 2 columns, that is, a line and two columns for each variable. The values in the first column are the lower limit for the respective variable and the second column has the upper limits. By default data types are integers, this means that parameter values will be incremented or decremented by multiples of 1. In this specific case, the domain used for each variable, or their range, is listed on Table 19.

Table 19 - Domains of the customized variables used with the *genoud* function.

Variable	Domain
Minimum_Number_of_Tweets	[0 - 20]
Minimum_Number_of_Companies	[1 - 20]
Number_of_Companies_to_Buy	[1 - 10]
Variable_on_Summary_Table	[1 - 13]
Minimum_Probability	[0.3, 0.4, 0.5, 0.6, 0.7, 0,8, 0.9]

From Table 19 we can see that the search space or, the number of possible solutions is $21 \times 20 \times 10 \times 13 \times 7 = 382200$.

Changing the value of the variable ***Minimum_Probability*** entails loading the classified tweets from file and recalculating the summary table for all the 504 companies for every day of our training period. Just this task takes about 8 minutes for 2000 tweets. That list, of 504 summary tables for every day of the period, is unique for each value of the variable ***Minimum_Probability*** so it is saved in memory and calculated only once.

Other variables could be included in the in the GA optimization such as the maximum number of hashtags per tweet or maximum number of tweets per user, however, that would require the tweets to be preprocessed and classified all over again. Loading the original tweets from file, preprocessing them and classifying them takes about 2 minutes per day or about 2.5 hours for the whole period. To give a rough idea, it would take at least 2.5 hours per each GA individual and if we consider a GA with a population of 30 individuals that runs for 50 generations the processing time required to complete

the task would be too long. Still, if one has access to a faster hardware setup, the R code can be easily altered in order to accommodate other variables.

Besides the referred arguments the *genoud* function has 9 other arguments that control cloning, mutation and crossover. For those we used the default values. Also, not captured in this brief explanation, are the intricate relationships between some arguments [98].

3.3. Chapter Conclusions

On this chapter we provided a full description of the proposed system. The paragraphs above give a good understanding of how the proposed solution is implemented and what were the reasons behind each step and each parameter choice, of which there were many. Ideally, every parameter values should be subjected to some kind of optimization technique such as a genetic algorithm but with the vast array of variables mentioned, from number of n-grams and minimum word length to maximum number of hashtags per tweet and SVM_PROB threshold, the testing of all possible combinations would be overwhelming. Instead, we tried to make choices based on what seems more logic and, of course, trying the parameters that gave the best results on other studies and stuck with those that gave best results on our work. Furthermore, almost every value used can be easily modified just by opening a file, so that a different parameter combination can be assayed.

Chapter 4. System Validation

In this chapter we present the results that we produced using the proposed system for trading on the stock market. We present the data of three different case studies. Each case study implements the system in a slightly different way. To evaluate the performance in each case we propose a few evaluation metrics which we describe below.

Case study 3 is our best solution as given by the genetic algorithm but is also riskier (in the training period) because the whole investment is concentrated in only one company's stock. Case studies 1 and 2 are also the best solution provided that we distribute the investment in two different stocks. The details of the optimum values of the parameters are explained below.

4.1. Simulation environment

Stock prices were obtained from Google Finance and return plots were created using the R package *PerformanceAnalytics* [99]. Our training period starts at May 18 and ends at September 2, 2016. We used tweets posted during this period to learn the best parameter combinations and to create 2 strategies. Case study 1 and 2 is the implementation of the first strategy on our training and testing period respectively. Case study 3 applies the second strategy also on our testing period.

There are stocks on the S&P 500 index that are a lot cheaper than others, for example, a Staples Inc. stock costs, at the time of writing, 8.36 US dollars while an Amazon.com, Inc. stock is priced at $816.11. In this work we assumed we have the purchasing power to buy any stock and that we are able to distribute our entire account balance equally between stocks. The minimum holding period for any stock is one day, that is, buy at open and sell at close time in the same day. In the case where we sell more than one company in a single day, that we previously bought, the returns obtained from each are averaged. Additionally, the amount invested on any given day is the amount we have left from the previous trading session. This means that all profits are reinvested and if we lost 50% of our account balance with previous trades it is the remaining 50% we will invest in the next trades.

We search for opportunities to enter the market on a daily basis and when we do enter it was always on long positions. Furthermore, dividends were not included in the calculations.

Every stock we acquire was bought at market open (time and price) and every position we close was done at market close (time and price). Thus, if we buy a stock at open and sell it at close on the same day our holding period is one day.

4.2. Evaluation Metrics

In order to evaluate the performance of the proposed solution we used different metrics. Return on investment, drawdown, number of trades and Hit Ratio and lastly variance and risk measures.

4.2.1. Return On Investment (ROI)

The ROI relates the amount invested to the amount gained or lost by the investment according to the

formula 3.

$$ROI = \frac{(Exit\ Price - Entry\ Price)}{Entry\ Price} \quad (3)$$

$$ROI(\%) = ROI \times 100 \quad (4)$$

So, for example, if we buy a stock today for $50 and sell it tomorrow for $100 we end up with a ROI of $\frac{(100-50)}{50} \times 100 = 100\%$, which means we doubled our initial investment.

Using the ROI value of the proposed solution alone is not sufficient to understand the quality of its performance. It is necessary to compare it to other strategies ROI. In this work, in each case study, we compare the obtained ROI with the one obtained on some indexes, namely, the S&P 500, NASDAQ 100 and the DJIA. The ROI that will be shown for those indexes supposes that one buys the index and holds it for the specified period (the buy & hold strategy) while the ROI for our case studies assumes many entries and exits on the market.

Figure 8 and Figure 9 show examples of ROI curves over different periods of time. The curves are plotted using daily return values, in other words, returns are calculated between consecutive days' close prices using the formula (*Close Price on Day t - Close Price on Day t-1*) / *Close Price on Day t-1* until we reach the end of the period. To calculate our ROI at any given time we just have to add 1 to the daily returns since the initial investment multiply them and subtract one.

For example, suppose we buy stocks at open time and sell them at close time every day for a week. The returns for each of the 5 days is 0.9(90%), 0.1(10%), 0.2(20%), 0.3(30%) and -0.9(-90%), respectively. Then the ROI for the week is:

- Add one to each of the returns: 1.9, 1.1, 1.2, 1.3, 0.1

- Multiply all terms from previous result: $1.9 \times 1.1 \times 1.2 \times 1.3 \times 0.1 = 0.32604$

- Subtract 1 to the previous result: 0.32604 – 1 = -0.67396 or -67.396%.

As we can see in Figure 8 and Figure 9, the curves are quite different. Actually they are both in respect to the S&P 500 but for different periods. Despite ending with a profit, the return on investment curve on Figure 8 has a sizable downswing (see drawdown below) which is undesirable because when the line drops below 0.0 the investor is losing money. On the contrary the curve on Figure 9 is indicative of a healthy investment with a profit of a little over 10%. Even more so considering the short 3 month period.

Figure 8 – Example ROI curve 1.

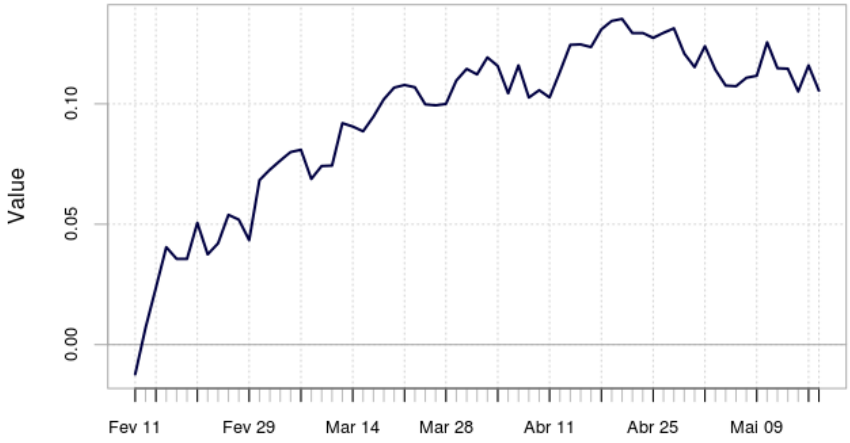

Figure 9 - Example ROI curve 2.

4.2.2. Drawdown

The drawdown records the amount lost from the previous high to the current price. For example, if we have a stock that, at day 1, is priced at $100 and at day 2 drops to $50 we have a drawdown of 50%. If, on day 3, the price rises to $80 the drawdown is measured at 20%. If, at day four the price drops to $20, the drawdown is 80% (the percent amount lost from the last high, $100, to the current price, $20). Ideally the drawdown value should be as close to zero as possible since, if it is zero, it means the price is rising to new highs, which is what the investor expects.

Figure 10 and Figure 11 are the drawdown curves for the corresponding return curves in Figure 8 and Figure 9, respectively. As expected, by definition, the investment shown in Figure 10 suffers a downswing in the end of June, that is, the price drops sharply from the previous high which means the drawdown is going to hit new lows. Besides, the peak drawdown in Figure 11 is half of that represented in Figure 10 which, again, is indicative of a better investment.

Figure 10 - Drawdown for return series 1.

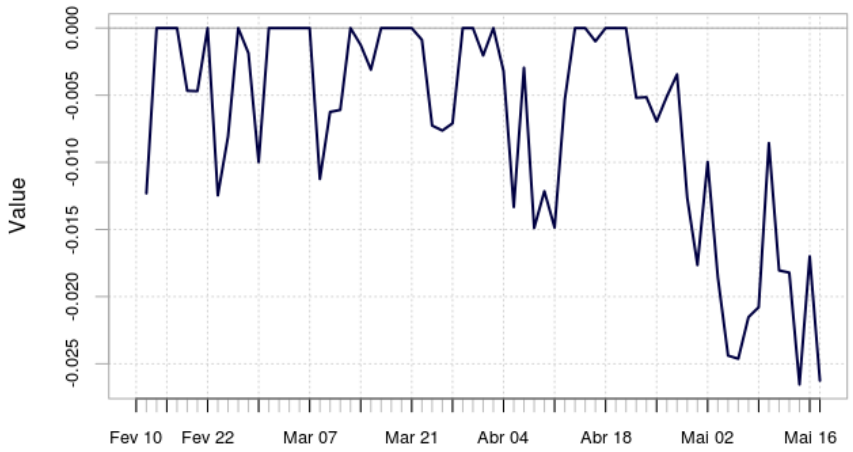

Figure 11 - Drawdown for return series 2.

4.2.3. Number of Trades and Hit Ratio

The number of trades is self-explanatory. The bigger this number is, the more confident we can be that the final profit (or loss) is due to the system's merit and not due to luck. In other words, when the number of trades is large and the system still is able to bank a profit, we can state, with statistical significance, that the system is lucrative. The downside, and not by any means a small one, of a large number of trades is that the value lost in broker commissions also gets bigger.

The hit ratio is the number of lucrative trades divided by the total number of trades.

4.2.4. Variance and Risk Measures

The following histograms refer to the daily returns of the same S&P 500 price series (Figure 8 and Figure 9) used in the examples above. These plots show the Risk Measures for the two examples. In dashed lines are the value at risk (VaR) and the modified value at risk (ModVaR) [99]. The VaR for a

confidence level, 99% in this case, estimates the amount an investment can lose given normal market conditions. In other words, from the dashed line in Figure 12, we estimate that the maximum amount we can lose on any given day is around 1.2% (-0.012 return on the plot) with a 99% confidence level. The ModVaR has the same meaning as VaR but it assumes the underlying return distribution is not normal and takes its kurtosis and skewness into consideration. If, for example, a return series distribution has many outliers and is far from following a well-shaped normal distribution the ModVaR value is going to be far apart from the VaR value. The ModVaR for the return series 2 in Figure 13 is around 3.2% (-0.032 rate of return on the plot) with a 99% confidence level.

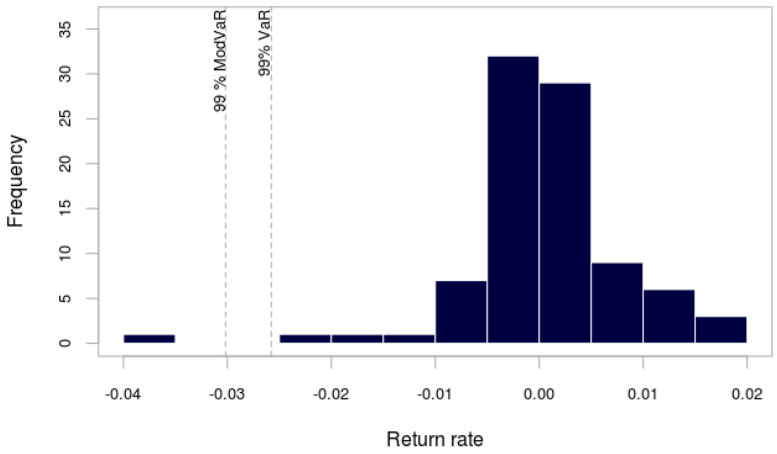

Figure 12 - Histogram and risk measures for return series 1.

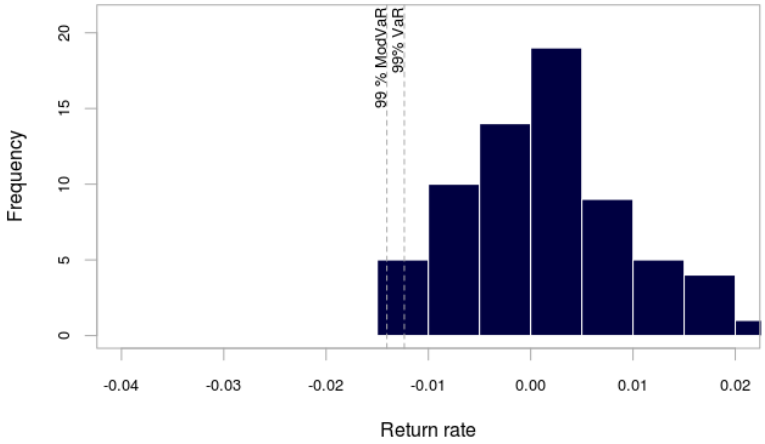

Figure 13 - Histogram and risk measures for return series 2.

4.3. Results

4.3.1. Case Study 1

i) Key Parameters

In this case study we used the values that were optimized by the GA which are 11, 20, 2, 6, 0.3 for the values of ***Minimum_Number_of_Tweets***, ***Minimum_Number_of_Companies***, ***Number_of_Companies_to_Buy***, ***Variable_on_Summary_Table*** and ***Minimum_Probability***. This means that we buy 2 companies with the higher values of the variable 6 on the summary table provided that there are at least 20 companies with more than 11 tweets on their summary table with an SVM_PROB greater than 30%. The variable 6 on the summary table is the *Average SVM_PROB of tweets with SVM_CLASS 1*. Sentiment class 1 includes emotion states such as excellent and happy. Key simulation parameters are listed on Table 20. The results that are presented refer to the ones obtained during the training period.

Table 20 - Simulation parameters for case study 1.

Parameter	Value
Trading frequency	Daily
Holding Period (Days)	1
Minimum_Number_of_Tweets	11
Minimum_Number_of_Companies	20
Number_of_Companies_to_Buy	2
Variable_on_Summary_Table	6
Minimum_Probability	0.3

ii) ROI

The ROI obtained in this case study is presented in Figure 14 alongside the cumulative returns of the S&P 500, NASDAQ 100 and DJIA. In the September 2 the ROI was 30.18%, 6.49%, 11.02% and 5.49% for the proposed solution, S&P 500, NASDAQ 100 and DJIA, respectively. As we can see, the proposed solution outperforms the other indexes.

During the first week of the period the proposed solution starts with a few negative returns, but from mid June onwards the proposed solution ROI starts to be very distinct from the ROI of the other indexes ending the period with more than twice the ROI of the second best performer.

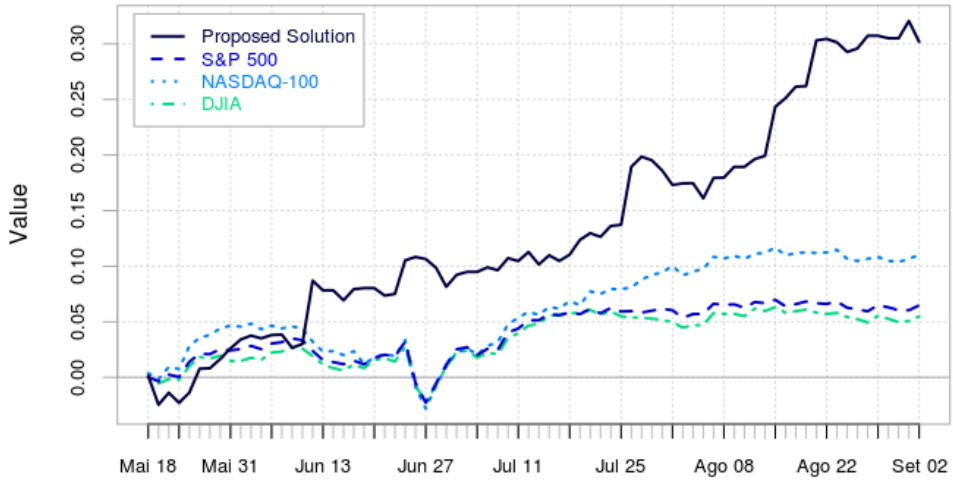

Figure 14 - Cumulative rate of return for case study 1.

iii) Drawdown

The drawdown in this case study and the drawdowns of the S&P 500, NASDAQ 100 and DJIA are plotted in Figure 15. The peak drawdown for the proposed solution is -2.72% that is better than the drawdowns of the DJIA at -5.00%, S&P 500 at -5.79% and NASDAQ 100 at -7.64%. From the peak drawdown values during this period we can conclude that our proposed solution is not riskier than investing on the indexes and doesn't make our investment more susceptible to the occasional increase of volatility of the stock market (Brexit election on June 23).

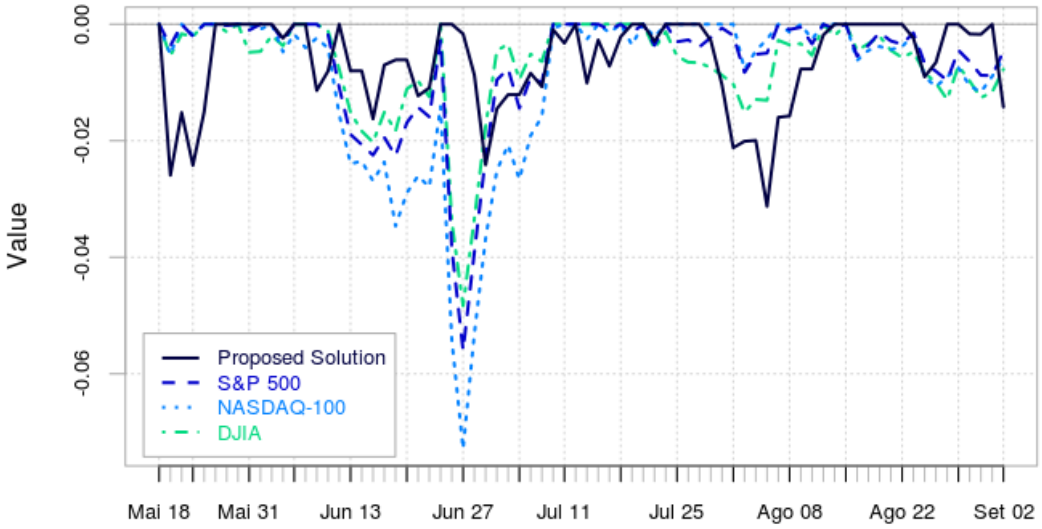

Figure 15 - Drawdown for case study 1.

iv) Hit Ratio and Number of Trades

Our proposed solution entered the market in 72 days out of 76 business days in the period. Each day

two different stocks were bought which means we made a total of 144 trades. The number of trades with positive, negative and zero return is 83, 60 and 1 respectively. This corresponds to a hit ratio of 83/144 = 57.64%. Of the 72 days on the market, 66.66% ended with a profit. On these business days, the daily returns of the S&P 500, NASDAQ 100 and DJIA were positive, 53.95%, 63.16% and 51.32%, respectively. This supports our conclusion that the proposed solution indicates good performance by ending two thirds of the trading days with a profit.

Please note that broker commissions were not accounted in this study. However, with the data provided here and the commissions that an eventual broker can charge, we can have a good idea on how much to deduct.

v) Variance and Risk Measures

Histograms for the daily returns of our proposed solution alongside the daily returns of the three indexes are presented in Figure 16. The risk measures VaR and ModVaR for a 95% confidence level are also shown. On the histogram of proposed solution the VaR and ModVaR lines are almost superimposed. Remember that VaR calculations assume that the underlying return series distribution follows a normal distribution. The ModVaR adjusts for the actual skewness and kurtosis of the return series distribution, so, in this case the dashed lines very closed to each other means the return series follows a normal distribution that is, in this case, there are less outliers. The proposed solution ModVaR value is comparable to that of the NASDAQ 100 index but still, a little bit worse than the other two indexes.

vi) Case Study Results Summary

In terms of ROI our system using the conditions of this case study far outperforms the S&P 500, NASDAQ 100 and DJIA (Figure 14) with more than 30% ROI while the best of the others is 11%.

The drawdown figures also are not disappointing (Figure 15) regarding our system. Peak drawdown is marked at -2.72% which is much better than the corresponding values for other indexes.

This case performs 144 trades with 57.64% of them with positive return. These values make the implementation of this case study a promising investment strategy, particularly if we look at the investment ROI.

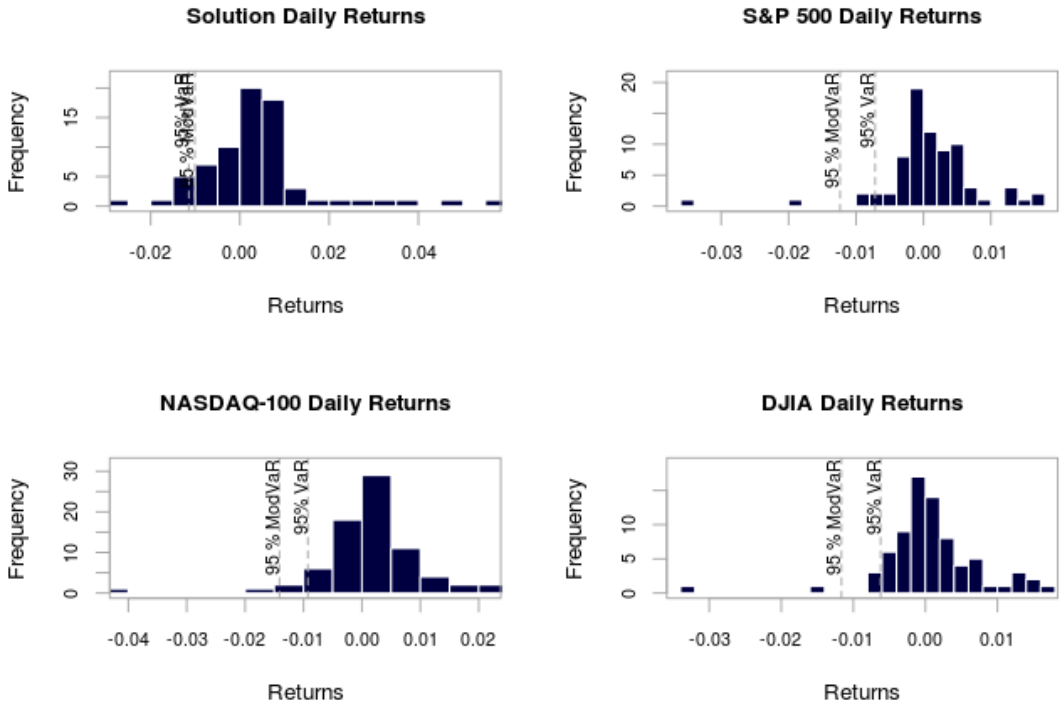

Figure 16 - Returns histogram for case study 1.

4.3.2. Case Study 2

i) Key Parameters

In this case study we used the same parameters as in case study 1. The difference here is that we apply the trading logic to our testing period between November 8 and December 16, 2016.

Summarizing, the trading rule is to buy 2 companies with the higher values of the variable 6 on the summary table provided that there are at least 20 companies with more than 11 tweets on their summary table with an SVM_PROB greater than 30%. Table 21 lists key simulation parameters.

ii) ROI

The ROI achieved using the parameters of this case study is presented in Figure 17 alongside the cumulative returns of the S&P 500, NASDAQ 100 and DJIA. In December 16 the ROI was 11.47%, 5.94%, 2.96% and 8.67% for the proposed solution, S&P 500, NASDAQ 100 and DJIA, respectively. From the figure we can see that the profit curve is a little bit swingy, but the equity curve is always superior to all the other indexes for the period.

Table 21 - Simulation parameters for case study 2.

Parameter	Value
Trading frequency	Daily
Holding Period (Days)	1
Minimum_Number_of_Tweets	11
Minimum_Number_of_Companies	20
Number_of_Companies_to_Buy	2
Variable_on_Summary_Table	6
Minimum_Probability	0.3

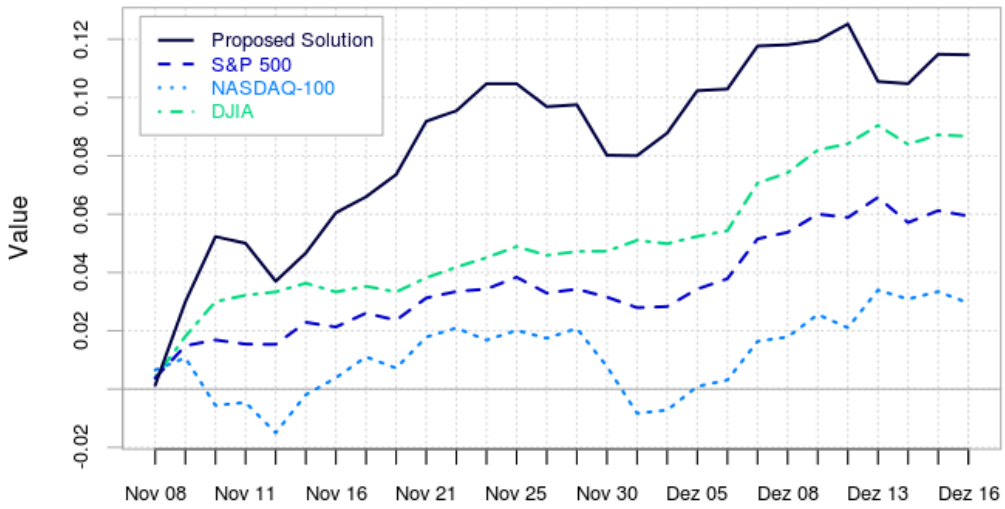

Figure 17 - Cumulative returns for case study 2.

iii) Drawdown

The drawdown of this case study and the drawdowns of S&P 500, NASDAQ 100 and DJIA are plotted in Figure 18. The peak drawdown for the proposed solution is -2.08%, which is better than the NASDAQ 100 drawdown at -2.91% but worse than the drawdown of the S&P 500 and DJIA at -0.98% and -0.55% respectively.

iv) Hit Ratio and Number of Trades

The trading strategy described in this case study entered the market in 27 days, each time buying two different stocks. This corresponds to 54 trades of which 35 had a positive return and 19 a negative return. The hit ratio is then 35/54 = 64.81% and of the 27 days 70.37% ended with a profit. The same

figure for the daily returns of the S&P 500, NASDAQ 100 and DJIA was 64.29%, 64.29% and 78.57%, respectively.

v) Variance and Risk Measures

Histograms for the daily returns of this case study alongside the daily returns of the three indexes are presented in Figure 19. The maximum daily amount we can expect to lose with this strategy is high. On any given day an investment implementing this strategy can lose triple that of an investment holding the S&P500 and DJIA indexes (according to the VaR value) but about the same as an investment holding the NASDAQ 100.

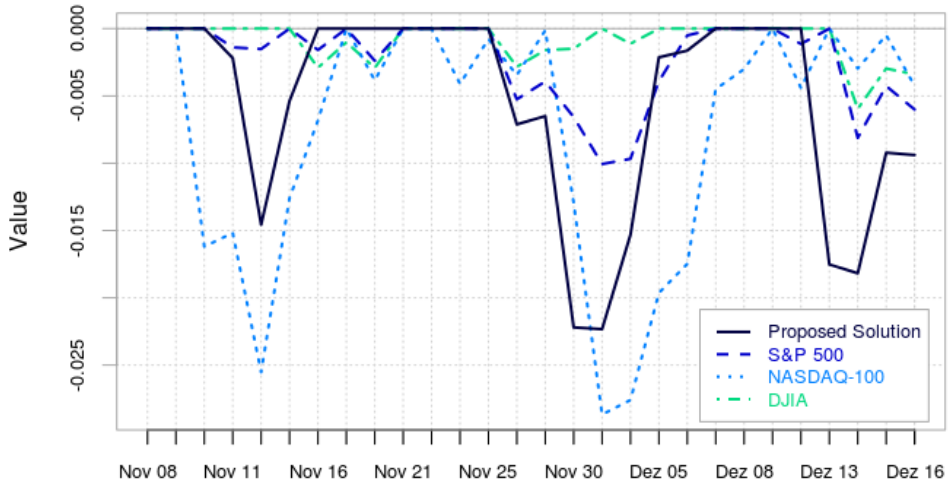

Figure 18 - Drawdown for case study 2.

vi) Case Study Results Summary

This case study presents a very interesting ROI outperforming all other indexes at almost 3%. This solution presents three downswings but they are small in size. Peak drawdown is about -2%.

Although the solution proposed by this case study was evaluated over a short period of time it presents interesting results particularly if we consider this is a follow up to case study 1.

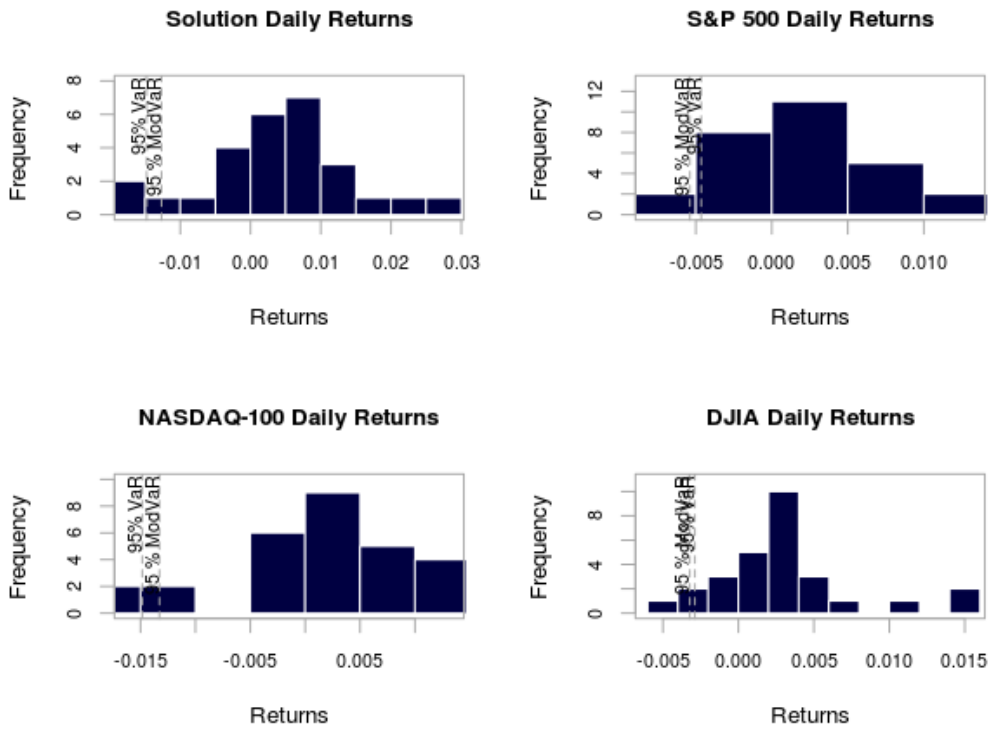

Figure 19 - Returns Histogram for Case Study 2.

4.3.3. Case Study 3

i) Key Parameters

This case study applies one of the best solutions, found for our training period, to our test period from November 8 and December 16, 2016.

The trading rule is to buy 1 company with the highest value of the variable 11 on the summary table provided that there are at least 15 companies with more than 4 tweets on their summary table with an SVM_PROB greater than 50%. Table 22 lists key simulation parameters.

ii) ROI

The ROI achieved using the parameters of this case study is presented in Figure 20 alongside the cumulative returns of the S&P 500, NASDAQ 100 and DJIA. In December 16 the ROI was 10.53%, 5.94%, 2.96% and 8.67% for the proposed solution, S&P 500, NASDAQ 100 and DJIA, respectively.

iii) Drawdown

The drawdown of this case study and the drawdowns of the S&P 500, NASDAQ 100 and DJIA are plotted in Figure 21. The peak drawdown for the proposed solution is -3.76%, which is better than the NASDAQ 100 drawdown at -2.91% but worse than the drawdown of the S&P 500 and DJIA at -0.98% and -0.55% respectively.

Table 22 - Simulation parameters for case study 3.

Parameter	Value
Trading frequency	Daily
Holding Period (Days)	1
Minimum_Number_of_Tweets	4
Minimum_Number_of_Companies	15
Number_of_Companies_to_Buy	1
Variable_on_Summary_Table	11
Minimum_Probability	0.5

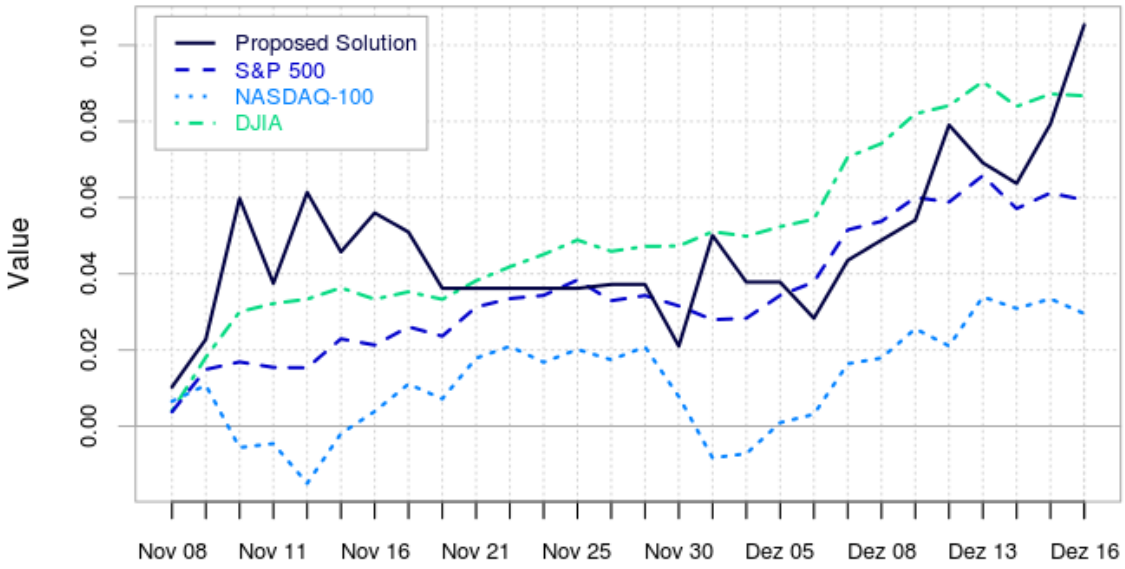

Figure 20 - Cumulative returns for case study 3.

iv) Hit Ratio and Number of Trades

The trading strategy described in this case study entered the market in 22 days, each time buying one company stock. This corresponds to 22 trades of which 13 had a positive return and 9 a negative return. The hit ratio is then 13/22 = 59.09% which is also the percentage of days ending with a profit.

v) Variance and Risk Measures

Histograms for the daily returns of this case study alongside the daily returns of the three indexes are presented in Figure 22. For this strategy the daily value at risk is high when compared to the daily returns of the S&P500, DJIA and NASDAQ 100. Using the modified value at risk with a 95% confidence level we conclude that, on any given day, we risk losing 3 times more following the strategy proposed by this case study than we risk losing when holding the S&P500 index. This is not

surprising if we recall that we only hold one company's stock at a time, so it is natural that this strategy falls victim to bigger swings.

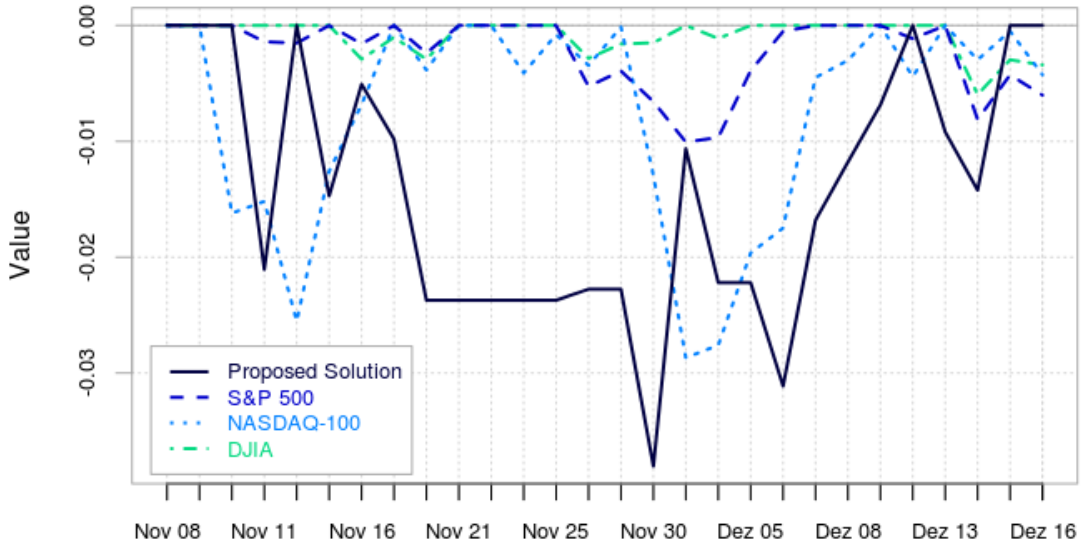

Figure 21 - Drawdown for case study 3.

vi) Case Study Results Summary

This case study ends our testing period outperforming the best performing index by almost 2%. Despite this, the strategy has a few less positive points. One is the peak drawdown which is about four times higher that the corresponding value for the DJIA and the second is the maximum amount we can expect to lose each day. These two values are not very appealing but are normal since, at any given day, we hold a maximum of one stock.

This case study applies the best performing strategy of our training period to our testing period. Despite having a few drawbacks the strategy still outperforms the market.

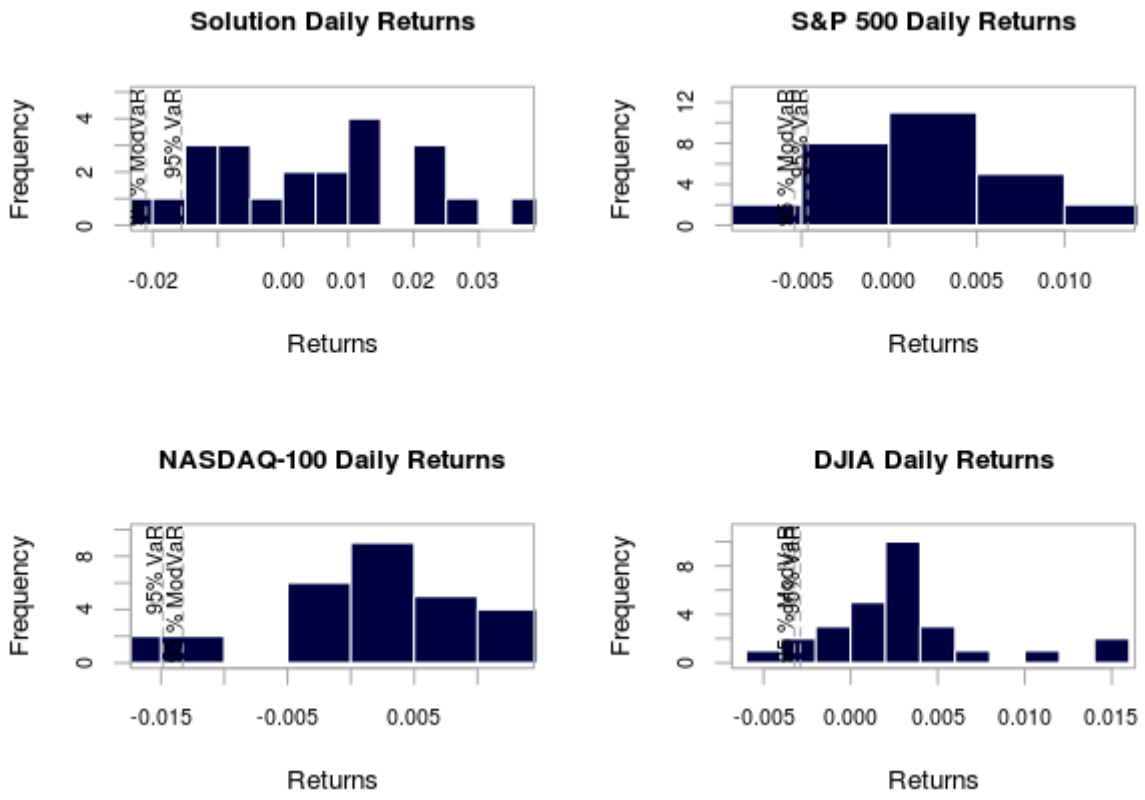

Figure 22 - Returns Histogram for Case Study 3.

4.4. Chapter Conclusions

In this chapter we intended to present the results of a practical implementation of our architecture proposed in Chapter 3. Of course, along the way, there were many parameter choices that influence the final profits and that is what we see in the three case studies presented.

Case studies 1 and 2 have the same parameters but they were applied to different time periods. Case study 1 incurs during our training period while case studies 2 and 3 refer to our testing period, that is, they were applied to unseen data. All case studies present interesting results and indicate that they could be sound guidelines for a future investment strategy.

The two main differences between case studies 2 and 3 are the different sentiment measure and the probability threshold for tweets exclusion, as given by the classifier. This has a profound impact since it greatly reduces the number of tweets the solution uses to make a prediction. Also, the number of selected daily companies is 1 in case study 3 instead of the 2 in case study 2. This increases risk and peak drawdown but also can swell returns.

Chapter 5. Conclusion and Future Work

In this work we implemented a system that predicts stock market fluctuations using public mood collected from Twitter. More precisely, we tried to use tweets to predict the future price movements of a selection of 504 stocks. To achieve that, the first step we took was to build a sentiment model using a Support Vector machine algorithm. To build that sentiment model we gathered tweets using emotion words as search terms. For example, we searched Twitter for the term "#happy" to obtain tweets from the happy emotion. Similarly to the emotion happy, the same procedure was done for a total of 80 emotions in order to guarantee a good coverage of the human sentiment. The 80 emotion words were based on the ones present in the Circumplex Model of Affect [62] and their synonyms. In total we gathered around 3 million tweets using these 80 emotion words as search terms, although, there are big disparities between the numbers of tweets collected for each emotion. Using the search term "#amazing" we were able to collect more than 200 000 tweets while the search term "#dispirited" culminated in less than 10 tweets. The big gap between both numbers is due to the fact that the word "amazing" is much more common than the word "dispirited" on Twitter and not because our procedure was different for different emotions.

The second step we took was to collect tweets that mention at least one of the 504 selected stocks. To collect these tweets we used the tickers of those companies as search terms. In total we collected around 4 million tweets mentioning companies. Also, here we verified the huge gap between the number of tweets mentioning different companies. We gathered more than 200 000 tweets mentioning Apple Inc. and about 350 mentioning TECO Energy. The sentiment model we built on the first step was used here to provide us the sentiment present on tweets from each company. Using that sentiment we built a trading rule that was optimized by a genetic algorithm in order to achieve the maximum profit.

Along the way we applied many preprocessing techniques to the text of the tweets. Most of them are very common in sentiment analysis. We used a stemming algorithm, remove punctuation and numbers, replace Twitter usernames, remove Twitter links, removed and stop words. We included emoticons in our feature set and question marks and exclamation points also. Additionally, we designed some rules that conducted to better results. Among these rules are:

- limiting of the number of hashtags per tweet;
- exclusion of tweets posted by users who submit a very high number of messages;
- exclusion of tweets submitted by users who post very similar messages;
- use the number of retweets and favorites to select the tweets to build the sentiment model;
- exclusion of tweets that don't have a predominant sentiment class as given by the sentiment model, or in other words, utilize only tweets with polarized sentiment.

We presented three case studies that demonstrate the results of the implementation of our trading rule on the stock market. Case study 1 presents the results obtained during our training period while case study 2 and 3 show strategies learned during our training period but applied to our testing period.

All case studies provide interesting results as evaluated along different metrics such as return on investment, drawdown and risk. Based on the results we obtained on our out of sample data we conclude that it is possible to create a profitable trading strategy using Twitter.

Future work could be devoted to adding more parameters to the GA optimization and calibrate it better in order to achieve better performances.

REFERENCES

[1] Bollen, J., Mao, H., & Zeng, X. (2011). Twitter mood predicts the stock market. Journal of Computational Science, 2(1), 1-8.

[2] Mittal, A., & Goel, A. (2012). Stock prediction using twitter sentiment analysis.Standford University, CS229 (2011 http://cs229.stanford.edu/proj2011/GoelMittal-StockMarketPredictionUsingTwitterSentimentAnalysis.pdf).

[3] Xu, F., & Keelj, V. (2014, July). Collective Sentiment Mining of Microblogs in 24-Hour Stock Price Movement Prediction. In 2014 IEEE 16th Conference on Business Informatics (Vol. 2, pp. 60-67). IEEE.

[4] Si, J., Mukherjee, A., Liu, B., Li, Q., Li, H., & Deng, X. (2013). Exploiting Topic based Twitter Sentiment for Stock Prediction. ACL (2), 2013, 24-29.

[5] Porshnev, A., Redkin, I., & Shevchenko, A. (2013, December). Machine learning in prediction of stock market indicators based on historical data and data from twitter sentiment analysis. In 2013 IEEE 13th International Conference on Data Mining Workshops (pp. 440-444). IEEE.

[6] Lo, A. W. (2007). Efficient markets hypothesis.

[7] Fama, E. F. (1995). Random walks in stock market prices. Financial analysts journal, 51(1), 75-80.

[8] Morstatter, F., Pfeffer, J., Liu, H., & Carley, K. M. (2013). Is the sample good enough? comparing data from twitter's streaming api with twitter's firehose.arXiv preprint arXiv:1306.5204.

[9] Baker, M., & Wurgler, J. (2007). Investor sentiment in the stock market. The Journal of Economic Perspectives, 21(2), 129-151.

[10] Liu, B. (2012). Sentiment analysis and opinion mining. Synthesis lectures on human language technologies, 5(1), 1-167.

[11] Buettner, R., & Buettner, K. (2016, January). A Systematic Literature Review of Twitter Research from a Socio-Political Revolution Perspective. In 2016 49th Hawaii International Conference on System Sciences (HICSS) (pp. 2206-2215). IEEE.

[12] Shawe-Taylor, J., & Cristianini, N. (2004). Kernel methods for pattern analysis. Cambridge university press.

[13] Berry, M. W. (2004). Survey of text mining. Computing Reviews, 45(9), 548.

[14] Porter, M. F. (1980). An algorithm for suffix stripping. Program,14(3), 130-137.

[15] Vapnik, V. (2013). The nature of statistical learning theory. Springer Science & Business Media.

[16] Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine learning,20(3), 273-297.

[17] Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992, July). A training algorithm for optimal margin classifiers. In Proceedings of the fifth annual workshop on Computational learning theory (pp. 144-152). ACM.

[18] Cristianini, N., & Shawe-Taylor, J. (2000). An introduction to support vector machines and other kernel-based learning methods. Cambridge university press.

[19] Schölkopf, B., & Smola, A. J. (2002). Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT press.

[20] Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. Data mining and knowledge discovery, 2(2), 121-167.

[21] Lee, Y. J., Mangasarian, O. L., & Wolberg, W. H. (2003). Survival-time classification of breast cancer patients. Computational Optimization and Applications, 25(1-3), 151-166.

[22] Huerta, R., Corbacho, F., & Elkan, C. (2013). Nonlinear support vector machines can systematically identify stocks with high and low future returns.Algorithmic Finance, 2(1), 45-58.

[23] Hu, Z., Zhu, J., & Tse, K. (2013, November). Stocks market prediction using support vector machine. In 2013 6th International Conference on Information Management, Innovation Management and Industrial Engineering (Vol. 2, pp. 115-118). IEEE.

[24] Heisele, B., Ho, P., & Poggio, T. (2001). Face recognition with support vector machines: Global versus component-based approach. In Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on(Vol. 2, pp. 688-694). IEEE.

[25] Matic, N., Guyon, I., Denker, J., & Vapnik, V. (1993, October). Writer-adaptation for on-line handwritten character recognition. In Document Analysis and Recognition, 1993., Proceedings of the Second International Conference on (pp. 187-191). IEEE.

[26] Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. Annals of eugenics, 7(2), 179-188.

[27] Hofmann, M. (2006). Support Vector Machines—Kernels and the Kernel Trick. Notes, 26.

[28] Schiilkopf, B. (2001). The kernel trick for distances. Advances in neural information processing systems, 13, 301-307.

[29] Holland, J. H. (1975). Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. U Michigan Press.

[30] Zhang, K., Cheng, Y., Liao, W. K., & Choudhary, A. (2011, August). Mining millions of reviews: a technique to rank products based on importance of reviews. In Proceedings of the 13th International Conference on Electronic Commerce (p. 12). ACM.

[31] McGlohon, M., Glance, N. S., & Reiter, Z. (2010, May). Star Quality: Aggregating Reviews to Rank Products and Merchants. In ICWSM.

[32] Asur, S., & Huberman, B. A. (2010, August). Predicting the future with social media. In Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on (Vol. 1, pp. 492-499). IEEE.

[33] Joshi, M., Das, D., Gimpel, K., & Smith, N. A. (2010, June). Movie reviews and revenues: An experiment in text regression. In Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (pp. 293-296). Association for Computational Linguistics.

[34] Sadikov, E., Parameswaran, A., & Venetis, P. (2009). Blogs as predictors of movie success.

[35] Liu, Y., Huang, X., An, A., & Yu, X. (2007, July). ARSA: a sentiment-aware model for predicting sales performance using blogs. In Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval (pp. 607-614). ACM.

[36] Neethu, M. S., & Rajasree, R. (2013, July). Sentiment analysis in twitter using machine learning techniques. In Computing, Communications and Networking Technologies (ICCCNT), 2013 Fourth International Conference on(pp. 1-5). IEEE.

[37] Gautam, G., & Yadav, D. (2014, August). Sentiment analysis of twitter data using machine learning approaches and semantic analysis. In Contemporary Computing (IC3), 2014 Seventh International Conference on (pp. 437-442). IEEE.

[38] Hong, Y., & Skiena, S. (2010, May). The Wisdom of Bookies? Sentiment Analysis Versus. the NFL Point Spread. In ICWSM.

[39] Tumasjan, A., Sprenger, T. O., Sandner, P. G., & Welpe, I. M. (2010). Predicting Elections with Twitter: What 140 Characters Reveal about Political Sentiment. ICWSM, 10, 178-185.

[40] Gayo Avello, D., Metaxas, P. T., & Mustafaraj, E. (2011). Limits of electoral predictions using twitter. In Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media. Association for the Advancement of Artificial Intelligence.

[41] Burnap, P., Gibson, R., Sloan, L., Southern, R., & Williams, M. (2016). 140 characters to victory?: Using Twitter to predict the UK 2015 General Election.Electoral Studies, 41, 230-233.

[42] Wang, H., Can, D., Kazemzadeh, A., Bar, F., & Narayanan, S. (2012, July). A system for real-time twitter sentiment analysis of 2012 us presidential election cycle. In Proceedings of the ACL 2012 System Demonstrations (pp. 115-120). Association for Computational Linguistics.

[43] Yano, T., & Smith, N. A. (2010, May). What's Worthy of Comment? Content and Comment Volume in Political Blogs. In ICWSM.

[44] Yano, T., Yogatama, D., & Smith, N. A. (2013). A penny for your tweets: Campaign contributions

and capitol hill microblogs.

[45] Vaziripour, E., Giraud-Carrier, C., & Zappala, D. (2016, March). Analyzing the Political Sentiment of Tweets in Farsi. In Tenth International AAAI Conference on Web and Social Media.

[46] De Choudhury, M., Counts, S., & Gamon, M. (2012, May). Not All Moods are Created Equal! Exploring Human Emotional States in Social Media. InICWSM.

[47] Groh, G., & Hauffa, J. (2011, July). Characterizing Social Relations Via NLP-Based Sentiment Analysis. In ICWSM.

[48] Cha, M., Haddadi, H., Benevenuto, F., & Gummadi, P. K. (2010). Measuring User Influence in Twitter: The Million Follower Fallacy. ICWSM, 10(10-17), 30.

[49] Bollen, J., Mao, H., & Pepe, A. (2011). Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena. ICWSM, 11, 450-453.

[50] O'Connor, B., Balasubramanyan, R., Routledge, B. R., & Smith, N. A. (2010). From tweets to polls: Linking text sentiment to public opinion time series.ICWSM, 11(122-129), 1-2.

[51] Park, M., Cha, C., & Cha, M. (2012, August). Depressive moods of users portrayed in Twitter. In Proceedings of the ACM SIGKDD Workshop on healthcare informatics (HI-KDD) (pp. 1-8).

[52] Paul, M. J., & Dredze, M. (2011). You are what you Tweet: Analyzing Twitter for public health. ICWSM, 20, 265-272.

[53] Mislove, A., Lehmann, S., Ahn, Y. Y., Onnela, J. P., & Rosenquist, J. N. (2011). Understanding the Demographics of Twitter Users. ICWSM, 11, 5th.

[54] Agarwal, A., Xie, B., Vovsha, I., Rambow, O., & Passonneau, R. (2011, June). Sentiment analysis of twitter data. In Proceedings of the workshop on languages in social media (pp. 30-38). Association for Computational Linguistics.

[55] Whissell, C. (1989). The dictionary of affect in language. Emotion: Theory, research, and experience, 4(113-131), 94.

[56] Moschitti, A. (2006, September). Efficient convolution kernels for dependency and constituent syntactic trees. In European Conference on Machine Learning (pp. 318-329). Springer Berlin Heidelberg.

[57] Haussler, D. (1999). Convolution kernels on discrete structures (Vol. 646). Technical report, Department of Computer Science, University of California at Santa Cruz.

[58] Barbosa, L., & Feng, J. (2010, August). Robust sentiment detection on twitter from biased and noisy data. In Proceedings of the 23rd International Conference on Computational Linguistics: Posters (pp. 36-44). Association for Computational Linguistics.

[59] Riloff, E., & Wiebe, J. (2003, July). Learning extraction patterns for subjective expressions. In Proceedings of the 2003 conference on Empirical methods in natural language processing (pp. 105-112). Association for Computational Linguistics.

[60] Witten, I. H., & Frank, E. (2005). Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann.

[61] Hasan, M., Rundensteiner, E., & Agu, E. (2014). Emotex: Detecting emotions in twitter messages.

[62] J. A. Russell, “A circumplex model of affect,” Journal of Personality and Social Psychology, vol. 39, pp. 1161– 1178, 1980.

[63] Davidov, D., Tsur, O., & Rappoport, A. (2010, August). Enhanced sentiment learning using twitter hashtags and smileys. In Proceedings of the 23rd international conference on computational linguistics: posters (pp. 241-249). Association for Computational Linguistics.

[64] Davidov, D., & Rappoport, A. (2008, June). Unsupervised Discovery of Generic Relationships Using Pattern Clusters and its Evaluation by Automatically Generated SAT Analogy Questions. In ACL (pp. 692-700).

[65] Go, A., Bhayani, R., & Huang, L. (2009). Twitter sentiment classification using distant supervision. CS224N Project Report, Stanford, 1, 12.

[66] Qasem, M., Thulasiram, R., & Thulasiram, P. (2015, August). Twitter sentiment classification

using machine learning techniques for stock markets. In Advances in Computing, Communications and Informatics (ICACCI), 2015 International Conference on (pp. 834-840). IEEE.

[67] Kanakaraj, M., & Guddeti, R. M. R. (2015, March). NLP based sentiment analysis on Twitter data using ensemble classifiers. In Signal Processing, Communication and Networking (ICSCN), 2015 3rd International Conference on (pp. 1-5). IEEE.

[68] Vu, T. T., Chang, S., Ha, Q. T., & Collier, N. (2012). An experiment in integrating sentiment features for tech stock prediction in twitter.

[69] Boudoukh, J., Feldman, R., Kogan, S., & Richardson, M. (2013). Which news moves stock prices? a textual analysis (No. w18725). National Bureau of Economic Research.

[70] Zhang, W., & Skiena, S. (2010, May). Trading Strategies to Exploit Blog and News Sentiment. In ICWSM.

[71] Baik, B., Cao, Q., Choi, S., & Kim, J. M. (2016). Local Twitter Activity and Stock Returns. Available at SSRN 2783670.

[72] Chung, S., & Liu, S. (2011). Predicting stock market fluctuations from twitter.Berkeley, California.

[73] Bing, L., Chan, K. C., & Ou, C. (2014, November). Public sentiment analysis in Twitter data for prediction of a company's stock price movements. In e-Business Engineering (ICEBE), 2014 IEEE 11th International Conference on(pp. 232-239). IEEE.

[74] Zhang, L. (2013). Sentiment analysis on Twitter with stock price and significant keyword correlation (Doctoral dissertation).

[75] Liew, J. K. S., & Wang, G. Z. (2015). Twitter Sentiment and IPO Performance: A Cross-Sectional Examination. Available at SSRN 2567295.

[76] Hu, Z., Jiao, J., & Zhu, J. (2014). Using Tweets to Predict the Stock Market.

[77] Feldman, R., Rosenfeld, B., Bar-Haim, R., & Fresko, M. (2011, August). The stock sonar—sentiment analysis of stocks based on a hybrid approach. InTwenty-Third IAAI Conference.

[78] Rosenfeld, B., Feldman, R., & Ungar, L. (2008, October). Using sequence classification for filtering web pages. In Proceedings of the 17th ACM conference on Information and knowledge management (pp. 1355-1356). ACM.

[79] Bouktif, S., & Awad, M. A. (2013, August). Ant colony based approach to predict stock market movement from mood collected on Twitter. In Advances in Social Networks Analysis and Mining (ASONAM), 2013 IEEE/ACM International Conference on (pp. 837-845). IEEE.

[80] Zhang, X., Fuehres, H., & Gloor, P. A. (2011). Predicting stock market indicators through twitter “I hope it is not as bad as I fear”. Procedia-Social and Behavioral Sciences, 26, 55-62.

[81] Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation.Journal of machine Learning research, 3(Jan), 993-1022.

[82] Teh, Y. W., Jordan, M. I., Beal, M. J., & Blei, D. M. (2012). Hierarchical dirichlet processes. Journal of the american statistical association.

[83] Ranco, G., Aleksovski, D., Caldarelli, G., Grčar, M., & Mozetič, I. (2015). The effects of Twitter sentiment on stock price returns. PloS one, 10(9), e0138441.

[84] Chua, C. C., Milosavljevic, M., & Curran, J. R. (2009, December). A sentiment detection engine for internet stock message boards. In Proceedings of the Australasian language technology association workshop (pp. 89-93).

[85] Rennie, J. D., Shih, L., Teevan, J., & Karger, D. R. (2003, August). Tackling the poor assumptions of naive bayes text classifiers. In ICML (Vol. 3, pp. 616-623).

[86] Yang, Y., & Pedersen, J. O. (1997, July). A comparative study on feature selection in text categorization. In ICML (Vol. 97, pp. 412-420).

[87] Wang, X., Wei, F., Liu, X., Zhou, M., & Zhang, M. (2011, October). Topic sentiment analysis in twitter: a graph-based hashtag sentiment classification approach. In Proceedings of the 20th ACM international conference on Information and knowledge management (pp. 1031-1040). ACM.

[88] Posner, J., Russell, J. A., & Peterson, B. S. (2005). The circumplex model of affect: An integrative

approach to affective neuroscience, cognitive development, and psychopathology. Development and psychopathology, 17(03), 715-734.

[89] Jeff Gentry (2015). twitteR: R Based Twitter Client. R package version 1.1.9. https://CRAN.R-project.org/package=twitteR.

[90] Batuwita, R., & Palade, V. (2013). Class imbalance learning methods for support vector machines. Imbalanced learning: Foundations, algorithms, and applications, 83-99.

[91] Jeni, L. A., Cohn, J. F., & De La Torre, F. (2013, September). Facing Imbalanced Data--Recommendations for the Use of Performance Metrics. InAffective Computing and Intelligent Interaction (ACII), 2013 Humaine Association Conference on (pp. 245-251). IEEE.

[92] Timothy P. Jurka, Loren Collingwood, Amber E. Boydstun, Emiliano Grossman and Wouter van Atteveldt (2014). RTextTools: Automatic Text Classification via Supervised Learning. R package version 1.4.2. https://CRAN.R-project.org/package=RTextTools.

[93] Kurt Hornik, David Meyer and Christian Buchta (2016). slam: Sparse Lightweight Arrays and Matrices. R package version 0.1-35. https://CRAN.R-project.org/package=slam.

[94] Simon Urbanek (2016). rJava: Low-Level R to Java Interface. R package version 0.9-8. https://CRAN.R-project.org/package=rJava.

[95] Schütze, H. (2008, July). Introduction to Information Retrieval. In Proceedings of the international communication of association for computing machinery conference.

[96] Ingo Feinerer and Kurt Hornik (2015). tm: Text Mining Package. R package version 0.6-2. https://CRAN.R-project.org/package=tm.

[97] David Meyer, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel and Friedrich Leisch (2015). e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien. R package version 1.6-7. https://CRAN.Rproject.org/package=e1071.

[98] Walter R. Mebane, Jr., Jasjeet S. Sekhon (2011). Genetic Optimization Using Derivatives: The rgenoud Package for R. Journal of Statistical Software, 42(11), 1-26. URL http://www.jstatsoft.org/v42/i11/.

[99] Brian G. Peterson and Peter Carl (2014). PerformanceAnalytics: Econometric tools for performance and risk analysis. R package version 1.4.3541. https://CRAN.R-project.org/package=PerformanceAnalytics.