INTERNATIONAL HELLENIC UNIVERSITY

Forecasting with Predictive Social Media Analytics

Eleana Tsiara

SID: 3301160007

SCHOOL OF SCIENCE & TECHNOLOGY

A dissertation submitted for the degree of

Master of Science (MSc) in Information and Communication Technology Systems

DECEMBER 2018

THESSALONIKI – GREECE

Acknowledgments

I would like to thank my supervisor, Mr. Christos Tjortjis, for his patience and support throughout the duration of the project, Fotis Touparis and Lazaros Oikonomou for their guidance and useful comments. Finally, I am grateful to Twitter for accepting my request and allowing me to use its data for the practical part of my dissertation.

Abstract

With the advent of social media concepts such as *forecasting* and *nowcasting* became part of the public debate. Past successes include predicting election results, share price movement in the stock market and forecasting many other events or behaviors. This project aims at using social media data, and specifically data from Twitter that are related the songs and artists that currently appear on the highest 10 ranks of the Billboard Hot 100 chart, perform sentiment analysis on the collected tweets, classify them as positive or negative and finally utilize this information to generate predictions about the chart of the following weeks. In more detail, the goal is to investigate the relation between the number of mentions of a song and its artist, as well as the semantic orientation of the relevant posts and the performance of the song on the next chart. Firstly, the problem was approximated through regression analysis, which estimated the difference between the actual and predicted positions and yielded moderate results. Secondly, the task was specialized into providing forecasts for some ranges of the chart, namely, for the top 5, 10 and 20 positions of the chart. According to the values of accuracy and F-score metrics and compared to previous research, the findings can be deemed as satisfactory, especially for the predictions of the top 20 hits.

Eleana Tsiara

12/07/2018

Contents

List of Tables

List of Figures

1 Introduction

During the last few years social media have not only made a dynamic appearance, but have penetrated our everyday lives to the point that they constitute an integral part of our daily routine. These microblogs are a medium that allows users to communicate and express themselves by sharing content, which is not limited to files, but additionally includes other information, like thoughts and opinions.

The vast amount of data that becomes available through these services can be useful in many ways, and as a result social media have attracted the attention of many companies who seek to exploit all this information for their benefit. In more detail, companies want to know what customers think of them and their opinion on their products or services that they provide. This would help them understand what kind of improvements should be made from their side in order to ultimately increase their revenue. Apart from that, social media can be utilized in a variety of different use cases, such as the prediction of future elections, forecasting the commercial success for movies to be released and stock market prediction.

In this dissertation, the social media network that was chosen to be examined, in order to generate predictions about the Billboard chart, is Twitter. The reason was that it is one of the most successful networks, with millions of active users, and it includes mainly content in the form of text, making it more suitable for opinion mining, compared to other social media applications, such as Facebook and Instagram, which emphasize more on images or videos. What is more, Twitter offers developers the opportunity to exploit its data, as it encompasses almost ready to use APIs to gather and manipulate tweets.

2 Background

The following sections include a few background information on the platforms and methodologies that were used and aim at helping the understanding of the practical part of the dissertation.

2.1 Twitter

Twitter is an online networking service that was founded in 2006 by Jack Dorsey, Evan Williams and Biz Stone. The company started with only a couple of employees but now includes more than 400 members and they are constantly increasing. About 200 million people are connected to Twitter around the globe and roughly 160,000 new users join the service each day. It is estimated that the number of tweets posted per day is more than 140 million, which adds up to almost a billion tweets on a weekly basis.

The basic philosophy of Twitter is publicity. Users can instantaneously share their thoughts, opinions, news, reply to others or retweet, in other words share, a post they feel that expresses them. Its name actually stands for "*a short burst of inconsequential information*".

Any user can post a tweet, which can be characterized as a small burst of information, on her profile. Initially there was no limit to character length, which changed when the service went public and a limit of 140 characters was set. However, on 2017 the limit was doubled, reaching up to 280 characters for most languages. Tweets are mainly text-based, but users have the option to share photos, GIFs, videos and links. The latter are usually shortened, since URLs are usually quite long, and they may not leave enough space for the text that will accompany it.

Twitter includes a variety of communication options for its users. Hashtags are a characteristic example; they are frequently found in tweets and their usage is generally very common in social media networks. A hashtag simply consists of the hashtag symbol (#) followed by a keyword that can include one or more terms, without spaces between them, and it constitutes a means of creating a thread revolving around a particular topic. This way, users can easily share their opinions and follow themes they are interested in.

2.1.1 Twitter APIs

Twitter focuses on the privacy of its users. Creating new application can only be achieved after applying for a developer account and getting approval. In order to achieve that, a form should be filled with information about the desired usage of the account. Specifically, the required fields are the following:

- Account usage: for personal use or usage in the context of an organization, business or institution.
- Types of use cases that the developer is interested in. For example, academic, advertising, customer experience or trend analysis.
- A detailed description of the product or service that should include the purpose of the whole project, the deliverables, the type of analysis that is planned to be performed on the gathered content and the level of disclosure concerning Twitter data (Figure 1).
- Stating any cooperation and possible information sharing with government entities.

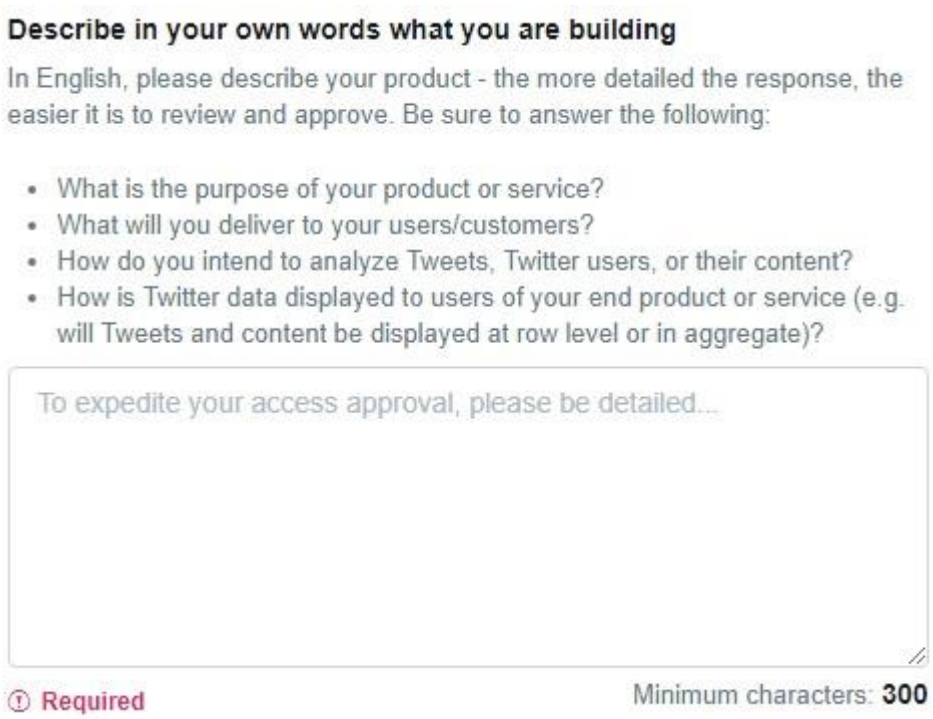

Figure 1: Screenshot of the form for the description of the project

After that, the user should create an application, in order to receive his credentials and proceed to the usage of the APIs. In this case, the required details include:

- Application name
- Application description visible to users
- Website URL
- Detailed application description visible only to Twitter employees

Streaming API

The standard version of Twitter's Streaming API is useful for searching and retrieving Tweets, in a similar way that the Search UI works. However, there are some limitations, since data can be gathered only if they have been published during the last 7 days and some results may not appear, as relevance is a key priority and completeness is secondary.

Interacting programmatically with the Streaming API demands the installation of the *tweepy* Python library. The *OAuth* interface authorizes a user application to gain access to Twitter and this is where the user credentials should be inserted (Figure 2).

```
import tweepy
from tweepy import OAuthHandler

access_token = 'YOUR-ACCESS-TOKEN'
access_token_secret = 'YOUR-ACCESS-SECRET'
consumer_key = 'YOUR-CONSUMER-KEY'
consumer_secret = 'YOUR-CONSUMER-SECRET'

auth = OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
```

Figure 2: Snippet of code for gaining access to Twitter with the *OAuth* interface

The code in Figure 3 defines a basic listener that prints the received tweets on the command line. Next, an attempt to connect to the Twitter Streaming API is made.

```
from tweepy.streaming import StreamListener

class StdOutListener(StreamListener):

    def on_data(self, data):
        print data
        return True

def on_error(self, status):
        print status
```

```
l = StdOutListener()
stream = Stream(auth, l)
```

Figure 3: Code for creating a listener and connecting to the Streaming API

Twitter Streams are filtered to capture data by specific keywords. In Figure 4 we want to collect tweets related to the artist Drake.

```
stream.filter(track=['drake'])
```

Figure 4: Collecting tweets with a specific keyword

The output on the command line can be saved to a txt file by typing the command in Figure 5.

```
python twitter_streaming.py > twitter_drake_data.txt
```

Figure 5: Saving data to a file

Search API

Following, the parameters of Twitter's Search API:

- **q:** This mandatory parameter is the search string which contains the term or combination of terms used as keywords in order to retrieve the tweets with the matching text. The string utilizes a UTF-8 and URL encoding and has a length limit of 500 characters, counting in operators. Additionally, the search query is subject to complexity limitations. For instance, it was noticed that it was impossible to combine more than 30 terms with logical operators.
- **geocode:** An optional field that forces a geographical restriction on the origin of the returned results. Only tweets posted by users which are located within the specified area will be captured by the API. An area is defined by its coordinates, meaning the latitude and longitude, and its radius, which can be expressed either in kilometers or miles.

- **lang:** When this parameter is set, the returned results should be of the specified language. The accepted values are defined by the ISO 639-1 code and language filtered is implemented in a best-effort manner.
- **locale:** A nonobligatory parameter for the definition of the language of the query. The only functional option at the moment is *ja*.
- **result_type:** This optional field can take only three values:
    - *recent:* Only the most recent tweets are returned.
    - *popular:* Most popular tweets will be prioritized over recent results.
    - *mixed:* This is mixture of both recent and popular results and is the default value for this parameter.
- **count:** With a default value of 15 and a maximum value of 100, this optional field is used to specify the number of tweets to be returned for each page.
- **until:** It is used to only select tweets created before the given date. The format used should be *YYYY-MM-DD*. However, this cannot override Twitter's default 7-day limit and it just restricts the timeframe of the returned data even further.
- **since_id:** This parameter sets a maximum threshold for the ID of the results. Only records with a greater (translated to newer) value will be returned. Since Twitter's API has already set a limit to the maximum number of tweets that can be retrieved, using this parameter will impose further restrictions and may be automatically reset to a new, older value.
- **max_id:** This parameter sets a maximum threshold for the ID of the results. Only records with an equal or smaller (translated to older) value will be returned.
- **include_entities:** An optional Boolean parameter defining the inclusion or exclusion of the *entities* node.

2.2 Sentiment Analysis

Sentiment analysis, also known as opinion mining or subjectivity analysis, can be defined as the examination of unstructured text and the extraction of opinions, emotions and subjectivity, with the purpose of deriving a favorable or unfavorable statement and classifying the attitude of the author as positive or negative, towards the subject mentioned in the current piece of writing. Categorizing the semantic orientation of a text segment in one of these two opposite classes is the process of polarization.

A common practice is to determine the polarity with the application of a scaling system, where the piece of text receives a score. For example, when using a 0 to 10 scale, if the emotional analysis provides a rating smaller than 5, the attitude of the author leans towards a negative disposition, while a score greater than 5 would reflect positivity. The advantage of this method is that the result is not limited in only two distinct states, but there is the chance to assign a value that indicates the weight of the current emotion. It is much more realistic that a text segment might not be downright positive or negative but may fall somewhere in the space between. It should be noted that an alternative solution is to introduce an additional neutral class and classify the text in one of these three emotional dispositions, based on the given score.

A more advanced form of sentiment classification deals with the identification of specific emotions, such as anger, happiness and sadness [12].

The concept of sentiment analysis goes hand in hand with natural language processing (NLP), which is the analysis of pieces of text written in human language by computers. For instance, words like sad or mad are emotionally charged and are definitive for the polarity score of sentences. So, the inclusion of such words would indicate that a statement is negative, whereas the presence of words with positive meaning, like love and like are strong indicators of positive emotions. The number of these words is also important, to determine the level of positivity or negativity and to discern the emotional inclination when many contradictory words appear in the same piece of text.

The most common approaches can be divided in two main categories: rule-based and machine learning.

In rule-based sentiment analysis, the analyst is required to define a set of his own rules for determining the polarity of the text. This approach relies highly on the human factor and produces a result that is specifically tailored for a particular dataset. However, it is a task that demands a lot of knowledge and research from the side of the analyst. Especially when dealing with a large volume of data that might be in an unstructured form, hard-coding rules becomes very time-consuming and error-prone and the optimal distribution of effort to each case is not easily achieved by an individual. Furthermore, this approach is quite rigid in the event of an extension or improvement, for example, if more data is provided for input or if it is desired to increase the accuracy of the classification. Such changes require the generation of a greater number of more complex rules, which definitely makes the tasks of managing and maintaining them very cumbersome.

On the other hand, the machine learning approach is based on automated processes. A corpus or lexicon with paradigms of classified data is necessary in this technique, in order to infer the set of rules that will be later used for the classification of the test dataset. Obviously, automation has a lot of benefits, such as the speed of the process, the handling of misspelled words, discrepancies and other flaws of the input data, the opportunities for improvement in terms of accuracy or expansion and generally the whole robustness of the classification.

Of course, each case is different, so the decision about which approach should be chosen depends on the special characteristics of every dataset and the objectives that should be achieved [9].

2.2.1 Lexicons

A sentiment lexicon can be described as a list of linguistic elements that can be symbols, words or phrases, where each instance receives a label or a score according to the emotion it expresses. Specifically, each element can be characterized as positive or negative, but it can also be characterized more accurately based on the strength of its emotion, meaning the intensity of the positivity or negativity.

Some of the features that impact the sentiment analysis of text are described below:

- **Word-sense disambiguation:** a word can have multiple meanings, and this is the process of identifying the correct one in the context that this word is being used. The surrounding text is therefore important, and the word should not be judged as a standalone term.
- **Punctuation marks:** especially the exclamation point, is inherently a means of enhancing the meaning of a sentence.
- **Capitalization:** similarly, it emphasizes on specific words or phrases and adds up to their emotional intensity.
- **Negation:** it reverses the meaning of a phrase and, thus, it has the same effect on its sentiment.
- **Contrastive adjectives or conjunctions:** they essentially split sentences into parts with opposite sentiment orientations, so each separate part should contribute differently to the sentiment score of the whole sentence.

- **Intensity or scale adverbs (like very, extremely, completely, etc.):** they are used to express the positive or negative intensity of the following noun and should be adjusted accordingly.

VADER

VADER (Valence Aware Dictionary for Sentiment Reasoning) is an open-source sentiment analysis tool under the MIT license, developed by Hutto C.J. and Gilbert Eric and is mainly suitable for analyzing social media texts. This kind of content is commonly characterized by the wide use of acronyms, initialisms, emoticons and slang terms. Despite that it specializes in social media networks, VADER can successfully be utilized in different contexts.

The lexicon borrowed some lexical elements from other trustworthy sources and extended the list with features that are frequently used in microblog texts. The lexicon was validated by humans. Out of the over 9000 elements that were initially generated, approximately 7500 features were kept, and the rest were dismissed due to their neutrality. Each feature of the list was rated using the "*Wisdom of The Crowd"* method, where volunteers are prompted to perform easy tasks and receive a minor payment. This way a group of individuals was employed to rate the collected lexical features. Raters first received a sentiment training and were validated in terms of English competency, the ability to identify emotions and their tendency to be objective.

For the evaluation process, VADER was used in the sentiment analysis of four lists related to different domains: social media, movie reviews, technical product reviews and opinion news articles. It was later compared to the Linguistic Inquiry Word Count (LIWC), General Inquirer (GI), Affective Norms for English Words (ANEW), SentiWordNet (SWN), SenticNet (SCN), Word-Sense Disambiguation (WSD) using WordNet, and the Hu-Liu04 opinion lexicons. The results showed it had great performance, achieving better than all the other lexicons in most categories. Notably, in the social media domain which is its specialty, not only did it perform exceptionally well compared to the rest of the lexicons, but it also exceeded the scores of human raters in most metrics.

The simplicity of the lexicon is also one of its advantages, since it makes it lightweight and it can be run in most computer systems in a short time, without sacrificing its performance, compared to other similar methods. In fact, its speed allows it to be used in real time when handling streaming data online. VADER does not require a training da-

taset and is relatively easy to use [6]. The source code for the manipulation of the lexicon is written in Python but there are also ports in order to use the lexicon with other programming languages, specifically Java, JavaScript, PHP and Scala.

VADER can be installed using the *pip* packet management system or by manually downloading the package with all its files from Github [17].

The lexicon file is comprised of a list of textual elements, each one followed by its mean sentiment rating and standard deviation which were calculated based on the evaluation of ten independent humans. Any user that wishes to enrich the lexicon should follow the same pattern in order to ensure consistency. Besides, authors are willingly offering their piece of work to anyone who is looking for resources to create his own lexicon.

Another file is a script dedicated to the implementation of heuristics processes for increasing the accuracy of emotional analysis. VADER considers punctuation marks, idioms and wide sets of words that serve as negation terms and intensifiers.

The package includes four lists with text related to the domains that were mentioned before:

- 4,000 actual tweets combined with 200 made-up tweet-like texts, which were added to enhance the diversity of the data.
- 5,190 sentences extracted from 500 opinion-related editorials and articles coming from the New York Times.
- 10,605 sentences coming from 2,000 movie reviews, half of which were positive and the other half negative, taken from rotten.tomatoes.com.
- 3,708 sentences originating from 309 reviews found in amazon.com, made by customers regarding 5 products.

Each file has a sentiment score calculated by VADER for each chunk of text it contains and it is accompanied by another file with the corresponding ratings for each element coming from 20 individuals, capable of providing accurate estimations.

VADER computes the final compound score of a sentence by summing the individual textual pieces, weighting each one accordingly, and normalizes the value so that it ranges from -1 to +1. Typical thresholds of the compound value for characterizing the sentiment orientation of a chunk of text are the following:

- **<= -0.05:** negative

- **> -0.05 and < 0.05:** neutral
- **>= 0.05:** positive

Additionally, VADER provides three more metrics for the separate positive, negative and neutral score of a sentence.

SentiWordNet

A sentiment analysis tool first introduced in 2006 by S. Baccianella and F. Sebastiani. With the collaboration of A. Esuli, the team finally released SentiWordNet 3.0, which is the current version.

The lexicon consists of a list of synsets, which means that each row contains one or a group of words that are cognitive synonyms and refer to a specific context. Including synsets instead of individual terms is a method to tackle the word-sense disambiguation problem. Each synset is associated with three scores indicating how positive, negative and neutral it is. The authors actually consider synsets to be divided into two major categories: objective (or neutral) and subjective, which is further separated into positive and negative. A statistical analysis of the lexicon's first version, showed that moderately objective synsets comprised about 24.63% of it and that number drops significantly in proportion to the increase of the subjectivity score, meaning more positive and negative terms. Dividing the synsets by part of speech, revealed that 39.66% of adverbs and 35.7% of adjectives were characterized as partially or highly subjective, while the same scores for verbs and names were 11.04% and 9.98%, respectively. This fact underlines the strength of adverbs and adjectives in determining the opinion of a piece of text [5].

Moving on to SentiWordNet 3.0, the annotation of the lexicon comprises of two steps: the use of a weak-supervision, semi-supervised algorithm and the usage of the result as an input to an iterative random walk algorithm that will terminate when it reaches convergence. The glossary that was initially used in the random walk step process, which was WordNet, has been replaced with the manually disambiguated Princeton WordNet Gloss, as it is considered more reliable.

SentiWordNet was evaluated using the *Micro-WN(Op)* corpus, with some mapping adjustments implemented beforehand, which consists of a list synsets and a negative, positive and neutral score for each group of elements. From the comparison of the scores of both datasets, SentiWordNet achieves an improvement of 17.11% in positivity ranking and 19.23% in negativity ranking against its previous version [2].

2.3 Regression and Classification

Data mining is the process of discovering interesting, useful and usually unexpected patterns out of a dataset with techniques that fall under the category of statistics, machine learning and database systems. The concept is associated to predictive modeling, which from a mathematical point of view, refers to the mapping of input variables to output variables using the most efficient function based on the amount of available resources. Predictive modeling can be divided into two major tasks: regression and classification.

In regression tasks the objective is to approximate a mapping function for matching input variables to a continuous-valued output variable, which can be either an integer or a real number. A variety of applications require the prediction of numeric values that could be related to finance, sizes, population or any other amount. Regression models can be evaluated and compared to each other using the convergence of the predicted from the actual values, depicted by some error-related metrics, such as mean absolute error (MAE) and root mean squared error (RMSE).

In contrast to regression, the task of classification is not to predict a quantity but to assign each instance with a set of input variables to a discrete-valued variable. The target variable is often called class, category or label and it can include two (binary classification problem) or more (multi-class classification problem) discrete classes. Usually, the prediction of the response variable is materialized through the generation of a continuous value expressing the likelihood that an instance will belong to each of the available classes, and finally assigning it to the class with the highest probability.

In order to describe the metrics of evaluating a classification algorithm it is better to briefly explain the meaning of the confusion matrix, which is a visualization of the predicted and actual classes. The confusion matrix in Table 1 is an example of a matrix for a binary classification problem.

Table 1: Confusion matrix for a binary problem

		Predicted Class	
		Positive	**Negative**
Actual Class	**Positive**	True Positive	False Positive
	Negative	False Negative	True Negative

The meaning of each cell is explained below:

- **True Positive (TP):** The number of instances that were correctly predicted to belong to a class.
- **False Positive (FP):** The number of instances that were incorrectly assigned to a class.
- **False Negative (FN):** The number of instances that were mistakenly predicted not to belong to a class.
- **True Negative (TN**): The number of instances that were predicted not to belong to a class and that actually do not.

Considering the previous terms, the following metrics can be defined:

- **Accuracy (Recognition Rate):** The percentage of true classifications, meaning the instances that were correctly classified to the appropriate class, divided by the total number of predictions:

$$\frac{TP + TN}{TP + FP + FN + TN}$$

- **Precision:** The percentage of correctly classified positive instances out of the total number of positive examples:

$$\frac{TP}{TP + FP}$$

- **Recall:** The ratio of the instances that are actually positive to the examples that were correctly or incorrectly classified as positive:

$$\frac{TP}{TP + FN}$$

- **F measure (F1 or F-score):** It offers a weighted average of precision and recall. Since these two metrics usually have an inverse relationship, the F-score is a convenient way to harmonize them and utilize them both at the same time:

$$\frac{2 * Precision * Recall}{Precision + Recall}$$

Classification and regression are two different techniques, which may present a few similarities and there is small set of algorithms that are suitable for handling both types of tasks, but ultimately, they differ in the way the results are interpreted and evaluated. It is a common practice to convert regression into classification problems through a pro-

cess called discretization. This results in the division of continuous values to a set of buckets, according to specific numeric ranges, that represent different labels.

2.3.1 Correlation Coefficient

Correlation coefficient is a metric indicating the relationship between two variables. The absolute value of the metric shows the strength of the relationship, while a negative or positive value signifies its direction and it ranges from -1 to 1. In more detail, the value of the correlation coefficient can be interpreted based on these 3 reference points:

- **1:** This is a perfect positive correlation. The increase of one variable will lead to the proportionate increase of the other variable.
- **0:** This indicates that there is no relation between the two variables, so the increase or decrease of the first variable will have no impact on the second one.
- **-1:** This is a perfect negative correlation. The increase of one variable will lead to the proportionate decrease of the other variable.

Some widely accepted guidelines for characterizing the correlation coefficient more accurately are presented below:

- **0 – 0.3:** These values indicate a weak positive correlation.
- **0.3 – 0.7:** When the value is in this range the correlation is considered moderate positive.
- **0.7 – 1:** This range includes correlation values that are considered strong positive.

A negative correlation coefficient values can be interpreted using the respective negative ranges in the same manner.

3 Literature Review

In [1], the authors used the Twitter chatter to predict the commercial success of movies. The choice of this specific industry was based on the number of discussions and the differences of opinions around the subject, as well as the convenience of obtaining financial information about movies and therefore evaluating the outcome of the research. The hypothesis was that the overall attention and the volume of positive tweets are determining factors for the success of a movie. After exploiting about 3 million tweets, using a linear regression model and performing a sentiment analysis, they concluded that only their first assumption was verified, which indicates a correlation between the fame of a movie prior to its release and the future revenue it will produce.

In [3], sales prediction targets at the famous company Nike. It is perceived as an ideal brand for social media forecasting because of its publicity and the sense of community that the majority of sports enthusiasts share, which ensures a lot of online activity and a variety of opinions. In detail, the authors wanted to estimate the impact of each variable on a Facebook page, like posts, comments and likes, from a group of Nike's pages, examining each variable and each page individually and as a combination for all variables and all pages. Moreover, they were interested in the predictive impact of search query data and the relation between Nike's events and the subsequent Facebook activity.

According to their findings, the simple regression scored as high as the Bloomberg forecasts in terms of accuracy for predictions pertaining to the near future. An interesting pattern was observed, since the high accuracy was also achieved for cases in the far distant future. This could be simply interpreted by taking into account that the time from the moment a user notices a product until he proceeds to the actual purchase might be very short or long, depending on the type of the item. On the contrary, the multiple regression method did not yield reliable results, mainly because of limitations of the dataset. Regarding the event study, the predictive power of events appears to be vague, although some events, for example, campaigns with hashtags, are probably more triggering than others. However, it is puzzling that the increased activity after an event is

not really related to it and thus more research is required preferably with the incorporation of behavioral science, to draw a concrete conclusion.

Another research, which focused on sentiment analysis but was made in a completely different marketing field is presented in [11]. In this case, the gathered data were about 11 car models in Netherlands. The gathering process lasted for about 4 years and obtained information from Twitter, Facebook, LinkedIn, YouTube, Google+, Hyves, Instagram and Pinterest, adding up to a total number of 502,681 posts. The analysis showed that while the number of posts related to a car model, as well as the search volume could be indicators of its future sales, sentiment analysis did not appear to have a positive correlation and therefore could not be characterized as a reliable predictive factor. This is something that might question the predictive power of social media, however, it is necessary to consider the limitations of the study, which was conducted in the limits of the Dutch market and attempted to forecast car sales for only 11 models.

Konstantopoulos L. [8] worked on a software capable of automatically analyzing an Instagram profile, given its basic parameters as input, such as posts, comments, mentions and followers. The analysis can be broken down into 4 parts. In the first part, a report that contains main information and metrics about the profile is generated. Secondly, the set of followers is analyzed and each one is classified as either real or fake. Next, the posts of the profile are labeled based on their context and, finally, the reactions of the audience to the posts are captured with sentiment analysis of the responses. The produced software was tested on four different profiles and it was successful, since it met the requirements that had been specified. Such a tool is extremely useful in the cases of influencer profiles, which are followed by a large audience and can have a great impact on many people. Companies spend large amounts for advertising through these profiles, so estimating the trustworthiness and actual influence of each influencer, could help them choose the individuals that are suitable for promoting their products or services.

In his dissertation, Touparis F. [12] takes on the challenging task of predicting stock movement about a company, Apple in particular, using sentiment analysis on posts that had been extracted from Twitter. The research is based on the notion that traders' opinions can reveal their willingness and as a result have some predictive power over the movement of stocks. Besides, companies are interested in receiving feedback and understanding people's feelings about them and their products, so as to determine their future actions. Concerning the methodology, Twitter's Search API, which is actually a

part of the REST API, was utilized to download an amount of approximately 3,00,000 tweets that were related to Apple, with specific geographical and language parameters. Afterwards, each tweet was classified as positive, negative or neutral, using the Naïve Bayes classifier. In order to evaluate the code that was developed for the forecasting, the comparison of the predicted and actual values was made on a daily basis, so the prediction of the stock movement for each day was based on the tweets collected in the previous 24-hour period. The accuracy that was achieved was 78%, which is an impressive score considering the difficulties of producing forecasts in the area of the stock market.

Another study that used Twitter to obtain data can be found in [10]. The purpose of the study was to make predictions about the, forthcoming at that time, 2016 presidential elections in USA. The first main objective was to retrieve data from Twitter and the task was performed by exploiting the REST API, while organizing the timing of the data collection according to important dates, such as when discussions or debates would occur. The second goal was to make a sentiment analysis on the 277,509 tweets that were gathered in total. For this purpose, each tweet was given a polarity and subjectivity score, depending on being negative, positive or neutral and expressing a subjective or objective statement. The procedures used were the machine learning approach and the Naïve Bayes classification method. Oikonomou L. suggests that future research could approach sentiment analysis from a different standpoint, meaning that the efficiency of other methodologies, namely the rule based and the support vector machined method, could be tested. The results of the analysis were very accurate and in fact predicted the right candidate, in contrast to the majority of the polls, which failed to forecast the outcome of the elections.

The following papers are associated with the prediction of the Billboard chart.

The first study [13] focuses on two tasks. On one hand, it tries to discover the relationship between Twitter activity regarding music and the forthcoming sales on this particular market. Additionally, it aimed at predicting the hit songs for the next Billboard chart. The researchers gathered more than 30 million tweets, searching for the keywords *nowplaying*, *np* and *itunes*, as it is presumed that these hashtags are used to indicate the song a Twitter user is currently listening to. They also collected information from the previous Billboard charts over a span of 10 weeks, which resulted in a dataset of songs, each one with its title, artist, rank and the time period it managed to stay on the chart. Alto-

gether, the retrieved information was related to 178 songs and 134 artists, although some of these were excluded during preprocessing.

The authors established 3 distinct metrics: song popularity, which was the number of tweets related to a specific song, artist popularity, which referred to the number of tweets mentioning this artist, and the number of weeks a song appeared on the Billboard chart. The forecasting targeted at generating predictions for the top 10 songs of the chart, since this was the range that achieved the highest accuracy. According to the findings of this research, using the Pearson correlation, artist popularity and number of weeks on the chart, are not strong predictors of a song's ranking. However, if song popularity is taken into account and all 3 metrics are combined, it is possible to predict the imminent success of a song on the Billboard chart quite accurately.

In [4], the authors not only tried to predict the rank of a song on the Billboard chart for the following week but went one step further and attempted to connect the predictions for many sequential weeks, in order to construct the whole path of the song on the chart. Apart from the previous chart positions for a specific song, the developed algorithm used parameters, such as mood, song genre and the artist's gender. Interestingly, these extra features decreased the accuracy of the predictions. The best result was obtained, when the input was solely the positions of the past weeks and specifically, using Ridge Regression, for the last 5 weeks, the best attempt deviated 4.47 ranks from the actual position.

Koenigstein N., Yuval S. and Zilberman N. [7] approached the Billboard chart prediction through the exploitation of peer-to-peer networks. The popularity of these services led them to perceive them as a means for providing useful information about user preferences and current trends. The data mining process was performed on the Gnutella network, which was chosen because of its high popularity, its usual reference in academic papers and its wide collection of music tracks. In total, 185,598,176 query strings, originated from the USA, were gathered during a 30-week period. Many experiments took place, using both the M5 algorithm and Quinlan's C4.5 classifier, for the prediction of the Billboard Hot 100 and the Billboard Digital Songs charts. Moreover, in some cases, except for the data retrieved from the Gnutella network, researchers also took into consideration a song's debut rank on the chart, while the predicted positions were usually either the top 10 or the top 20. The hypothesis of the authors was correct and queries used in peer-to-peer services seem to be strong predictors of a song's suc-

cess. For the Billboard Hot 100 chart, precision reached over 86%, while for the Billboard Digital Songs accuracy exceeded 89%.

Finally, [14] examines the relationship between song-related tweets and their ranking on the Billboard Hot 100. In more detail, the objective is to investigate the resemblance of the amounts of Twitter data referring to Billboard tracks to the state of the actual chart, along with the temporal offset between them, so that it can be determined that tweets have a predictive value for the chart and not vice versa. Eventually, the point is to discover how these tweets can be utilized to enhance the forecasting of future charts which is based solely on historic data and provide more accurate predictions. The authors used a ready dataset consisting of 111,260,925 tweets that included the term *#nowplaying*, gathered during the years 2014 and 2015. In addition to that, they collected data from the Billboard Hot 100 chart for the exact same time period, which corresponded to information pertaining to 886 distinct songs. The first observation was that a song remains on the chart for an average of 11.74 weeks, with a minimum of one week and may stay on the top 100 hits over the span of 58 weeks, maximum.

In order to determine the correlation of rankings, the authors calculated three different metrics for each song on the chart based on the dataset of tweets. They estimated the median number of play-counts per week, the mean play-counts per day and the total number of play-counts for a whole week. According to their results, the first metric achieved the highest correlation (0.5), which is characterized as moderate, for 481 songs or 54.29% of the dataset. The temporal relationship between tweets and charts was investigated through a cross-correlation analysis that increased the mean correlation to 0.57 compared to the value of 0.5 that was previously found. 89.23% of all the examined tracks appear to have a temporal lag in relation to the Billboard chart, while 41.09% of all tracks have a negative lag. This means that the last percentage is exploitable for providing predictions about the chart's ranking for the weeks to come. Moreover, the authors followed the same process for songs that were first noticed in Twitter data and appear on the chart at a later point (619 tracks in total). Similarly, 42.64% of them featured a negative lag and could be helpful for the forecasting of future charts.

Regarding the predictive capability of Twitter data, the investigators compared 3 predictions models: one based exclusive on Billboard chart data, one relying only on tweets and the last one combining both sources of data. In terms of the RMSE, the Twitter-based model had the worse score of 116.1, while the first one achieved an RMSE of

26.8. However, the multivariate model displayed a notable performance of 14.1, which is 48.38% lower than the RMSE of the model that utilized only the data retrieved from the chart. The conclusion is that a combination of data originating from both the Billboard chart and Twitter can reduce the error of the forecasting significantly and, thus, song-related tweets can be useful for increasing the accuracy of ranking predictions. Summing up, about 41% of the collected tweets could be used for the prediction of the Billboard chart ranking as long as they are handled properly. This means that they would not be efficient if they were used on their own, but in conjunction with the chart's data they would be able to provide satisfactory results.

4 Dissertation Objectives

The dissertation has the following main objectives:

- Acquire data from the Billboard chart, including rank, artist and song title of the top 100 songs at the current time. For this purpose, a script written in JavaScript, using the platform node.js has been developed. The algorithm extracts the aforementioned parameters from the official site and saves them in a *json* and a *csv* file.
- Collect Twitter posts concerning the top ten songs that have been gathered. Specifically, the target is to utilize the Twitter Search API and gather tweets regarding the artists and song titles that have been previously saved. The most common programming language for interacting with Twitter is Python.
- Preprocess data and bring them into a homogenous, structured format. Any duplicate or redundant information should also be removed.
- Perform sentiment analysis on the tweets. This involves categorizing each post as positive, negative or neutral.
- Assess the contribution of the features, extracted from the Billboard chart and the collected posts, to the arrangement of the chart for the week to come.
- Attempt to predict the top N songs for the following week and determine the efficiency of the process. This requires the usage of a classifier to generate these kinds of predictions. The optimal number of N, the highest performing classification algorithm and the best combination of features can only be determined after performing a lot of testing and contrast the results.

4.1 Research Questions

RQ1. What is the correlation between the number of airplays of a track, as it is captured by Twitter posts with the *#nowplaying* keyword, compared to the number of play-counts for the other songs, and the future ranking of this song on the chart?

RQ2. How is a song's performance on the Billboard Hot 100 related to the total number of play-counts its artist gets on Twitter, compared to the play-counts of the artists of the other songs?

RQ3. How is a song's performance on the chart related to the total number of mentions its artist gets on Twitter, compared to the mentions of the artists of the other songs?

RQ4. What is the connection between an artist's reception, expressed in emotionally charged tweets, measured against the popularity of the artists of the other songs, and the performance of his track on the chart?

RQ5. How decisive are Twitter-based features in the formation of the chart for the following week and to what extent can they be used in conjunction with Billboard data to generate predictions about song ranking?

The questions above are based on the assumption that the set of Twitter posts can provide a representative sample of the song airplays that take place across the USA and the public's opinion towards an artist. It is speculated that the more attention a song gathers, the more likely it is for people to listen to it and possibly end up purchasing it in a physical or digital form, further adding up to its total popularity. Similarly, the whole publicity an artist gets and the image she presents, taking into account both her career and her personal life, will probably urge the public to check the pieces of her work and increase their commercial success.

5 Implementation

This chapter includes the experimental part of the dissertation and describes the technologies and the methodology that was followed. Concisely, each subtask of the process and the respective tools are mentioned below:

- The Search API was utilized for the interaction with Twitter.
- The scripts for interacting with the API, searching for tweets, storing the posts and performing sentiment analysis were written in Python.
- Database management tasks, including storing Twitter data and retrieving them, were implemented using the SQL Server software provided by Microsoft.
- All useful information extracted from the database records was processed through Excel and then given as input to the Weka suite, in order to produce statistical results.

5.1 Search API Queries

Artist search queries include the names of artists joined with the OR operator. If a name consists of more than one words, they are placed in parenthesis, so as to make sure that the tweet will include all of them, in any order. Additionally, the query consists of all artist names with the hashtag symbol appended in front of them. In the case of a name with multiple words, they are joined together by removing the spaces between them (Figure 6).

```
(Murda Beatz) OR (Kanye West) OR (Travis Scott) OR (Bad Bunny) OR (Post Malone) OR Drake OR Eminem OR (Nicki Minaj) OR (Lil Pump) OR (Maroon 5) OR (J Balvin) OR (Juice WRLD) OR (Cardi B) OR 6ix9ine OR (5 Seconds Of Summer) OR #MurdaBeatz OR #KanyeWest OR #TravisScott OR #BadBunny OR #PostMalone OR #Drake OR #Eminem OR #NickiMinaj OR #LilPump OR #Maroon5 OR #JBalvin OR #JuiceWRLD OR #CardiB OR #6ix9ine OR #5SecondsOfSummer
```

Figure 6: Example of search query for artist tweets

Regarding song searching, tweets related to specific words are collected mainly in order to make an estimation about their airplay count, either from radio stations or from individual users. These search queries incorporate the term *#nowplaying*, which is joined via the AND operator with the titles of all the songs, in parenthesis. Each individual title is placed in quotes and parenthesis, if it consists of multiple words. Quotes are necessary in this occasion to ensure that the post contains all the words in the exact same order, and they are not just spread in the text. This would lead to the collection of many irrelevant tweets and would add extra noise to the dataset. Moreover, similarly to the queries about artists, all words of each title are connected with the removal of spaces and the hashtag symbol is inserted in the beginning of each concatenated word, as shown in Figure 7.

The strings for searching for songs do not include artist names, since it would be possible to exceed the 500-character limit and much complexity would be added to the queries. Instead, artist names and titles are matched, when retrieving data from the database where all the tweets are stored. This makes sure that the posts refer to the particular songs that need to be analyzed.

```
#nowplaying AND (("Girls Like You") OR ("Lucid Dreams") OR ("Better Now") OR ("In My Feelings") OR Killshot OR ("Sicko Mode") OR ("I Like It") OR FEFE OR ("I Love It") OR Youngblood OR #GirlsLikeYou OR #LucidDreams OR #BetterNow OR #InMyFeelings OR #Killshot OR #SickoMode OR #ILikeIt OR #FEFE OR #ILoveIt OR #Youngblood)
```

Figure 7: Example of search query for song tweets

5.2 Database Configuration

As mentioned in the beginning of the chapter, database management was implemented using the SQL Server tool by Microsoft. After creating a database, two different tables were used in order to store the tweets that were captured via the Search API. The first one, named *artistTweets* was used for saving tweets referring to artists and the second one, *songTweets,* gathered all tweets related to songs.

The columns described here are common in both database tables:

- **id (int):** This is the primary key of the table which increments automatically with the insertion of each new record.

- **tweed_id [nvarchar(20)]:** This field is the unique identifier for each post. The *Tweet* object has two id attributes. The first one, named *id* is an integer number greater than 53 bits, so it requires a 64 bit integer field to be saved. Since, this attribute may be difficult to handle in some programming languages, it is preferable to use the *id_str* attribute, which is basically the same number represented in a string format. As it has been previously mentioned, more recent tweets will have a greater id value.
- **text [nvarchar(380)]:** This is the text of the tweet as it is extracted from the homonymous attribute of the tweet object. The UTF-8 encoded text has a limit of 280 characters as defined by Twitter. However, the posts captured by the API may overcome the limit imposed, because of the extra characters that are appended when replying or retweeting a post. For this reason, the text column has a character limit greater than 280.
- **keywords [nvarchar(500)]:** This is the "*q*" parameter or the search query of the Search API. The column has a 500-character limit in compliance with the API's parameter.
- **retweets (int):** This is where the *retweet_count* attribute of the tweet object is stored. Obviously, it refers to the number of retweets for each post.
- **location [nvarchar(5)]:** This column was initially used to store the *country_code* attribute that belongs to the *place* object. The idea was to filter out all tweets with a country code other than *US*, since the Billboard chart uses data from the USA. Nevertheless, the *place* object is nullable and most of the times it is not filled. The result is for the object to cause an error when the code for retrieving it is run. Even though, it is possible to handle such exceptions, the number of tweets that include their country code is very small and thus inadequate for the creation of a dataset. The method that was finally adopted for geographically restricting tweets was through the API's *geocode* attribute.
- **created (datetime):** Matches with the *Tweet* object's *created_at* attribute and indicates the exact date and time this post was created. The format of this field is *YYYY-MM-DD hh:mm:ss*.
- **saved (datetime):** This is the exact time the tweet is saved in the database and it actually servers as a timestamp. This value is retrieved programmatically through the Python code and has the same format as the previous field.

The columns below are only part of the *artistTweets* design and are derived from performing sentiment analysis on the *text* field:

- **compound1 [decimal(6, 4)]:** This is the sentiment score produced after performing sentiment analysis with the VADER lexicon on the *text* field of the current record.
- **sentiment1 [char(2)]:** A column that displays the polarity of the tweet based on the compound value as it was found by the VADER lexicon. If the compound is greater than or equal to 0.05, the text is considered positive, if it is less than or equal to -0.05 it is considered negative, while the tweet is characterized as neutral for the values in between. The field accepts 3 discrete values: -1, 0, 1 for negative, neutral and positive, respectively. This field was later discarded, since the *compound1* column was considered to be more precise.
- **compound2 [decimal(6, 4)]:** Similar to the *compound1* column, but with the score generated via the SentiWordNet lexicon.
- **sentiment2 [char(2)]:** Similar to the *sentiment1* column, but with the score generated via the SentiWordNet lexicon. This column was also discarded.

5.3 Storing Twitter Data

The interaction between SQL Server and Python can be conveniently achieved with the language's *pyodbc* module in just a couple of lines of code. The library can be installed using a packet management system, preferably Python's *pip*.

Interacting with the database presupposes that the SQL Server is up and running. Microsoft's SQL Management Studio is an excellent tool for managing the SQL server, and any similar infrastructure. It offers the interface to check whether the server has started and change its state if it is not currently running.

The piece of code presented in Figure 8 displays the retrieval of Twitter data using the Search API and their direct storing to the database via the communication with the SQL Server.

```
1. connection = pyodbc.connect('Trusted_Connection = yes', driver = '{SQL Serv-
   er}', server = 'ELEANA-PC', database = 'TwitterDataDB')
2. cursor=connection.cursor()

3. count = 0

4. lang='en'
```

```
5.  geocode = '39.8,-95.583068847656,2500km'

6.  for status in tweepy.Cursor(accessAPI().search, q, lang, geo-
    code).items(100000):
7.  print (status.id_str)
8.  print (status.text.encode('utf-8'))
9.  print (status.created_at)
10. count += 1
11. cursor.execute("INSERT INTO songTweets(tweet_id, created, retweets, text,
        saved, keywords) VALUES (?,?,?,?,?,?)",(str(status.id_str),
        str(status.created_at), str(status.retweet_count), status.text,
        datetime.datetime.now(), q))
12. connection.commit()

13. connection.close()
14. print count, " tweets were saved"
```

Figure 8: Retrieving and storing tweets in database

From that point on the process is quite simple: a connection has to be established with the database by passing the appropriate string to the *connect* method, which has a different format for database and Windows authentication. In line 1, the connection is made using Windows authentication and providing the required parameters, which have been set for the SQL Server. In line 2, a parameter is defined for accessing the *cursor* method, which is used to execute the string query provided to the *execute* method (line 11). The operation is committed (line 12) and eventually the connection is terminated (line 13), outside the object iteration loop.

Each commit to the database occurs multiple times for the insertion of each individual post, rather than being outside the loop and handling all the changes at once. The reason behind it is that if an unexpected event occurs that ceases the execution of the script, such as the interruption of the internet connection or the SQL server, all the tweets iterated so far would have been saved in the database. Since these unfortunate incidents occur recently, it would be a waste of resources to go through this time-consuming process, especially if the iteration involves a great number of objects, and not manage to save any data.

A very handful tool for accessing Twitter's API and manipulating its data is Python's *Tweepy* library. The iteration of objects, which can be timelines, user lists or tweets, is a very common practice when interacting with the Twitter API but requires some complicated coding. *Tweepy's Cursor* object can handle the cumbersome task of paginating through objects with very few lines of code and allow the programmer to focus on processing the retrieved data. Lines 6 to 12 show the process of iterating through Twitter

items, printing some specific information, which is the id, text and date of creation and finally inserting all the necessary attributes of each tweet in the database as a record of the *songTweets* table.

The attributes of the *status* object match the attributes of the *Tweet* object:

- **q:** It is the search query defined as it was described before.
- **lang:** It is set to "en" because the returned tweets should all be written in English.
- **geocode:** It has been set using the parameters found in [15]. The author claims that these coordinates cover a geographical area incorporating almost all states of the USA.

Method *datetime.now()* (line 11) from Python's *datetime* module provides a timestamp for each post. The *count* variable is incremented with each loop (line 10) and used to store the total number of tweets that were saved in the database and does not serve any other purpose apart from notifying the programmer about the results of the procedure (line 14).

5.4 Sentiment Analysis

Overall, the VADER lexicon is packed with a useful set of scripts that make it user-friendly for any developer who has basic programming skills and is acquainted with Python. One can also find efficient documentation online. The only minor difficulty encountered was that since its migration to Python 3, there was a compatibility issue with version 2.7, so file *vaderSentiment.py* had to be modified by adding the line "*from io import open*" at the top of it.

Comparing to VADER, sentiment analysis with SentiWordNet is not that easy to implement, since the downloadable package from the official website [18] includes only the lexicon without any ready-to-use code. Therefore, it is up to the programmer to develop her own code to manipulate the lexicon and making the appropriate configurations to get the result she needs. The task is rather cumbersome because each word needs to be treated separately. It needs to be tagged based on the part of speech they represent, which can be *noun, verb, adjective, adjective satellite* (a subcategory of adjective according to WordNet) and *adverb*, and also each term has to be assigned with a score indicating its usage frequency.

For that reason, a handy script was utilized taken from [16], which features a class with a set of methods for calculating different sentiment parameters for a given word or phrase. Most importantly there is a ready procedure for estimating the total sentiment score of a sentence and this is the metric of interest that was used to make comparisons with the respective result of the VADER sentiment analysis. The scoring method takes into account the following negation words: *not*, *n ’t*, *less*, *no*, *never*, *nothing*, *nowhere*, *hardly*, *barely*, *scarcely*, *nobody*, *none*. Furthermore, there are 3 options for the generation of the sum of scores: average, geometric and harmonic.

In order to use this module, the file containing the class had to be placed in a folder named *__init__.py* that was created in the same location as the script that was going to call it. This would allow Python to recognize and treat the file as a module. The script that implements sentiment analysis should also import the *nltk* module.

Before performing sentiment analysis on the actual tweets, the two lexicons were briefly tested. Tables 2 and 3 depict the compound value or sentiment score after analyzing the same sentences (the name of the lexicon is different in each group of sentences, but this does not modify the results) with the VADER and SentiWordNet lexicons, respectively.

Table 2: The results of sentiment analysis with the VADER lexicon

Sentence	Sentiment Score
VADER is smart, handsome, and funny.	0.8316
VADER is smart, handsome, and funny!	0.8439
VADER is very smart, handsome, and funny.	0.8545
VADER is VERY SMART, handsome, and FUNNY.	0.9227
VADER is VERY SMART, handsome, and FUNNY.	0.9342
VADER is VERY SMART, handsome, and FUNNY!!!	0.9469
VADER is not smart, handsome, nor funny.	-0.7424
The book was good.	0.4404
At least it isn't a horrible book.	0.431
The book was only kind of good.	0.3832
The plot was good, but the characters are uncompelling and the dialog is not great.	-0.7042
Today SUX!	-0.5461
Today only kinda sux! But I'll get by, lol	0.5249
Make sure you :) or :D today!	0.8633
Catch utf-8 emoji such as such as 💕 and 💋 and 😁	0.7003

Not bad at all	0.431

Table 3: The results of sentiment analysis with the SentiWordNet lexicon

Sentence	Sentiment Score
SentiWordNet is smart, handsome, and funny.	0.220703125
SentiWordNet is smart, handsome, and funny!	0.220703125
SentiWordNet is very smart, handsome, and funny.	0.220703125
SentiWordNet is VERY SMART, handsome, and FUNNY.	0.03125
SentiWordNet is VERY SMART, handsome, and FUNNY!!!	0.03125
SentiWordNet is VERY SMART, uber handsome, and FRIGGIN FUNNY!!!	0.0208333333333
SentiWordNet is not smart, handsome, nor funny.	-0.197265625
The book was good.	0.304059565067
At least it isn't a horrible book.	0.078125
The book was only kind of good.	0.202706376712
The plot was good, but the characters are uncompelling and the dialog is not great.	0.100010414918
Today SUX!	0.03125
Today only kinda sux! But I'll get by, lol	0.0125
Make sure you :) or :D today!	0.093865234375
Catch utf-8 emoji such as such as 💕 and 💋 and 😁	0.114580643674
Not bad at all	0.1875

VADER seems indeed to be more efficient for capturing the sentiment of social media posts. SentiWordNet lacked the ability to identify intensity factors, such as capitalization, exclamation marks, use of multiple degree adverbs, as well as slang terms and contractions, so the final score was not adjusted properly. Of course, these example sentences were provided by VADER's official documentation to demonstrate its effectiveness, and SentiWordNet could have generated much more accurate results in another set of texts. So, it was decided to also experiment with a lexicon that is not tailored to social media text in the implementation of tweet sentiment analysis.

Figure 9 depicts the script that implements sentiment analysis using the VADER lexicon on artist-related tweets currently stored in the database.

```
1.  import pyodbc
2.  from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
```

```
3.  analyzer = SentimentIntensityAnalyzer()

4.  def compound_polarity(text):
5.  compound = analyzer.polarity_scores(text)['compound']
6.  if compound >= 0.05:
7.  return '1'
8.  elif compound <= -0.05:
9.  return '-1'
10. else:
11. return '0'

12. cnxn = pyodbc.connect('Trusted_Connection=yes', driver = '{SQL Server}',server =
    'ELEANA-PC', database = 'TwitterDataDB')
13. cursor=cnxn.cursor()

14. cursor.execute("SELECT text FROM artistTweets WHERE sentiment is NULL")
15. tweets = cursor.fetchall()

16. count = 0

17. for tweet in tweets:
18. tweet_text = tweet[0].encode("utf-8")
19. cursor.execute("""UPDATE artistTweets SET sentiment = (?), compound = (?) WHERE text =
        (?)""", (compound_polarity(tweet_text), analyz-
        er.polarity_scores(tweet_text)['compound'], tweet[0]))
20. count += 1
21. print count
22. cnxn.commit()

23. cnxn.close

24. print count, " tweets were analyzed"
```

Figure 9: Code for analyzing tweets with the VADER lexicon

The connection to the database and the execution of queries is implemented in a similar way it was described in section 5.3. However, in this script, there is a need for an extra query to fetch all the tweets that have not been analyzed yet (line 14). This implemented with the *fetchall* method (line 15). Next, there is an iteration over the retrieved tuples and for each one, the *text* column is extracted (line 18) and analyzed. The table is updated with the *compound* column receiving the sentiment score through the *compound* dimension of the *polarity_scores* method with the extracted tweet text as a parameter. If the *compound* column was not specified, the method would also return all other metrics, meaning the positive, negative and neutral scores (line 19). In order to set the *sentiment* column, a new method has been defined (lines 4 to 11), which takes a chunk of text as a parameter and returns its semantic orientation (-1 for negative, 0 for neutral and 1 for positive) based on the common thresholds of the compound value.

The script that implement sentiment analysis with the SentiWordNet lexicon is presented in Figure 10. Similarly to the code that uses the VADER lexicon, an analyzer object

of the aforementioned class is created. The parameters that should be given are the location of the SentiWordNet lexicon and the type of weighting for the calculation for the sum of scores. From that point, the process is almost identical to the previous script; the score is estimated by calling the *score* method through the *analyzer* object and the desired text set as parameter.

```
from sentiment import SentimentAnalysis
import nltk

analyzer = SentimentAnaly-
sis(filename='SentiWordNet_3.0.0_20130122.txt',weighting='geometric')
```

Figure 10: Code for analyzing tweets with the SentiWordNet lexicon

5.5 Mining Information from the Database

In terms of preprocessing, tweet replication is treated with the removal of all records that have the same *tweet_id* and leaving only one copy of this tweet. Any records with a black *text* field are also redundant and therefore are deleted. Figure 11 shows the script for preprocessing the *artistTweets* table.

```
DELETE
FROM artistTweets
WHERE text = ''

DELETE a
FROM artistTweets a
WHERE id < (SELECT MAX(id) FROM artistTweets b WHERE a.tweet_id=b.tweet_id GROUP BY
tweet_id HAVING COUNT(*) > 1)
```

Figure 11: Preprocessing tweets

Twitter typically does not allow to tweet a blank text and all tweets that are captured by the Search API are retrieved using specific keywords which should be part of a post's text. Discarding posts with an empty string is just an add form of protection against any unexpected behavior.

Duplicate tweets with a different *tweet_id* field, but identical text, were not removed because they were perceived to add information and depict extra attention towards a topic. Moreover, especially in the case of songs that are gathered using the *#nowplaying* term, the content of the posts is mainly fixed as it usually includes the artist and title of the track, sometimes followed by a link.

Twitter's Search API captures all tweets, including retweeted posts. These can be sometimes recognized from their textual content, which begins with the pattern "*RT @[username]*", where *[username]* refers to the user being quoted. Never the less, this is not an official feature and it is not necessarily inserted in every retweet, so there is no accurate way to determine that a post captured via the API is definitely a retweet.

In order to produce a tradeoff between the number of occurrences for each distinct text value (that is the number of posts that have the exact same text) and the number of retweets, the greatest value of the two is chosen. The idea behind this is to get the biggest sample possible and exclude the difference between the two parameters, which is probably tweets that coincide.

For retrieving song tweets from the database, the song title is used with and without spaces between words. For most tracks, the artist must also appear in the text. This is a measure to filter out any records with different songs that happen to have the same title or phrases which are absolutely irrelevant to the tracks under examination. The rules that generally apply for artist names are the following:

- The text field must include all words of an artist's name in the correct order, whether this is a band or an individual. Group names usually consist of a phrase and as a result when a group name is mentioned all words should be in a predetermined order. Some artists may also use nicknames, which constitute a specific sequence of words.
- Even when real names are used it is quite unusual that an artist is referred by his name in an unordered manner (for example, "*West Kanye*" instead of "*Kanye West*").
- Alternately, the text field must include all words of an artist's name in the correct order, with no spaces between them.

- If a name has a word that is a single letter or number (for instance, "*J Balvin*" or "*Maroon 5*"), it can be omitted, as the rest of the name along with the song title are considered sufficient to zero in the correct records.

Figures 12 and 13 show two examples of SQL queries that comply with the rules above.

```
SELECT SUM(retweetsSumTune) AS 'Girls Like You' FROM
(SELECT CASE WHEN SUM(retweets) > COUNT(text) THEN SUM(retweets) ELSE
COUNT(text) END
AS retweetsSumTune
FROM songTweets
WHERE (text LIKE '%girls like you%' OR text LIKE '%girlslikeyou%')
AND (text LIKE '%maroon%' OR text LIKE '%cardi%')
AND CONVERT(VARCHAR(25), created, 126) >= '2018-10-02%'
AND CONVERT(VARCHAR(25), created, 126) < '2018-10-09%'
GROUP BY text) AS temp
```

Figure 12: Example 1 of an SQL query

```
SELECT SUM(retweetsSumTune) AS 'Better Now' FROM
(SELECT CASE WHEN SUM(retweets) > COUNT(text) THEN SUM(retweets) ELSE
COUNT(text) END
AS retweetsSumTune
FROM songTweets
WHERE (text LIKE '%better now%' OR text LIKE '%betternow%')
AND (text LIKE '%post malone%' OR text LIKE '%postmalone%')
AND CONVERT(VARCHAR(25), created, 126) >= '2018-10-02%'
AND CONVERT(VARCHAR(25), created, 126) < '2018-10-09%'
GROUP BY text) AS temp
```

Figure 13: Example 2 of an SQL query

It should be noted that these rules are not strict because each song title and artist name is different, and the same list of conventions is not essential for all of them. The most efficient approach to retrieve related tweets is to experiment using a trial and error procedure.

5.6 Attribute Description

In order to use Weka, all the information extracted from the Billboard chart and the gathered tweets should be organized in attributes and formatted properly to create an

arff file. The file consisted of 80 instances in total and no missing values. Each instance described a specific track for one week, during which the track was at one of the top 10 positions of the chart. So, the data consist of 10 instances per week.

Since the hypothesis is that most of the attributes are positively correlated to the success of a song, the song will get to a higher position of the chart when they increase. The Billboard chart, just like most charts, ranks the songs in ascending order, starting with the most popular ones. So, mainly for harmonization with the rest of the attributes, as it was seen in literature [13], the position of each track is inverted by subtracting it from 101. For instance, number 1 song will be in position 100, while number 10 in 91. This applies to *previous-position* and *position* attributes.

Each instance of the data represents a different song for a single week and it has to be compared to all other instances, which are the remaining 9 tracks from the top 10 positions of the chart for the same week and 10 more tracks for every week that has been monitored. In the main, data gathering took place multiple times per day in a scheduled manner, but it makes sense that the total number of tweets captured on a weekly basis would differ each time because it was affected by a variety of reasons, despite of requesting a consistent number of items programmatically. Therefore, measuring every instance against all the others required some sort of transformation on the specimen of tweets. For each week, the total number of posts for the top 10 songs was summed and the percentage of each track in comparison to the other 9 hits was calculated. Attributes *song-play-count*, *artist-play-count* and *artist-tweets* were tuned as described above.

The attributes are described thoroughly below:

- **Previous-position:** The position that the song was at one week before.
- **Chart-weeks:** The number of weeks that the song has been in one of the 100 positions of the chart since its first entry. This value is set to 1 as soon as the song makes its chart debut.
- **Top-weeks:** The number of weeks that the song has been at position number 1. This value is set to 1 as soon as the song reaches the first position.
- **Song-play-count:** The tuned number of tweets, including retweets, which are related to the specific song and include the *#nowplaying* keyword. Presumably these tweets depict the number of times it was played.

- **Artist-play-count:** The tuned number of tweets, including retweets, which are related to the artist who sung this particular track, along with the *#nowplaying* keyword, but do not necessarily refer to this specific song. This is an estimation of the total number of play-counts for the artist, irrelevant of the song being played. If a song belongs to more than one artists, the artist with the maximum number of tweets associated with her is chosen to represent the song.
- **Artist-tweets:** The tuned number of tweets, including retweets, which are related to the artist who sung this particular track. If a song belongs to more than one artist, the artist with the maximum number of tweets associated with her is chosen to represent the song.
- **Artist-sentiment-analysis-1:** The average value of the compound, in other words the score, which is produced by the VADER lexicon through the applications of sentiment analysis on all tweets associated with the artist of this song. If there are many artists, the choice is made similarly to the previous cases.
- **Artist-sentiment-analysis-2:** This attribute is calculated like *artist-sentiment-analysis-1* and the only difference is the lexicon exploited for the score generation, which is SentiWordNet.
- **Position:** The position of the song for that week. This is the attribute that needs to be predicted.

Table 4 shows the type and value restrictions for each attribute.

Table 4: Attribute values and restrictions

Attribute	Type	Range
Previous-position	Integer	0 - 100
Chart-weeks	Integer	>= 1
Top-weeks	Integer	>= 0
Song-play-count	Real	0 - 100
Artist-play-count	Real	0 - 100
Artist-tweets	Real	0 - 100
Artist-sentiment-analysis-1	Real	-1 - 1
Artist-sentiment-analysis-2	Real	-1 - 1
Position	Integer	0 - 100

5.7 Regression and Classification Analysis

Pearson's correlation coefficient can be easily calculated for all attributes through Weka. After loading the *arff* file from the *Preprocess* tab, the "*Select attributes"* tab becomes active, which provides a variety of options for attribute evaluation. The estimation of the values found in Table 5 was implemented using the *CorrelationAttributeEval* evaluator and *Ranker* with the default options set by Weka, as the search method. The attributes are presented in descending order according to Pearson's correlation value. The value for each attribute was further verified using Excel's *CORREL* function.

Table 5: Pearson's correlation for each attribute

Attribute	Pearson's Correlation Coefficient
Chart weeks	0.3772
Previous position	0.3704
Song playcount	0.2917
Artist playcount	0.1467
Artist sentiment analysis 2	0.134
Artist sentiment analysis 1	0.1331
Top weeks	0.123
Artist tweets	0.0134

Based on the output the attributes could be divided in 3 categories in relation to their correlation (r) with the predicted class (position):

- **Moderate correlation attributes (0.3 < r < 0.7):** These are *chart-weeks, song – play-count* and *previous-position*. The number of weeks a track has been in the chart appears to be the most determinant factor for its future ranking. *song – play-count* is included here, although its correlation is slightly below 0.3, because it is very close to the threshold and at the same time differs a lot from the attributes with lower values.
- **Weak correlation attributes (0 < r < 0.3):** *Artist-play-count, artist-sentiment-analysis-2, artist-sentiment-analysis-1* and *top-weeks* are the attributes included in this category. Unexpectedly, the artist sentiment analysis implemented by the

SentiWordNet lexicon has a greater correlation than the analysis with the VADER tool, although the difference between the two of them in insignificant.

- **Unrelated attributes (r~0):** *Artist-tweets* is the only attribute falling in this category with the value of the correlation coefficient extremely close to 0. Notably, the number of tweets an artist gets does not seem to have much of an effect for a track's position in the short term.

5.7.1 Regression Analysis

Table 6 presents the best scoring algorithms, with the smallest mean absolute error and a value smaller than 100% for relative absolute error and root relative squared error, tested with a 10-fold cross-validation.

The Support Vector Regression algorithm has the smallest mean absolute error (4.0515) and Random Forest has the lowest root mean square error (8.8117). Apart from that, Bagging also achieved decent results with a root mean square error of 8.961 and the highest correlation coefficient (0.5697).

Table 6: Results of regression analysis

	Correlation coefficient	Mean absolute error	Root mean squared error	Relative absolute error	Root relative squared error
Gaussian Processes	0.3463	5.2324	10.0787	87.5252 %	93.2138 %
Linear Regression	0.4391	5.8252	9.7087	97.4413 %	89.7916 %
Support Vector Regression	0.4458	4.0515	10.0453	67.7718 %	92.9047 %
LWL	0.4677	5.5322	9.8914	92.5404 %	91.4815 %
Bagging	0.5482	4.6146	8.961	77.1915 %	82.8761 %
Randomizable Filtered Classifier	0.4125	4.875	9.9649	81.5473 %	92.1613 %
Random Subspace	0.4747	4.9849	9.5538	83.3862 %	88.359 %
M5 Rules	0.4737	4.7111	9.6547	78.8049 %	89.2922 %
Decision Stump	0.4577	5.3391	9.6458	89.3104 %	89.2097 %
M5P	0.5266	4.4493	9.1883	74.4262 %	84.9786 %
Random Forest	0.5697	4.2543	8.8117	71.1637 %	81.496 %

5.7.2 Hit Prediction

Hit prediction refers to forecasting whether a song is going to be within a specific range of positions or not. This is a classification problem, as each instance should be matched to one of the two available classes. For this purpose, a new attribute should be created through Weka's preprocessing functionality.

First of all, the *AddExpression* filter, under the *unsupervised* and *attribute* sections, is utilized to add a new attribute deriving from the ones already existing. In this occasion the goal is to split the position values in two ranges: the top 10 positions and the lower 90 positions. So, the following mathematical expression is filled in the "expression" field:

ifelse ((A9 >= 91), 1, 0)

A9 refers to attribute number 9, which is the position. For each instance, if the value of the position attribute is equal to or greater than 91, it will be classified in class 1, otherwise it will be assigned to class 0. As it was previously mentioned, 91 corresponds to position number 10 in the actual chart. The application of this filter will create an extra attributed named *hit*.

Nevertheless, Weka cannot yet recognize that this should be treated as a nominal attribute. For that reason, another filter should be used. The *NumericToNominal* filter is in the same section as the previous one. The attribute to be transformed is chosen by setting the *attributeIndices* option to 10. At this point, the position attribute can be removed, and the dataset is finally ready for use.

After experimenting with all available algorithms using 10-fold cross-validation, only classifiers with accuracy greater than 80% are displayed in Table 7. J48 and PART seem to be the classifiers with the best performance in terms of both accuracy and F1 scores.

Table 7: Results of classification for the prediction of the top 10 hits

		Top 10 Hits			**Non-hits**		
Algorithm	**Accuracy**	**Precision**	**Recall**	**F1 measure**	**Precision**	**Recall**	**F1 measure**
Logistic	81.25%	0.843	0.937	0.887	0.600	0.353	0.444

SGD	80%	0.797	1.000	0.887	1.000	0.059	0.111
Simple Logistic	80%	0.822	0.952	0.882	0.571	0.235	0.333
AdaBoost M1	82.5%	0.845	0.952	0.896	0.667	0.353	0.462
Bagging	81.25%	0.816	0.984	0.892	0.750	0.176	0.286
Iterative Classifier Optimizer	81.25%	0.816	0.984	0.892	0.750	0.176	0.286
Multi-Class Classifier	81.25%	0.843	0.937	0.887	0.600	0.353	0.444
JRip	82.5%	0.866	0.921	0.892	0.615	0.471	0.533
PART	85%	0.892	0.921	0.906	0.667	0.588	0.625
J48	85%	0.870	0.952	0.909	0.727	0.471	0.571
Random Forest	80%	0.831	0.937	0.881	0.556	0.294	0.385
Random Tree	81.25%	0.843	0.937	0.887	0.600	0.353	0.444

Similarly, the same process was followed to make predictions for different ranges of hit songs. The only difference in the procedure was in the step of defining a new attribute with the “AddExpression” filter. The “expression” field was set to:

ifelse ((A9 >= 96), 1, 0)

for predicting the top 5 songs and:

ifelse ((A9 >= 81), 1, 0)

for the case of the top 20 tracks. Essentially, only the split point would change.

The best scoring algorithms in the case of top 5 hits are Filtered Classifier and Decision Table, achieving 90% accuracy and the same value for the metrics of precision, recall and F1. F1-scores were 0.875 for hits and 0.917 for non-hits. Table 8 depicts the results of the classification for all algorithms with accuracy greater than 85%.

Table 8: Results of classification for the prediction of the top 5 hits

		Top 5 Hits			**Non-hits**		
Algorithm	**Accuracy**	**Precision**	**Recall**	**F1 measure**	**Precision**	**Recall**	**F1 measure**

Logistic	86.25%	0.824	0.848	0.836	0.891	0.872	0.882
SMO	88.75%	0.833	0.909	0.870	0.932	0.872	0.901
LWL	88.75%	0.875	0.848	0.862	0.896	0.915	0.905
AdaBoost M1	87.5%	0.829	0.879	0.853	0.911	0.872	0.891
Bagging	87.5%	0.848	0.848	0.848	0.894	0.894	0.894
Classification Via Regression	86.25%	0.806	0.879	0.841	0.909	0.851	0.879
Filtered Classifier	90%	0.903	0.848	0.875	0.898	0.936	0.917
Iterative Classifier Optimizer	86.25%	0.867	0.788	0.825	0.860	0.915	0.887
Multi-Class Classifier	86.25%	0.824	0.848	0.836	0.891	0.872	0.882
Decision Table	90%	0.903	0.848	0.875	0.898	0.936	0.917
JRip	87.5%	0.871	0.818	0.844	0.878	0.915	0.896
OneR	88.75%	0.875	0.848	0.862	0.896	0.915	0.905
Decision Stump	88.75%	0.875	0.848	0.862	0.896	0.915	0.905

For the prediction of the top 20 hits of the following week, the classifiers that achieve high accuracy and F1 values are LMT and Simple Logistic. All the classifiers that scored more than 93% in accuracy are displayed in Table 9.

Table 9: Results of classification for the prediction of the top 20 hits

		Top 20 Hits			**Non-hits**		
Algorithm	**Accuracy**	**Precision**	**Recall**	**F1 measure**	**Precision**	**Recall**	**F1 measure**
Simple Logistic	96.25%	0.973	0.986	0.980	0.800	0.667	0.727
KStar	93.7 %	0.986	0.946	0.966	0.556	0.833	0.667
AdaBoost M1	90%	0.949	1.000	0.974	1.000	0.333	0.500
LogiBoost	93.75%	0.948	0.986	0.967	0.667	0.333	0.444
Random Committee	93.75%	0.960	0.973	0.966	0.600	0.500	0.545
PART	95%	0.961	0.986	0.973	0.750	0.500	0.600

J48	95%	0.961	0.986	0.973	0.750	0.500	0.600
LMT	96.25%	0.973	0.986	0.980	0.800	0.667	0.727
Random Forest	95%	0.961	0.986	0.973	0.750	0.500	0.600
Random Tree	93.75%	0.960	0.973	0.966	0.600	0.500	0.545
REPTree	93.75%	0.937	1.000	0.967	1.000	0.167	0.286

6 Results

This dissertation investigated the forecasting strength of social media, specializing in the prediction of the Billboard's Hot 100 positioning, based on data extracted from the chart and music-related Twitter posts referring to the artists and songs that occupy one of the top ten positions for each week. In total, more than one million tweets were gathered, and data collection lasted about 2 months, during October and November of 2018.

Twitter data appear to be quite representative when it comes to the number of total play-counts for a song. Regarding RQ1, considering the generally accepted ranges, there is a moderate correlation (0.2917) between the number of tweets that include the title of a song and the *#nowplaying* terms as a subset and the imminent success of the song on the Billboard Hot 100 chart for the following week. On the contrary, to answer RQ2, tweets that provide an estimation for an artist's total play-counts in general are not adequate to give an accurate picture of the future performance of a particular song, as the correlation coefficient was proved to be weak (0.1467).

There seems to be no relation between the publicity of an artist, either positive or negative and expressed as the total number of tweet mentions, and his ranking on the chart (RQ3), since the value of the correlation coefficient was pretty close to 0 (0.0134). Of course, the publicity game that celebrities engage themselves into in their attempt to attract the attention of media is not irrational and benefits their career, but this is presumably achieved in more indirect ways. Furthermore, there is no evidence that the whole positive attention an artist gets, represented by a value that estimates how favorable the posts related to her are, is significantly correlated to the positioning of her tracks on the Billboard chart, at least in the short term (RQ4). Findings concerning sentiment analysis cannot be generalized for other domains other than this particular chart, as many other studies have ascertained [1, 10, 12].

Answering RQ5, if features derived from tweets are combined with the chart's data, they indeed can provide results of noteworthy accuracy, but this happens only in the investigation of specific aspects of the problem.

Regarding regression analysis, the best performances were achieved by the Support Vector Regression classifier with a mean absolute error of 4.0515 and Random Forest

with a root mean square error of 8.8117. The value of the mean absolute error means that the predictions derived about the position attribute were on average about 4 ranks away from the actual values. Nevertheless, the mean square error metric is, in most cases, considered more reliable, as it penalizes outlier values. According to it, the average derivation is approximately 8 positions. The evaluation of the results depends on the type of problem that needs to be solved. For example, if the goal is to a have a depiction of the whole Billboard chart for the following week, the values could be considered useful. On the other hand, if it is desired to predict the positions for a particular range and especially if this range is limited, the top 10 hits, for instance, then this model would not be able to provide an accurate prediction. The highest correlation coefficient was 0.5697, with Random Forest algorithm, and the squared correlation coefficient was approximately 0.325, which is a low performance compared to the best result found in [13], with a squared correlation of 0.57, using the Support Vector Regression Algorithm and a 5-fold validation.

Hit prediction, implemented via classification, yielded some promising results: top 10 songs could be predicted with an accuracy of 85% utilizing the J48 and PART classifiers. The F-scores for PART 0.906 for hits and 0.625 for non-hits, while the same values for J48 were 0.909 and 0.571. In [13], with the Random Forest Classifier and a 5-fold validation, the accuracy achieved was 90% and the F-scores for hit and non-hits were 0.901 and 0.899, respectively.

Experimenting with different ranges for hits, specifically song in the top 5 and top 20 positions, the scores were higher than the top 10 range. Filtered Classifier and Decision Table algorithms achieved an accuracy of 90% and F-scores of 0.875 for the top 5 hits and 0.917 for non-hits. For the prediction of the songs in the 20 highest positions Simple Logistic and LMT achieved an accuracy of 96.25%. Specifically, the F-scores for hits and non-hits were 0.980 and 0.727, respectively. In [13], the accuracy for the same range (with the Random Forest Classifier and a 5-fold validation) was 88.2% and the F-score for hit songs was 0.885. However, they achieved a better F-score for non-hits which was 0.879.

6.1 Limitations and Future Research

The research gathers and examines Twitter and Billboard chart data for hit songs, which are at the top 10 positions for the current week. Tracks at lower ranks are not taken into account and thus it is impossible to predict their way up. So, it is advisable for future

research to investigate the chart at a larger scale and consider all songs on the chart, and, even better, create a more comprehensive dataset consisting of the chart's data for a period of one or more years, like the ones used in [13] and [14].

From a technical point of view, researchers are encouraged to try to capture tweets incorporating the *#nowplaying* term in general, without targeting specific songs or artists. Finding Twitter data referring to a set of 100 songs for each week would require a complicated methodology if the Search API is used, due to the search query limitations that were discussed in section 2.1.1. On the other hand, the more generalized approach would demand a tremendous amount of space for storing the data and would capture a lot of irrelevant tweets that would need to be excluded at a later stage. Thus, there should also be extra effort put on preprocessing. Nevertheless, collecting tweets indiscriminately ought to give a better insight regarding airplay trends and could probably aid the discovery of songs that are not at the chart the given moment, but will be introduced in the weeks to come.

Undoubtedly, an exhaustive study of the optimized way to predict the Billboard chart should rely on a model that considers all the parameters that affect song ranking, such as streaming activity and song sales. During the data gathering and experimenting stage, it was observed that many tracks made their debut into the chart by occupying one of the first 10 positions, consequently they could not have been monitored even if all 100 songs were accounted for. They also tended to follow a specific pattern: usually they dropped significantly after their successful entrance into the chart and their play-counts seemed to be quite low at that time. It is, therefore, speculated that their steep rise to the top is basically owing to music sales and streaming data. In order to get a better idea about this phenomenon, research should be extended beyond the limits of the Billboard chart and the airplay counts and consider the other dimensions that contribute to the formation of the chart each week.

With regards to sentiment analysis, the degree to which tweets constitute a trustworthy sample for determining the positive or negative disposition of the population towards an artist is still vague. So, there may be a relationship, but it may not be directly visible. This fact raises new questions about the predictive strength and limits of sentiment analysis in social media and is something worth investigating in the future. It is suggested that academics explore the impact of sentiment analysis in a more long-term context. This essentially means, monitoring the sentiment orientation for an artist and track the

performance of her songs for many weeks, possibly taking into account some time lag, as the public's response to a likeable or unlikeable person may be revealed gradually.

Different angles can also be investigated, for instance, the relationship between a positive or negative bias towards an artist and the number of tracks belonging to him that are on the chart at the present time or the rate at which these tracks ascend and descend in the chart. Researchers could, additionally, consider improving the mechanism of sentiment analysis and attempt to enhance existing lexicons in order to make them more suitable for music-related data. For instance, some terms referring to music, like *radio*, *album*, *concert*, *listen*, etc., which would otherwise be characterized as neutral, could infuse a positive tone into a tweet, when the name of an artist is included in the text.

Other recommendations for extending the research on this topic, is to capture more features from Twitter, such as likes and comments for each tweet. However, this functionality is not inherently supported by Twitter APIs so it would require more advanced methods of data scraping. Eventually, other social media networks, like Facebook and Instagram, could be explored, although this is also a challenging task, especially after the new regulations enacted in 2018 about digital data protection.

7 Conclusions

This dissertation gathered data from two different sources: titles, artist names and rankings from the Billboard Hot 100 chart and data related to the songs occupying the top 10 positions from Twitter, in order to test how the integration of these components could be used for forecasting future Billboard charts. The findings suggest that there is a moderate correlation of 0.2917, between the number of mentions of a song and its performance on the chart. No significant relationship was observed between the total attention an artist gets on Twitter, even if the relevant tweets were emotionally charged, and the success of their tracks.

Regarding regression analysis, the best score that was produced included a mean square error of 4.0515 and a root mean square error of 8.8117. Both metrics show the average number of positions between the actual and predicted values and their evaluation depends on the range of positions that is desired to be predicted. With the best scoring algorithm, hit prediction could be achieved with an accuracy of 80% and F-scores of 0.881 and 0.385 for hits and non-hits, for the 10 highest ranks. Similarly, for the top 5 hits those values were 90%, 0.917 and 0.875, respectively. Finally, for the top 20 songs, the results were competitive to previous researches achieving a maximum accuracy of 96.25% and F-scores 0.980 for hits and 0.727 for non-hits.

Overall, the study has shown that mining Twitter data, extracting specific information and handling them properly can provide some useful conclusions regarding the formation of the next Billboard chart, although it is not able to predict it perfectly.

References

[1] Asur S., Huberman B., "Predicting the Future With Social Media", Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on, vol. 1, 2010, pp. 492–499

[2] Baccianella S., Esuli A., Sebastiani F., "SENTIWORDNET 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining"

[3] Boldt L. C., Vinayagamoorthy V., et al., "Forecasting Nike's Sales using Facebook Data", 2016 IEEE International Conference on Big Data (Big Data)

[4] Cibils C., Meza Z., Ramel G., "Predicting a Song's Path through the Billboard Hot 100", 2015

[5] Esuli A., Sebastiani F., "SENTIWORDNET: A Publicly Available Lexical Resource for Opinion Mining"

[6] Hutto C.J., Gilbert E., "VADER: A Parisomonious Rule-based Model for Sentiment Analysis of Social Media Text", Proceedings of the Eighth International AAAI Conference on Weblogs and Social Media, Association for the Advancement of Artificial Intelligence, 2014

[7] Koenigstein N., Yuval S., Zilberman N., "Predicting Billboard Success Using Data-Mining in P2P Networks", 11th IEEE International Symposium on Multimedia, 2009

[8] Konstantopoulos L., "An Automated Evaluation Tool for Social Media Influencers", MSc dissertation, IHU, 2017

[9] Oikonomou L., "Predicting the USA presidential elections using Twitter Data", MSc dissertation, IHU, 2016.

[10] Oikonomou L., Tjortjis C., "A Method for Predicting the Winner of the USA Presidential Elections using Data Extracted from Twitter", 3rd IEEE SE Europe Design Automation, Computer Engineering, Computer Networks, and Social Media Conference (IEEE SEEDA-CECNSM18), 2018.

[11] Plant H., "Social media and sales: Determining the predictive power of sentiment analysis towards car sales, University of Twente", 7th IBA Bachelor Thesis Conference, July 1st, 2016, Enschede, The Netherlands

[12] Touparis F., “Predicting Stock Movement Using Social Media Analytics”, MSc dissertation, IHU, 2017.

[13] Yekyung K., Bongwon S., Kyogu L., “#nowplaying the Future Billboard: Mining Music Listening Behaviors of Twitter Users for Hit Song Prediction”, 2014

[14] Zangerle E., Pichl M., Hupfauf B., Specht G., “Can Microblogs Predict Music Charts> An Analysis of the Relationship Between #nowplaying Tweets and Music Charts”, University of Innsbruck, Austria

[15] https://github.com/agalea91/twitter_search/blob/master/twitter_search.py

[16] https://github.com/anelachan/sentimentanalysis (accessed 07/12/2018)

[17] https://github.com/cjhutto/vaderSentiment (accessed 07/12/2018)

[18] https://sentiwordnet.isti.cnr.it/ (accessed 07/12/2018)