INTERNATIONAL
HELLENIC
UNIVERSITY

Predicting Stocks Movement using Social Media Analytics

Touparis Fotios

SID: 3301150007

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Information and Communication Systems

I would like to thank some people who helped me with their own way for this project.

First of all, I would like to thank my supervisor Dr. Tjortjis for the excellent mentoring and the essential help he gave during this project.

Secondly, my parents, who helped me in my academic way, support me in every aspect and I believe they did (and do) the best for me.

Finally, I would like to thank John and Christina for their support during my studies in IHU.

Table of Contents

1. Introduction

In the first chapter, I will introduce the challenge of the project. First, I'll explain the basics of Stock Market forecasting and then why Social Media Analytics is the proper tool to forecast its movement. Finally, I will point out the advantages of Twitter and its API and why these are the best option for such analysis.

1.1 The problem

The stock market is one of the greatest examples of the chaotic model. The main precept behind this theory is the underlying notion of small occurrences significantly affecting the outcomes of seemingly unrelated events. And in a stock market we have a lot of parameters where the traders take care of and make their moves, which moves are parameters again in the chaos of the stock market. So, it was and of course it is impossible to predict such moves.

There are of course techniques to predict with good accuracy and in special conditions the movement of the stock price, which is actually the behaviour of the trader, as explained below. All the analysts are trying to understand the future behaviour of the traders. What actually is critical to find is the trader's opinion regarding a stock, because his thoughts for a company, let's say positive or negative for simplicity, will determine his moves for this stock.

Historical analysis or stocks comparison can predict the movement of a stock using processed historical data and can be very accurate only when all the other parameters stay as before. But in a chaotic environment like the stock market this is almost impossible to happen. What really is the movement of a stock, is the intension of the trader. And every trader has different behaviour even with the same parameters.

So, one way to predict the movement is to understand the intension of the traders regardless the traditional methods. Nowadays, there are a lot of traders joining the stock market just because they think that a stock will increase its value because they believe that the company goes well and makes profit. As long as there are such 'simple' traders, the traditional methods of forecasting are outdated. And in order to increase the forecast accuracy, the understanding of the traders intension is very important.

That's exactly what I will try to do with this project. Using social Media analysis, I 'll try to extract the sentiment that Twitter users have for specific companies and use it to forecast the stock movement. So, I will try to forecast the movement using the most unstable and chaotic parameter but in the same time the most accurate than the traditional methods. Of course, it doesn't mean that when a lot of users 'like' a company its stock will raise. There is a lot of methodology explained in chapters 2 and 4.

1.2 Stock market forecasting

The history of forecasting stocks using any method starts a lot of years ago. The first articles published on 1933[1] and until now many researchers tried to find out the best way to predict the movement of a stock. But, the stock market model is so chaotic which makes it almost impossible to find a formula to predict correctly or with high accuracy the movement of it.

It is time to explain as simple as it can be how a stock market 'operates'. For simplicity reasons, we assume that only simple users can buy or sell stocks. Some companies make available for the users some of their stocks. Let's say that user1 owns ten stocks of company1 with a nominal amount of 100 euros each. That means that he owns an amount of 1000 euro in stocks. Another user wants

to buy some company's1 stocks and the only way is to find out someone who sells. If anyone wants to sell a stock then user2 cannot buy. So, one way for user2 to deal with it is to offer more money than the nominal value of the stock in order to allure user1 and make him sell his stocks. On the other hand, if user1 wants to sell his stocks, someone has to buy them. So, in order to attract buyers, he offers the stocks in less value than the nominal. The price of the transaction is actually the stock price.

For every stock transaction, there must be a buyer and a seller. When you buy 100 shares of stock someone else must sell it to you. Either buyers or sellers can be more aggressive than the other, pushing the price up or down.

When the price of a stock goes down, sellers are more aggressive because they want to sell at a lower and lower price. The buyers are also timid and only intension to buy at lower at lower prices. The price will continue to fall until the price reaches a point where buyers step in and become more aggressive and intension to buy at higher prices, pushing the price back up.

Summurizing all thee above, the stock price is formed from the intension and what the users believe about the companies. So, it is more important what the traders believe about a company than what actually is the company's situation. This is the reason of the complexity of each stockmarket and of course the difficulty of forecasting considering the intension of the traders.

1.3 Forecasting with Sentiment Analysis

As described above, in a very general way, the intension of the trader is the key of stock movement and so the best way to forecast its movement. It is really hard to extract the trader's intension in a way that can give you usable results.

The sentiment analysis is the process of determining whether a piece of writing is positive, negative or neutral and there are cases that more 'sentiments' can be extracted, such as happy, unhappy, angry etc. Its also known as opinion mining, deriving the opinion or attitude of a speaker. A common use case for this technology is to discover how people feel about a particular topic like the stock market.

In this project I'll try to perform a sentiment analysis for twitter users and extract their sentiments according to their tweets. This is somehow the way to find out the public opinion, which the traders are consulting for their decisions.

1.4 Why Twitter

Twitter is one of the most known social media network, with over 400 million active users. The main difference than other social media networks such as Facebook, Linkedin, Instagram, Flickr or Google+, is that Twitter has the most textual context than the others and that makes it the best choice to perform analysis based on text like sentiment analysis. There are cases where images can be also analysed, but the text is much easier and more accurate than the other. Also, tweets are homogeneous with similar characteristics and this can eliminate outliers in the further analysis.

Microblogging today has become a very popular communication tool among Internet users. Millions of users share opinions on different aspects of life everyday. Therefore microblogging web-sites are rich sources of data for opinion mining and sentiment analysis. Because microblogging has appeared relatively recently, there are a few research works that were devoted to this topic.

In this project, I focus on using Twitter, the most popular microblogging platform, for the task of sentiment analysis. I'll show how to automatically collect a large amount of tweets for sentiment analysis. I perform linguistic analysis of the collected tweets and I'll try to match the results with the stockmarket movement. Using the downloaded tweets, I build a sentiment classifier (based on Naive Bayes), that is able to determine positive, negative and neutral sentiments for a tweet.

In this project, I worked with English language, however, the proposed technique can be used with any other language, as far as the algorithms libraries have the corresponding language.

Moreover, the tweets are texts up to 140 characters and make them more comprehensive texts because of this limit. But, one of the most important reasons of Twitters advantage in gathering data is its API. It is really easier for a junior developer to start gathering data from Twitter than other platforms. The main API that is used is the Search API. With that you can query through the API and download tweets with specific criteria, such as specific hashtags, geolocations, language ect in the suitable form for each problem.

2. Literarture Review

In this chapter we will analyse the similar work and similar ways which have done in the past in order to solve similar problems.

The past few years, the international financial world has suffered two serious stock exchange collapses. Surprisingly however, these crashes did not have the devastating effects on the international economy which could have been expected from a historical point of view. A certain separation of the stock exchange from the economic situation seems to have emerged, the stock exchange no longer being the direct barometer of economic development in some areas.

This partial disappearance of the narrow cause and time relationship between the stock exchange and the actual economic fluctuations makes the prediction of stock prices even more difficult than it had been until now. In addition to this, stock trading has become able and forced to react to political and other relevant events faster and faster due to the ever increasing use of data processing technology and to the improvement of communication systems.

Banks, financial institutions, large-scale investors and stock brokers therefore now more than ever face the problem of having to buy and resell at a profit a maximum number of stocks within the shortest possible time. In this regard, a time span of only a few hours between buying and selling stocks is not unusual.

Under these conditions, what are the possibilities for predicting stock prices with the help of data processing and computer programs?

Conventional (i.e., non-adaptive) prediction programs are working best in situations where the real course of stock prices can be smoothened for the prediction by means of statistical procedures without loss of prediction relevance. In the case of medium-term investment, this is realistic and useful,

for instance, for private investors who usually keep their stocks for a number of weeks or months. Large-scale investors and brokers can use such procedures on a limited scale only. They make their profits, for the most part, not from a maximum price increase of fewer stocks over a longer period of time, as is the case for private in vestors, but rather as a rule from a relatively smal 1 price increase of a great number of stocks in an extremely short time.

Under these conditions, conventional prediction methods can only be used with a great amount of effort. The price fluctuations are for the most part too minimal to be successfully recognized and predicted with the statistical procedures which smoothened the course of stock prices for prediction purposes. Statistical smoothing usually disregards just that factor which is most important for predictions in short periods of time: small upward or downward price fluctuations.

A further and very important disadvantage of conventional nonadaptive program techniques is that for the analysis of the data, all relevant influencing factors must already be known in advance.

Conventional programs must "know" all parameters and influencing factors, i.e., they must contain the parameters along with their assessments and weightedness a priori in their program code. The designer of this type of program must therefore know and determine exactly which influencing factors he thinks are important and to what extent the wants to consider them.

In the case of the economic development and intertwinement of today's world, such as we are witnessing with the upheavals in Eastern Europe, it is practically impossible to recognize all the relevant influencing factors and explicitly formulate them. The fundamental analysis is basically restricted by the incredible abundance and complexity of information to be considered. No one can determine today what relevant factors will affect the world economy in the next few years. Too many factors are unsure.

As a result, it is highly improbable that a conventional program should be able to correctly recognize and predict, even with only approximate accuracy, the

trends implicitly contained in the economic data of the future. This would only be possible if the program were constantly adapted to the current situation, which is as a rule very costly and difficult, and which would probably take more time than the life-span of the trends themselves.

3. Data Gathering

3.1 Introduction

Social Media platforms today has become a very popular communication tool among Internet users. Millions of messages are appearing daily in popular web-sites that provide services for Social Media such as Twitter, Tumblr, Facebook. Authors of those messages write about their life, share opinions on variety of topics and discuss current issues. Because of a free format of messages and an easy accessibility of Social Media platforms, Internet users tend to shift from traditional communication tools (such as traditional blogs or mailing lists) to Social Media services. As more and more users post about products and services they use, or express their political and religious views, Social Media web-sites become valuable sources of people's opinions and sentiments. Such data can be efficiently used for marketing or social studies.

3.2 The Twitter case

I use a dataset formed of collected tweets from Twitter. Twitter contains a very large number of very short texts created by the users of this Social Media platform. The contents of the messages vary from personal thoughts to public statements. As the audience of Social Media platforms and services grows every day, data from these sources can be used in opinion mining and sentiment analysis tasks. For example, companies may be interested in the following questions:

- *What do people think about their products (services, company etc.)?*
- *How positive (or negative) are people about the company?*
- *What would people prefer their products to be like?*

Political parties may be interested to know if people support their program or not. Social organizations may ask people's opinion on current debates. All this information can be obtained from Social Media services, as their users post everyday what they like/dislike, and their opinions on many aspects of their life.

In this project, I study how Social Media can be used for sentiment analysis purposes. I'll show how to use Twitter as a corpus for sentiment analysis and opinion mining. I use Social Media and more particularly Twitter for the following reasons:

- Social Media platforms are used by different people to express their opinion about different topics, thus it is a valuable source of people's opinions.

- Twitter contains an enormous number of text posts and it grows every day. The collected corpus can be arbitrarily large.

- Twitter's audience varies from regular users to celebrities, company representatives, politicians. Therefore, it is possible to collect text posts of users from different social and interests' groups.

- Twitter's audience is represented by users from many countries.

Although users from U.S. are prevailing, it is possible to collect data in different languages. I collected about 300000 text posts from Twitter evenly split automatically between three sets of texts:

1. texts containing positive emotions, such as happiness, amusement or joy
2. texts containing negative emotions, such as sadness, anger or disappointment
3. objective texts that only state a fact or do not express any emotions

I will perform a sentiment analysis of the downloaded tweets and show how to build a sentiment classifier that uses ready libraries for training and the actual data from prediction.

Figure 1: Twitter's profile

3.3 The Search API

The Twitter Search API is part of Twitter's REST API. It allows queries against the indices of recent or popular Tweets and behaves similarly to, but not exactly like the Search feature available in Twitter mobile or web clients, such as Twitter.com search. The Twitter Search API searches against a sampling of recent Tweets published in the past 7 days.

3.4 The TweePy python library

Python is great language for all sorts of things. Very active developer community creates many libraries which extend the language and make it easier to use various services. One of those libraries is tweepy. Tweepy is open-sourced, hosted on GitHub and enables Python to communicate with Twitter platform and use its API.

At the time of writing, the current version of tweepy is 3.50. It was released on August, and offers various bug fixes and new functionality compared to the previous version. The 4.x version is being developed but it is currently unstable so a huge majority of the users should use the regular version. Installing tweepy is easy, it can be cloned from the Github repository.

Tweepy supports accessing Twitter via Basic Authentication and the newer method, OAuth. Twitter has stopped accepting Basic Authentication so OAuth is now the only way to use the Twitter API.

Here is a sample code of how to access the Twitter API using tweepy with OAuth:

```python
import tweepy

# Consumer keys and access tokens, used for OAuth
consumer_key = '7EyzTcAh[redacted]'
consumer_secret = 'a44Ri[redacted]Y4l'
access_token = 'z00Xy9Ah[redacted]'
access_token_secret = 'A[redacted]NMqMW6p'

# OAuth process, using the keys and tokens
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)

# Creation of the actual interface, using authentication
api = tweepy.API(auth)

# Sample method, used to update a status
api.update_status('Hello Python Central!')
```

Figure 2: Twitter API using tweepy with OAuth

For security reasons the consumer keys and access tokens are invisible. In order to get such credentials, the developer can login to Twitter and register to Search API for free.

The main difference between Basic and OAuth authentication are the consumer and access keys. With Basic Authentication, it was possible to provide a username and password and access the API, but since 2010 when the Twitter started requiring OAuth, the process is a bit more complicated. An app has to be created at dev.twitter.com.

OAuth is a bit more complicated initially than Basic Auth, since it requires more effort, but the benefits it offers are very lucrative:

- Tweets can be customized to have a string which identifies the app which was used.

- It doesn't reveal user password, making it more secure.

- It's easier to manage the permissions, for example a set of tokens and keys can be generated that only allows reading from the timelines, so in case someone obtains those credentials, he/she won't be able to write or send direct messages, minimizing the risk.

The application doesn't reply on a password, so even if the user changes it, the application will still work.

After logging in to the portal, and going to "Applications", a new application can be created which will provide the needed data for communicating with Twitter API.

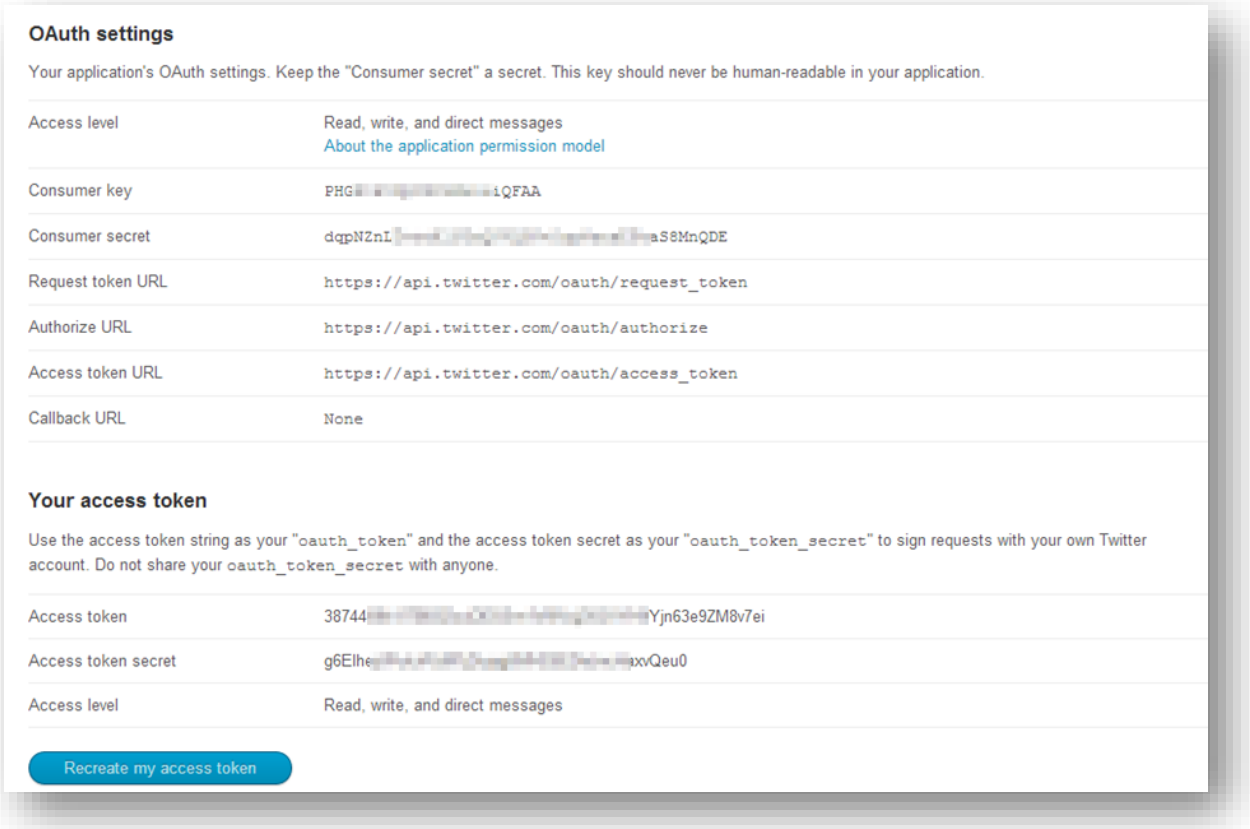


Figure 3: OAuth settings and access token

This is a screen which has all of the data needed to talk to Twitter network. It is important to note that by default, the app has no access to direct messages,

so by going to the settings and changing the appropriate option to "Read, write and direct messages", you can enable your app to have access to every Twitter feature.

Tweepy provides access to the well documented Twitter API. With tweepy, it's possible to get any object and use any method that the official Twitter API offers. For example, a User object has its documentation at https://dev.twitter.com/docs/platform-objects/users and following those guidelines, tweepy can get the appropriate information.

Main Model classes in the Twitter API are Tweets, Users, Entities and Places. Access to each returns a JSON-formatted response and traversing through information is very easy in Python.

```
api = tweepy.API(auth)

with open('tweet3.txt', 'w') as outfile:
    for status in tweepy.Cursor(api.search, q='#Apple',lang='en').items(100000):
        data1 = str(status.id) +";"+ str(status.created_at) +";"+  str(status.retweet_count) +";"+  status.text

        json.dump(data1, outfile)
```

Figure 4: Python sample code of requesting specific info

In Figure 3 we can see that data1 is filled by id, created_at, retweet_count and text. These are the data used for the sentiment analysis as well. The whole json file contains a lot of data which is not important for this project.


```
[{
  "created_at": "Thu Jun 22 21:00:00 +0000 2017",
  "id": 877994604561387500,
  "id_str": "877994604561387520",
  "text": "Creating a Grocery List Manager Using Angular, Part 1: Add &amp;amp; Display Items
https://t.co/xFox78juL1 #Angular",
```

```
"truncated": false,
"entities": {
  "hashtags": [{
    "text": "Angular",
    "indices": [103, 111]
  }],
  "symbols": [],
  "user_mentions": [],
  "urls": [{
    "url": "https://t.co/xFox78juL1",
    "expanded_url": "http://buff.ly/2sr60pf",
    "display_url": "buff.ly/2sr60pf",
    "indices": [79, 102]
  }]
},
"source": "<a href=\"http://bufferapp.com\" rel=\"nofollow\">Buffer</a>",
"user": {
  "id": 772682964,
  "id_str": "772682964",
  "name": "SitePoint JavaScript",
  "screen_name": "SitePointJS",
  "location": "Melbourne, Australia",
  "description": "Keep up with JavaScript tutorials, tips, tricks and articles at SitePoint.",
  "url": "http://t.co/cCH13gqeUK",
  "entities": {
    "url": {
      "urls": [{
        "url": "http://t.co/cCH13gqeUK",
        "expanded_url": "http://sitepoint.com/javascript",
        "display_url": "sitepoint.com/javascript",
        "indices": [0, 22]
      }]
    },
    "description": {
```

```
        "urls": []
      }
    },
    "protected": false,
    "followers_count": 2145,
    "friends_count": 18,
    "listed_count": 328,
    "created_at": "Wed Aug 22 02:06:33 +0000 2012",
    "favourites_count": 57,
    "utc_offset": 43200,
    "time_zone": "Wellington",
  },
}]
```


Above, we can see a full json response from Twitter API[2], where all the information is available to use. From all the available, only the 4 where the most important. The elements which used are:

- ID: The ID is the unique identifier of each tweet. This field used for finding the duplicates and of course as a primary key in the database
- created_at: This is the timestamp of each tweet. It contains the creation date of the tweet.
- retweet_count: This is a counter of retweets. This field contains how many times this tweet was retweeted, when the parser downloaded this tweet. That means that it counted the retweets from 1 to 7 days from the creation date, because as it was explained above only the last 7 days tweets are available for download.
- Text: The text of the tweet with the hashtags. This field used for sentiment analysis.

3.5 Database Storing

The downloaded tweets only for Apple were more than half a million. The total downloaded tweets for this project were over a million. Thus, it was essential to find a way and manipulate such great amount of data. Microsoft SQL Server is a relational database management system developed by Microsoft. As a database server, it is a software product with the primary function of storing and retrieving data as requested by other software applications, which may run either on the same computer or on another computer across a network. Microsoft markets at least a dozen different editions of Microsoft SQL Server, aimed at different audiences and for workloads ranging from small single-machine applications to large Internet-facing applications with many concurrent users.

For the project, MSSQL Server 2016 version was used. It is the last version, during the beginning of my project. Moreover, SQL code was used for preprocessing purpose, in order to delete duplicate tweets.

A simple database structure was used, in order to store the downloaded tweets. The first 4 columns of each table contain the id, timestamp, retweets count, text respectively. One more column was added later, which contains a flag. This flag is -1,0,1 and means negative, neutral and positive respectively. As expected this column was filled after the sentiment analysis.

3.6 Conclusion

To sum up, tweepy is a great open-source library which provides access to the Twitter API for Python. Although the documentation for tweepy is a bit scarce and doesn't have many examples, the fact that it heavily relies on the Twitter

API, which has excellent documentation, makes it probably the best Twitter library for Python, especially when considering the Streaming API support, which is where tweepy excels. Other libraries like python-twitter provide many functions too, but the tweepy has most active community and most commits to the code in the last year.

Last but not least, Microsoft's SQL server, one of the most known and used database server all over the world, was an essential tool of the pre-processing step and it helped me to find the best results for the purpose of the project. It is the one the fastest, if not the fastest, and the most stale db server.

4.Sentiment Analysis

The 'heart' of the project is the Sentiment Analysis. After the proper pre-processing of the data, we expect that a sentiment analysis of the tweets will predict the movement of specific stocks. The main part is a classification problem, where we 'll try to extract from the tweets the public opinion of Twitter users and use it to predict a stock movement. It is really important to pre-process the data gathered in the previous chapter in order to have an accurate result. Finally, a detailed review of the corresponding code will be described as well. The code implementation of this part is the most difficult and complicated of all the project.

In our days, Internet and especially social media platforms, are considered to be one of the most powerful tools available for data analysts. Via the internet, users can make purchases, read reviews about products, movies, events etc. After finishing their actions, most of them have the urge to express their opinion by sharing their actions with their online friends. There are many ways for someone to express their opinion while being online. Most websites offer comment sections, forums etc. However most of the users choose to express their sentiments by posting something in their favorite social media. This information is very important for companies which would like to know what the public opinion about a specific subject is. New products, movie critics and in our case stock market movement would definitely be one of the most important data to analyze.

Scott Cook, a well-known entrepreneur once said: "A brand is no longer what we tell a customer it is – it is what customers tell each other it is". By that we understand that a positive public sentiment expressed in social media can have an extremely positive effect for the brand (or in its stock as well). Nowadays, businesses choose to hire people that their only job is to monitor social media and come up with conclusions about what people think about their products and services. Of course, the amount of data is huge, so analysts

try to perform automated analysis by parsing the reviews and extracting information about the feelings of the consumers or in our case traders. This is the idea behind the term Sentiment Analysis. To be able to perform sentiment analysis we need to understand the term of Natural Language Processing (NLP) which is the main feature of this field. The purpose is to extract subjective information from pieces of text. In this project we try to find the opinion of social media users, who are actually stock market users as well or potential users. We want to know if they have a positive or negative opinion about a company.

This problem is definitely a classification problem. We want to classify tweets that refer to specific companies as positive or negative. To be more specific we are going to determine the polarity of tweets or as some call it, the semantic orientation of the text data we gathered. There are different approaches in which this problem can be addressed. Actually there are tons of different approaches. One of the main challenges of sentiment analysis is to choose the correct approach, depending on what the nature of the problem we are dealing with is. There might be a case where the design and implementation process of a model is correct, but the solution proposed in not the optimal for the specific problem. In sentiment analysis the main approach that analysts use to address this problem is:

Machine Learning approach. It is based on machine learning techniques and statistics. In this approach the human factor is less important since we don't have to give the rules beforehand. However, a training dataset with data that are already classified as positive or negative is required. With the help of an initial training data we train a classifier that will later help us classify our test data.

4.1 Pre-Processing Step

In order to perform any kind of analysis the data should be ready to be processed. Data pre-processing is an important step in the data mining process. The phrase "garbage in, garbage out" is particularly applicable to data mining and machine learning projects. Data-gathering methods are often loosely controlled, resulting in out-of-range values (e.g., Income: −100), impossible data combinations (e.g., Sex: Male, Pregnant: Yes), missing values, etc. Analyzing data that has not been carefully screened for such problems can produce misleading results. Thus, the representation and quality of data is first and foremost before running an analysis.

If there is much irrelevant and redundant information present or noisy and unreliable data, then knowledge discovery during the training phase is more difficult. Data preparation and filtering steps can take considerable amount of processing time. Data pre-processing includes cleaning, Instance selection, normalization, transformation, feature extraction and selection, etc. The product of data pre-processing is the final training set.

1. Real world data are generally
    - Incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data
    - Noisy: containing errors or outliers
    - Inconsistent: containing discrepancies in codes or names
2. Main tasks in data pre-processing
    - Data cleaning: fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies. Fill in missing values (attribute or class value):
        1. Ignore the tuple: usually done when class label is missing.

2. Use the attribute mean (or majority nominal value) to fill in the missing value.
3. Use the attribute mean (or majority nominal value) for all samples belonging to the same class.
4. Predict the missing value by using a learning algorithm: consider the attribute with the missing value as a dependent (class) variable and run a learning algorithm (usually Bayes or decision tree) to predict the missing value.
5. Identify outliers and smooth out noisy data:
6. Binning
7. Sort the attribute values and partition them into bins (see "Unsupervised discretization" below);
8. Then smooth by bin means, bin median, or bin boundaries.
9. Clustering: group values in clusters and then detect and remove outliers (automatic or manual)
10. Regression: smooth by fitting the data into regression functions.
11. Correct inconsistent data: use domain knowledge or expert decision.

- Data integration: using multiple databases, data cubes, or files.

- Data transformation: normalization and aggregation.

1. Normalization: Scaling attribute values to fall within a specified range.
2. Example: to transform V in [min, max] to V' in [0,1], apply V'=(V-Min)/(Max-Min)
3. Scaling by using mean and standard deviation (useful when min and max are unknown or when there are outliers): V'=(V-Mean)/StDev
4. Aggregation: moving up in the concept hierarchy on numeric attributes.
5. Generalization: moving up in the concept hierarchy on nominal attributes.
6. Attribute construction: replacing or adding new attributes inferred by existing attributes.

- Data reduction: reducing the volume but producing the same or similar analytical results.
    1. Reducing the number of attributes
    2. Data cube aggregation: applying roll-up, slice or dice operations.
    3. Removing irrelevant attributes: attribute selection (filtering and wrapper methods), searching the attribute space (see Lecture 5: Attribute-oriented analysis).
    4. Principle component analysis (numeric attributes only): searching for a lower dimensional space that can best represent the data..
    5. Reducing the number of attribute values

6. Binning (histograms): reducing the number of attributes by grouping them into intervals (bins).
7. Clustering: grouping values in clusters.
8. Aggregation or generalization
9. Reducing the number of tuples
10. Sampling

- Data discretization: part of data reduction, replacing numerical attributes with nominal ones.
    1. Unsupervised discretization - class variable is not used.
    2. Equal-interval (equiwidth) binning: split the whole range of numbers in intervals with equal size.
    3. Equal-frequency (equidepth) binning: use intervals containing equal number of values.
    4. Supervised discretization - uses the values of the class variable.
    5. Using class boundaries. Three steps:
    6. Sort values.
    7. Place breakpoints between values belonging to different classes.
    8. If too many intervals, merge intervals with equal or similar class distributions.
    9. Entropy (information)-based discretization.

- Remove Duplicate tweets: It was crucial to eliminate the duplicates. This issue occurs because tweeter allows you to download only the lasts 7 days tweets. Because it was not safe to download tweets every 7 days only and the parsing was made almost every day, a duplicate removal was made in the database

  Microsoft SQL Server tables should never contain duplicate rows, nor non-unique primary keys. For brevity, we will sometimes refer to primary keys as "key" or "PK" in this article, but this will always denote "primary key." Duplicate PKs are a violation of entity integrity, and should be disallowed in a relational system. SQL Server has various mechanisms for enforcing entity integrity, including indexes, UNIQUE constraints, PRIMARY KEY constraints, and triggers.

  Despite this, under unusual circumstances duplicate primary keys may occur, and if so they must be eliminated. One way they can occur is if duplicate PKs exist in non-relational data outside SQL Server, and the data is imported while PK uniqueness is not being enforced. Another way they can occur is through a database design error, such as not enforcing entity integrity on each table.

  Often duplicate PKs are noticed when you attempt to create a unique index, which will abort if duplicate keys are found. This message is:

  Msg 1505, Level 16, State 1 CREATE UNIQUE INDEX terminated because a duplicate key was found for object name '%.*ls' and index name '%.*ls'. The duplicate key value is %ls.

  This article discusses how to locate and remove duplicate primary keys from a table. However, you should closely examine the process which allowed the duplicates to happen in order to prevent a recurrence.

For this example, we will use the following table with duplicate PK values. In this table the primary key is the two columns (col1, col2). We cannot create a unique index or PRIMARY KEY constraint since two rows have duplicate PKs. This procedure illustrates how to identify and remove the duplicates.

```
create table t1(col1 int, col2 int, col3 char(50))
insert into t1 values (1, 1, 'data value one')
insert into t1 values (1, 1, 'data value one')
insert into t1 values (1, 2, 'data value two')
```

The first step is to identify which rows have duplicate primary key values:

```
SELECT col1, col2, count(*)
FROM t1
GROUP BY col1, col2
HAVING count(*) > 1
```

This will return one row for each set of duplicate PK values in the table. The last column in this result is the number of duplicates for the particular PK value.

If there are only a few sets of duplicate PK values, the best procedure is to delete these manually on an individual basis. For example:

```
set rowcount 1
delete from t1
where col1=1 and col2=1
```

The rowcount value should be n-1 the number of duplicates for a given key value. In this example, there are 2 duplicates so rowcount is set to 1. The col1/col2 values are taken from the above GROUP BY query result. If the GROUP BY query returns multiple rows, the "set rowcount" query will have to be run once for each of these rows. Each time it is run, set rowcount to n-1 the number of duplicates of the particular PK value.

Before deleting the rows, you should verify that the entire row is duplicate. While unlikely, it is possible that the PK values are duplicate, yet the row as a whole is not. An example of this would be a table with Social Security Number as the primary key, and having two different people (or rows) with the same number, each having unique attributes. In such a case whatever malfunction caused the duplicate key may have also caused valid unique data to be placed in the row. This data should have copied out and preserved for study and possible reconciliation prior to deleting the data.

If there are many distinct sets of duplicate PK values in the table, it may be too time-consuming to remove them individually. In this case the following procedure can be used:

1. First, run the above GROUP BY query to determine how many sets of duplicate PK values exist, and the count of duplicates for each set.

2. Select the duplicate key values into a holding table. For example:

```
SELECT col1, col2, col3=count(*)
INTO holdkey
FROM t1
```

```
GROUP BY col1, col2
HAVING count(*) > 1
```

3. Select the duplicate rows into a holding table, eliminating duplicates in the process. For example:

```
SELECT DISTINCT t1.*
INTO holdups
FROM t1, holdkey
WHERE t1.col1 = holdkey.col1
AND t1.col2 = holdkey.col2
```

4. At this point, the holddups table should have unique PKs, however, this will not be the case if t1 had duplicate PKs, yet unique rows (as in the SSN example above). Verify that each key in holddups is unique, and that you do not have duplicate keys, yet unique rows. If so, you must stop here and reconcile which of the rows you wish to keep for a given duplicate key value. For example, the query:

```
SELECT col1, col2, count(*)
FROM holddups
GROUP BY col1, col2
```

   should return a count of 1 for each row. If yes, proceed to step 5 below. If no, you have duplicate keys, yet unique rows, and need to decide which rows to save. This will usually entail either discarding a row, or creating a new unique key value for this row. Take one of these two steps for each such duplicate PK in the holddups table.
5. Delete the duplicate rows from the original table. For example:

```
DELETE t1
FROM t1, holdkey
WHERE t1.col1 = holdkey.col1
AND t1.col2 = holdkey.col2
```

4.2 Naïve Bayes Classification

In machine learning, naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

Naive Bayes has been studied extensively since the 1950s. It was introduced under a different name into the text retrieval community in the early 1960s, and remains a popular (baseline) method for text categorization, the problem of judging documents as belonging to one category or the other (such as spam or legitimate, sports or politics, etc.) with word frequencies as the features. With appropriate pre-processing, it is competitive in this domain with more advanced methods including support vector machines. It also finds application in automatic medical diagnosis.

Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression, which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.

In the statistics and computer science literature, naive Bayes models are known under a variety of names, including simple Bayes and independence

Bayes.[4] All these names reference the use of Bayes' theorem in the classifier's decision rule, but naive Bayes is not (necessarily) a Bayesian method.

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. It is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness, and diameter features.

For some types of probability models, naive Bayes classifiers can be trained very efficiently in a supervised learning setting. In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood; in other words, one can work with the naive Bayes model without accepting Bayesian probability or using any Bayesian methods.

Despite their naive design and apparently oversimplified assumptions, naive Bayes classifiers have worked quite well in many complex real-world situations. In 2004, an analysis of the Bayesian classification problem showed that there are sound theoretical reasons for the apparently implausible efficacy of naive Bayes classifiers. Still, a comprehensive comparison with other classification algorithms in 2006 showed that Bayes classification is outperformed by other approaches, such as boosted trees or random forests.

An advantage of naive Bayes is that it only requires a small number of training data to estimate the parameters necessary for classification.

4.3 Correlation between tweets and stocks

The main purpose of this project is to find a correlation between the tweets and the stock of a company. In general, we want to find out if the opinion of the social media users for a company affects the stock movement of this company. So, after the parsing and the classification, the next big step was to analyze the results and find out a correlation.

This analysis was made manually, because it is really hard to code such a case. Moreover, with simple SQL queries I can get directly a lot of information and this was very helpful to find the best scenario. Moreover

4.3.1 Variance

It was crucial to find a variance for each tweet. Especially in stock-markets not every tweet counts the same. For example, a tweet for Apple from The Financial Times would affect more traders than a tweet of a simple user. Known Newspapers or large organizations which many people believe and read a lot, are called authorities. These authorities, have many followers who spread the opinion of authorities. In Twitter, users can repost a tweet and this a way to count the variance of each tweet. In this project we would like to give a variance in each tweet, in order to give more importance to authorities. One way to track such value is the retweets count. Every post has as a parameter the retweets count during the download. So, in every tweet we used the below variance equation:

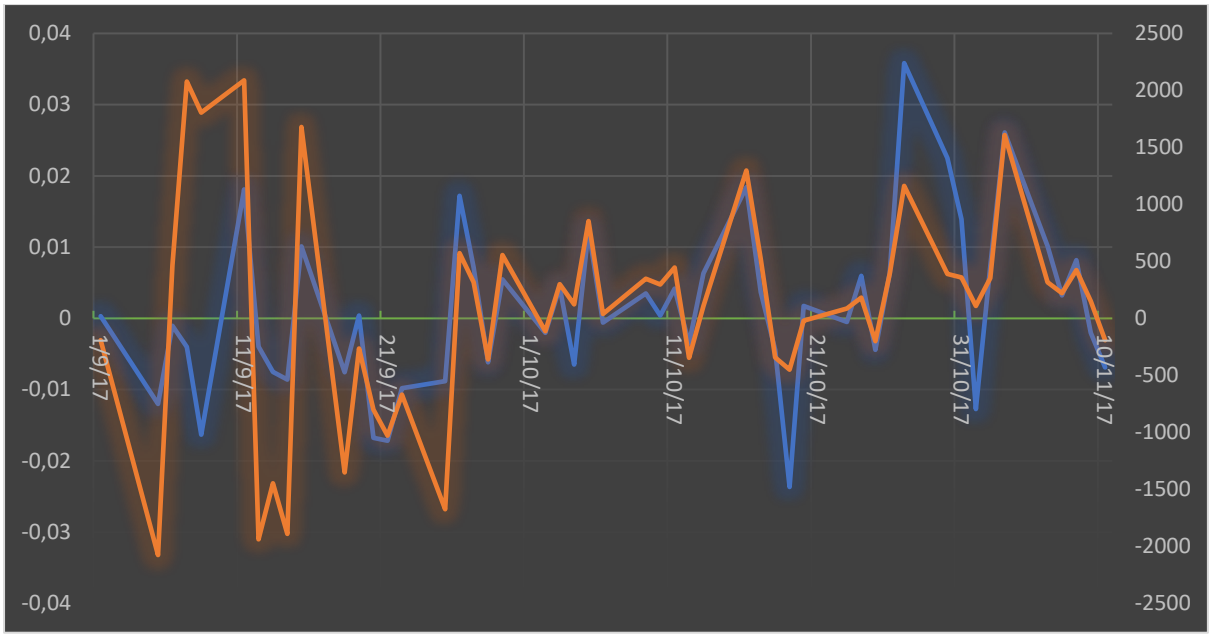
$$VAR = retweets\_count + 1$$

So, now a tweet with 600 retweets, which affects more traders than another's with 0 retweets will count 601 times, while the second will count only 1. Moreover, with this variance we do count the 0 retweeted posts but with a small variance.

4.3.2 The Apple stock

Apple is one of the most known company all over the world. Moreover, it seems the last few years, that it is a good investment stock for non-expert traders. This is actually the advantage for this project, because we want to analyse stocks with many traders who don't have the expertise and get influenced easily.

The correlation between the sentiments and the actual stock price was made daily. That means that the collected tweets from the past 24 hours will be analysed, in order to predict the next day's stock movement. The time slot in which the tweets should be counted is not clear. So, I don't count tweets until 23:59 for the next day's prediction. This was the initial plan, but after testing the best gap was from 06:00 AM until the next 24 hours will count in the daily stock movement. Below I will present the stock movement and the sentiment analysis of each day for Apple Stock. The data for sentiment analysis begin from 01/09/2017 (which means tweets from 06:00AM on 31/08/17 until 05:59 AM on 01/09/17).



In the above chart we can see the stock movement (blue line) and the positive-negative tweets (orange line). In the left side and horizontal, we have the percentage of the stock movement, with max values 3,58% on 27/10/2017 and min value 2,36% on 19/10/2017. In the right side we have the positive – negative tweets, with max value 2088 more positive than negative tweets on 11/09/2017 and a minimum value of -2076, which means 2076 more negative than positive tweets on 05/09/2017.

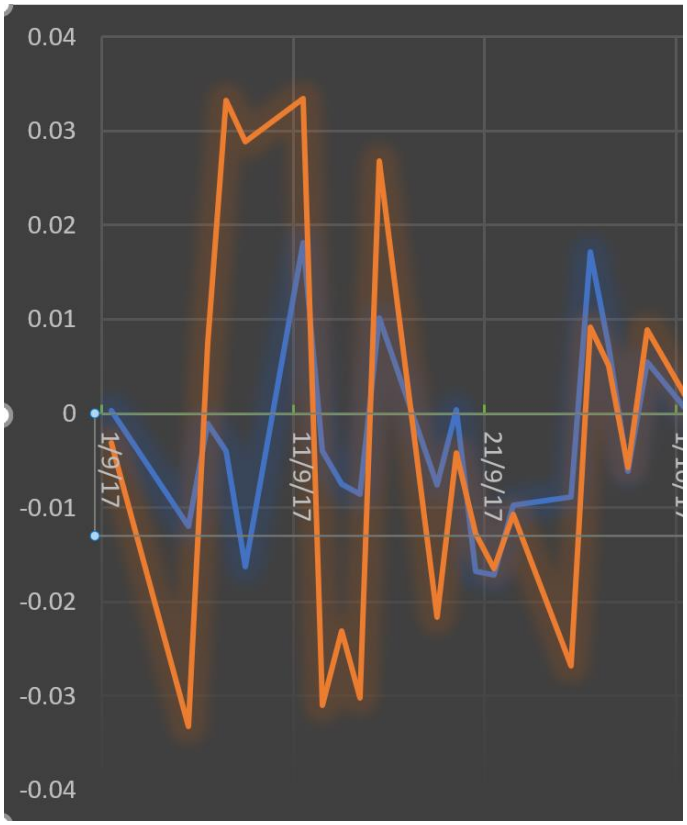
In more detail:

- Up / Down success: 39 out of 50 stock market working days
- 6 out of 11 unsuccessful prediction have
    - >= 0,5% stock movement
    - >= 260 positive-negative value (AVGp = 3371, AVGn = 3285)

The 78% is a promising percentage and if we see the small differences in the false predicted days, then we can say that the sentiment analysis worked well for this stock. Of course, this definitely not a proper or safe way for a trader to predict the movement of the stock, but still for the purpose of the project the

results are very good. Now, we will see in more detail the chart and comment about it in 2 periods.

- 01/09/2017 to 30/09/2017



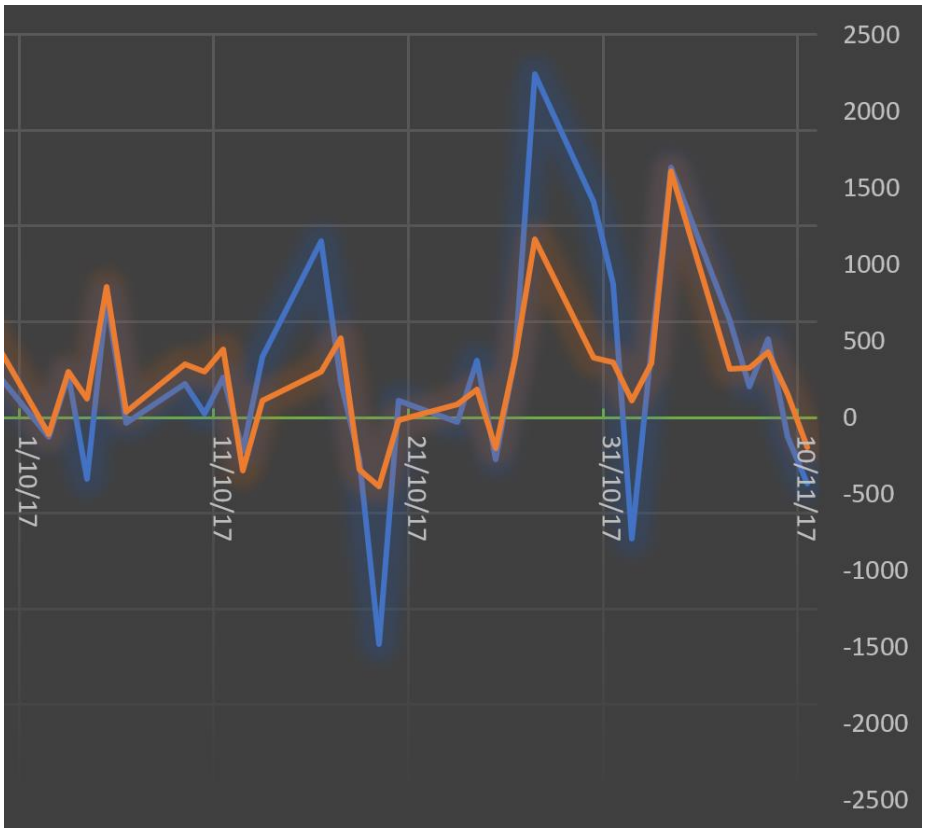
The Data Gathering started at 31/08/2017 for the hashtag Apple. The python code for the Search API and the storing should be ready early because it was very important, in order to have a significant period to analyse. As it was explained above the 7 days limit from Twitter was the main issue for that. During this period (September) I was running the parsing algorithm every 3-4 days and not every day. There were some days when the gathering had executed every day.

The parameter retweets_count contains the value of retweets during the download. Because the downloads were not executed in similar gaps, the retweets_count contain the sum of 3 days for some tweets, 4 for others or even

1 for some other. That occurs an issue. The heterogeneity of this value and this period. As we can see also, this period has the more dissimilarity than the rest where the parsing has been executed every day, in order to solve this issue. Nevertheless, this period was included in order to see how this parameter may affect the results. In this period, the 5 out of the 11 unsuccessful were included and the most of them with large differences.

It doesn't mean that this parameter affects the results but we can see clearly the large dissimilarity. This fixed in the rest gathering, because the python code was able to be executed easily every day.

- 01/10/2017 to 10/11/2017



In this period the tweets were not only better for prediction, but the 2 lines were closer as well. That means that a day with a lot of positive tweets had also a big rise in the stock value. In this period, we have the 6 out of 11 unsuccessful predictions in with the 4 out of 6 have less than 0,5% stock movement and in

the same time less than 260 positive-negative absolute value (AVGpos = 3371, AVGneg = 3285)

5. Testing & Evaluation

This chapter contains two parts. The first part is the testing and includes all the necessary tests on the written code, in order to make sure that the code for both Data Gathering and Sentiment Analysis can be executed in any case without problem. The second part is the evaluation and includes all the techniques required to evaluate if the sentiment analysis and its results are correct. Actually, this evaluation is for the whole project and not only for the sentiment analysis, because it will evaluate if the method and the way which followed can provide correct results or not.

5.1 Testing

The testing performed in both Data Gathering and Sentiment Analysis python code. For the Data Gathering, I execute the code, with a lot of parameters and a lot of times before starting downloading, in order to ensure that the parameters in the queries were transferred to the Search API and the downloaded tweets include only the requested info. The main parameters I used is the language (English only) and the timestamp, but only in the second period, as it was mentioned in the chapter 4.3.2. All the available parameters are included to the Cursor function.

```
for status in tweepy.Cursor(api.search, q='#Apple',lang='en').items(100000):
```

In the above query, we are searching the first 100000 tweets in English language, having the hashtag #Apple. Unfortunately, the testing was a manual work and as in many coding projects this part takes really long time. The only way to evaluate the code is the continuous testing.

5.2 Evaluation

The evaluation of the results in this prediction project is made by comparing the actual data with the predicted. This was the purpose of the project as well. Having real data is the most accurate way to evaluate the predicted data. In the following table we can analyse the results. In the first column is the stock market day. Second row, contains the stock price of this day and in thee third we can find the price difference from the previous day. In the fourth column we can see the stock movement in percentage and colour as well. Finally, the last column contains the positive-negative value, which is coloured as well.

1/9/2017	164,050003	0,050003	0,030%	-192
5/9/2017	162,080002	-1,970001	-1,201%	-2076
6/9/2017	161,910004	-0,169998	-0,105%	466
7/9/2017	161,259995	-0,650009	-0,401%	2078
8/9/2017	158,630005	-2,62999	-1,631%	1806
11/9/2017	161,5	2,869995	1,809%	2088
12/9/2017	160,860001	-0,639999	-0,396%	-1939
13/9/2017	159,649994	-1,210007	-0,752%	-1446
14/9/2017	158,279999	-1,369995	-0,858%	-1889
15/9/2017	159,880005	1,600006	1,011%	1678
18/9/2017	158,669998	-1,210007	-0,757%	-1350
19/9/2017	158,729996	0,059998	0,038%	-263
20/9/2017	156,070007	-2,659989	-1,676%	-801
21/9/2017	153,389999	-2,680008	-1,717%	-1027
22/9/2017	151,889999	-1,5	-0,978%	-670
25/9/2017	150,550003	-1,339996	-0,882%	-1673
26/9/2017	153,139999	2,589996	1,720%	573
27/9/2017	154,229996	1,089997	0,712%	317
28/9/2017	153,279999	-0,949997	-0,616%	-361
29/9/2017	154,119995	0,839996	0,548%	555
2/10/2017	153,809998	-0,309997	-0,201%	-112
3/10/2017	154,479996	0,669998	0,436%	299
4/10/2017	153,479996	-1	-0,647%	122
5/10/2017	155,389999	1,910003	1,244%	852
6/10/2017	155,300003	-0,089996	-0,058%	38
9/10/2017	155,839996	0,539993	0,348%	349
10/10/2017	155,899994	0,059998	0,038%	297
11/10/2017	156,550003	0,650009	0,417%	445
12/10/2017	156	-0,550003	-0,351%	-346
13/10/2017	156,990005	0,990005	0,635%	109
16/10/2017	159,880005	2,89	1,841%	296
17/10/2017	160,470001	0,589996	0,369%	518
18/10/2017	159,759995	-0,710006	-0,442%	-344
19/10/2017	155,979996	-3,779999	-2,366%	-449
20/10/2017	156,25	0,270004	0,173%	-21
23/10/2017	156,169998	-0,080002	-0,051%	88
24/10/2017	157,100006	0,930008	0,596%	182
25/10/2017	156,410004	-0,690002	-0,439%	-199
26/10/2017	157,410004	1	0,639%	392
27/10/2017	163,050003	5,639999	3,583%	1161
30/10/2017	166,720001	3,669998	2,251%	390
31/10/2017	169,039993	2,319992	1,392%	360
1/11/2017	166,889999	-2,149994	-1,272%	108
2/11/2017	168,110001	1,220002	0,731%	353
3/11/2017	172,5	4,389999	2,611%	1609
6/11/2017	174,25	1,75	1,014%	317
7/11/2017	174,809998	0,559998	0,321%	321
8/11/2017	176,240005	1,430007	0,818%	424
9/11/2017	175,880005	-0,36	-0,204%	152
10/11/2017	174,669998	-1,210007	-0,688%	-192

As it was explained in the 4th chapter the 78% successful prediction is a very good result, which can be evaluated from the actual stock movement.

6.Conclusion

This project was one of the most challenging I ever worked on. Its purpose was to predict a stock's movement by using sentiment analysis to tweets based on this stock. Data Gathering was too crucial to start early in the project and develop the corresponding python code for that. Python, especially the tweepy library and Search API were the perfect tool to do that.

Then, the sentiment analysis performed in all tweets and stored in a Microsoft SQL Database which helped me a lot for the pre-processing step and of course to find the best results as well. After a lot of testing a 78% successful prediction is a really good and promising result, but as I explained in the presentation, this method can work under only a few cases.

Apple stock attracts the most unexperienced trades and people who want to invest a small amount of money in a 'safe' tock as Apple's. Such traders are affected a lot from the internet news and newsfeed, in which I gave a significant variance. Moreover, the period was the proper for analysis, because the company introduced 3 new devices and a lot of people commented about them in social media.

So, in the question if the stock movement prediction is possible the answer is definitely no, but analysing one of the most unstable parameter, like the traders intension, can give you a really good aspect of their opinion.

7.Bibliography

1. Alfred Cowles, *Stock Market Forecasting*, Econometrica,1944, pp. 206-214
2. https://developer.twitter.com/en/docs/tweets/sample-realtime/api-reference/get-statuses-sample (accessed 01/12/2017)
3. https://support.microsoft.com/en-us/help/139444/how-to-remove-duplicate-rows-from-a-table-in-sql-server (accessed 30/11/2017)
4. E. Schoneburg, *Stock price prediction using neural networks*, Neurocomputing 2 (1990) pp. 17-27
5. http://thinknook.com/twitter-sentiment-analysis-training-corpus-dataset-2012-09-22/ (accessed 13/11/2017)
6. https://apps.twitter.com/app/14061092/keys (accessed 25/08/2017)
7. CF Tsai, YC Hsiao, *Combining multiple feature selection methods for stock prediction: Union, intersection, and multi-intersection approaches,* Decision Support Systems, 2010 - Elsevier, pp. 258-269
8. J Si, A Mukherjee, B Liu, Q Li, H Li, X Deng, *Exploiting Topic based Twitter Sentiment for Stock Prediction,* ACL (2), 2013, aclweb.org