University of Macedonia
Program of Postgraduate Studies
Department of Applied Informatics

Data analysis and mining for Twitter using R

Master Thesis
by

Athanasiadou Maria

Thessaloniki, June 2019

Data analysis and mining for Twitter using R

Athanasiadou Maria

B.Sc.in Electronic Engineering,
Technological Educational Institute of Crete, 2013

Master Thesis

Submitted as a prerequisite in fulfillment of the partial requirements for the acquisition of the postgraduate degree in Applied Informatics

Supervisor:
Dr. Koloniari Georgia

This thesis has been approved by the Examining Committee on 25/06/2019

Dr. Koloniari Georgia	Prof Dr. Evangelidis Georgios	Dr. Xinogalos Stylianos
....	....	....

Athanasiadou Maria

....

Dedications

To my memorable and beloved
Grandmother Maria(5.02.2019)

Abstract

The purpose of this thesis is to export and exploit useful information from Twitter. In particular, the interest focuses on the collection of real-time tweets, which was achieved with the capabilities that we have gives us with the highly organized and functional programming interface Twitter APIs, namely the Twitter Streaming API. Then, to access to the data of Twitter Streaming API was used the Python programming language where specific fields were extracted from the tweets with the purpose to analyze them and draw useful conclusions. Furthermore, followed a data exploration in some fields and performed text mining techniques such as text classification and text clustering implemented on common algorithms such as Naive Bayes, SVM, K-means, LDA, aiming on high-quality information from the text data using the R programming language. An experimental evaluation of the algorithms was performed in the recall, the precision, and in more evaluation metrics. The final results indicate the Naive Bayes as the best text supervised classifier and LDA as the best unsupervised classifier.

Keywords: Twitter, Twitter Streaming API, Python, R, data analysis, text classification, text clustering

Acknowledgements

The end of an important stage of my educational journey is completed. This journey was a unique experience, a huge challenge and a dream come true. The journey is completed in the best way, choosing a very interesting topic. This opportunity was given to me from my supervisor Ms. Koloniari, to whom I would like to express my most sincere thanks for her valuable advices and comments, her essential guidance, the time she dedicated and the fact that she was so patient and supportive all the way from topic selection to the finishing of this thesis. I would like to thank everybody who facilitated the completion of this thesis and of this program as a whole. In particular, I am very grateful to an amazing man, Dimitris, for being always ready to help me and to support me in my every step with incredible patience and willingness. Special thanks to Catherine for her tremendous help and constant support during the entire period of the program. Also, I would like to thank my friends, Anna, George, Maria and Olga that have been next to me all this period. Writing this thesis would not have been possible without their unconditional love, belief, and encouragement in every sense of my family. I am eternally grateful to my grandmother Maria, my mother Olympia and my sister Olympia.

Athanasiadou Maria
Thessaloniki, June 2019

Contents

List of Figures

Page

List of Tables

Page

Chapter 1 Introduction

1.1 Goal of the Thesis

Having attracted the interest of a large majority of people in recent years, the social media have become an integral part of daily life on a global level with the people choosing to have a personal digital footprint on them. Huge amounts of social data are produced by Facebook, Twitter, and social companies on the web. Thus, social media have become valuable sources for collecting information. The purpose of this thesis is to extract and exploit useful information from the social media platform, Twitter. The goal we seek is: the explaining and implementing of algorithms in text mining tasks and techniques, trying to achieve the desired results. We begin by exploring the impact on texts using the Naïve Bayes and SVM classification algorithms, and then the k-means and LDA clustering algorithms follow.

1.2 Outline of the Thesis

This thesis is structured as follows:

Social networks and Twitter (Chapter 2): The chapter initially provides a brief description of a social network and its importance. Then, the social network Twitter is being studied, where they are being discussed the historical background about its foundation, its functional environment, and necessary information associated with it.

Methodology (Chapter 3): The chapter presents the necessary procedure to be followed, with the main purpose of collecting the tweets. Initially, the steps followed to build a Twitter application are described in order to be gained access to Twitter data. Then, the codes are being described that used for the communication with the API, the collection of particular tweets and the storing of them. Furthermore, the programs and the dataset are being presented that were utilized in this thesis.

Theoretical Background (Chapter 4): The chapter describes in detail the theoretical background regarding the key components of our work. It is being presented the area of the text mining and their tasks as well as the algorithms for the implementation of them. In addition, are described the different techniques that were used in the text attribute as well as the evaluation measures that were utilized in the algorithms.

Experiments and Results (Chapter 5): The chapter presents the principal findings of the research by explaining them with the experimental approaches. The findings can be divided into two sections. The first section presents an exploratory data analysis, aiming to obtain a general view of the data, while the second section presents the text mining tasks such as text classification and text clustering aiming to extract meaningful insights. The instrumentation that was utilized in the research is the programming language R.

Conclusion and Future Work (Chapter 6): The chapter discusses the results of the experiments and presents a suggestion for future work.

References (Chapter 7): The chapter presents the references that were utilized for writing the thesis.

Chapter 2 Social networks and Twitter

The chapter initially provides a brief description of a social network and its importance. Then, the social network Twitter is being studied, where they are being discussed the historical background about its foundation, its functional environment, and necessary information associated with it.

2.1 Introduction to social networks

Technology is rapidly evolving and the Internet has produced a huge impact on how people can communicate and exchange information. The internet has reduced the high cost of traditional communication to a minimum and has allowed new forms of communication, collective action, and collaboration [Boyd, & Ellison, 2007]. Thus, one of the new forms is the development of social networks.

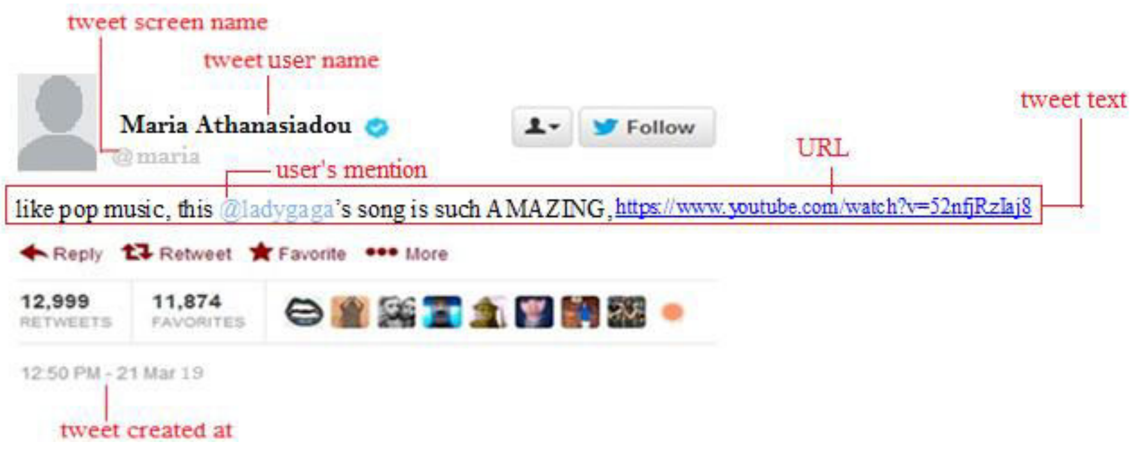
A social network is a social structure made up of individuals or organizations called 'nodes' that are linked to one or more specific types of relationships such as friendship, affinity, knowledge, common interests, financial exchanges, dislikes, sexual relations or common beliefs [Wikipedia. Social network; Wikipedia. Social networking service]. Social networks are a way of socializing both individuals and society [Burt,1995; Bourdieu, 1985; Portes,& Sensenbrenner, 1993]. A way of social networking is through social networking sites. Social networking sites are web services that allow people to create a public profile, create a list of users with who can share information, and view the connections within the system. They have any number of members, from a few tens to millions and offer people the ability to get in touch with each other based on their common interests. Internet users show a rapid increase in which social networking services play an important role and constantly gaining more users. The social networking sites are flooded by a plethora of data, also known as, social (digital) data, many of which are accompanied by metadata which includes a variety of information. These social data are very important for many researchers as they constitute a research interest in their analysis. Therefore, it is evident the need and the challenge that these data can be collected and properly managed to extract useful information. In general, social networks are now an important part of internet users as they carry a piece of their life into the digital world. Some examples of these are Twitter, Facebook, LinkedIn, Instagram, YouTube. This dissertation is based on Twitter.

2.2 The social network: Twitter

Twitter is one of the most popular online social networking and microblogging tools the main idea of which is the users to post instant messages anytime long, called tweets, with a maximum length of 140 characters. As a social networking service, it quickly gained recognition throughout the world, today it counts more than 350 million active users who publish 400 million tweets, the day [Zi,Chu et al.,2012]. Twitter is also considered as "the Internet SMS". This limited number of characters on the content of a tweet caused people to use acronyms and abbreviations to enlarge the expressibility of their message. Thus, Twitter language has generated its own specialized slang, acronyms and abbreviations. For instance, hashtag, namely words or phrases prefixed with a # symbol, can group tweets by topic, the symbol @ followed by a user in a tweet enables the direct delivery of the tweet to that user, the ″RT″ abbreviation for Retweet at the beginning of a tweet indicates that someone else's content is being re-posted [Zi,Chu et al.,2012; Twitter Help Center. FAQs about Retweets]. The retweeting is the key mechanism for information diffusion on Twitter.

As in each social network, and so on Twitter, there are relationships between the users of the network. An important feature of Twitter that separates it from other social networking sites like Facebook is that a Twitter's user relationship is directed and consists of, friend and follower. These two concepts must be explained with an example for better understanding. Twitter allows a user A, to ″follow″ updates from other users who are added as ″ friends″. A user B who is not a friend of user A but ″follows″ the updates are known as a ″follower″[Akshay, Java et al.,2007]. When user A follows user B the following relationship is unidirectional from A to B, the user B is not required to follow the user A. The relationship becomes bidirectional only when the users A and B follow each other [Zi,Chu et al.,2012].

Tweets are the essence of Twitter associated with the user's status update but in fact, there are many more metadata within them. In addition to the text content of the tweet itself, tweets are accompanied by two additional pieces of metadata that are of particular importance: entities and locations. The entities of a tweet are essentially user mentions, URLs, hashtags, and media that may be related to a tweet whilst, the places are locations in the real world that may be attached with a tweet, either the actual location in which it was authored or a reference to a place that is described in its contents [Matthew, A. Russell., 2014]. A structure of a sample of a tweet can be seen in Figure 2.1.



like pop music, this @ladygaga's song is such AMAZING, https://www.youtube.com/watch?v=52nfjRzlaj8

Figure 2.1: A structure of a tweet.

2.3 Trends and Influences on Twitter

Twitter was founded in March 2006 and launched by Jack Dorsey, Evan Williams, Biz Stone and Noah Glass in July of the same year in San Francisco, California [Wikipedia. Twitter]. The response to the new social media was phenomenal. It started with 400,000 tweets posted per quarter, that is 5,000 tweets per day, in the year 2007 namely about over 94 thousand active users by April while the year 2008 during the same 3-month period the tweets reached 100 million with 300,000 messages per day and over a million active users by March [Claudine,B.,2010]. In the following years, 2009 to 2011 the active users were 18 million, 58 million and 117 million respectively. This rapid rate of increase continued in the years 2012, with 185 million active users and 340 million tweets per day, 2013 with 241 million active users and 500 million tweets per day, 2014 with 288 million active users, 2015 with 305 million active users, 2016 with 318 million active users, 2017 with 330 million active users and 2018 with 350 million active users [Statista]. Figure 2.2 illustrates the active users between the years 2007 to 2018.

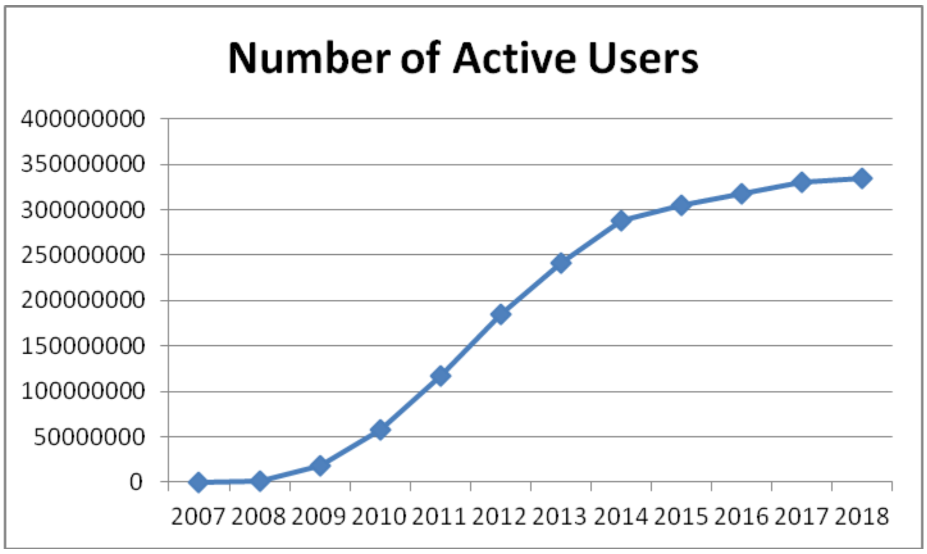


Figure 2.2: The active users of Twitter the years 2007-2018.

The continuous rise in the number of users has produced record numbers in tweets some of which are the following [Mid-day]:

1. On 2 August 2013 the total number of tweets was 143,199 per second during a television screening of the movie ″Castle in the Sky″, when some fans tweeted the word ″balse″ making the most tweeted moment in the history of Twitter.
2. A selfie of Helen DeGeneres with 12 Oscar winning actors posted on 2nd March during the 86 Academy Awards ceremony was retweeted over 2,8 million times within 24 hours.
3. On 25th June 2009 the number of tweets regarding music legend Michael Jackson's death reached 100,000 per hour causing the Twitter servers to crash when users decided to update their status to include his name.

2.4 Justification for choosing Twitter

Taking the above into account, it can be concluded that Twitter consists of a revolutionary way of communication and exerting influence on public opinion. But, why do users choose Twitter? Some reasons for utilizing Twitter are:

1. it is a website, besides a social network, that the users can quickly and validly be informed of the most important issues that happen in the world when they happen as well as posting issues of their interest.
2. it plays an important role in politics, as many politicians use it on a daily basis, as well as in a campaign period. Also, it can be a great tool for academics to promote their research, as well as a good metric for measuring the response of the audience to what they are writing, for the scientists and developers also, as they gain access to huge volumes of data for social research providing a public record of peoples' attitudes, beliefs, and activities over a long time horizon, for businesses, can also benefit as using Twitter, as it has attracted advertising and marketing interest to improve brand awareness, co-operating and loyalty for customers[Sledgianowski,D.,& Kulviwat,S., 2009]
3. Twitter's API is well-designed and easy to access with the Twitter data being freely available to the public via the Twitter Streaming API and being a convenient format for analysis.
4. twitter's terms of use for the data are relatively liberal as compared to other APIs and Twitter is based on the asymmetric following model that allows access to any account without request for approval.

2.5 Twitter API

Twitter data are the "electronics words of mouth" happening at the speed of time thought which are available for consumption in real-time and can be obtained from anywhere in the world. For tweet collection, Twitter provides a rather robust API. In particular, APIs generally are machine-readable interfaces that connect multiple applications, provide methods to govern application interaction and establish communication between applications without the need to know the inner workings of how an API's functionality is provided [Jacobson, D et al., 2011]. On Twitter, there are two possible methods to gather tweets: the Streaming API and the Search API. These methods are different in design and in the way they access Twitter data. In this dissertation, the Streaming API method is used due to the fact that it can provide a continuous stream of tweets as they are posted in real-time, having a limitation to only have access to a 1% sample of all Twitter data at most (namely, 1 million Twitter data a day) by providing some parameters (the keywords, geographical boundary boxes and userID) [Twitter Developer].

Furthermore, the Streaming API method allows users to obtain real-time access to tweets from an input query. The user first requests a connection to a stream of tweets from the server. Then, the server opens a streaming connection and tweets are streamed in as they occur to the user. For better understanding, Figure 2.3 indicates the described method. The Streaming API method has an advantage as after the streaming process, it retrieves tweets from the stream and it can manipulate data before storing the results. User requests to access data are then handled with the HTTP process, which query requested data from data storage. Twitter service offers several types of streaming API endpoints that can be customized to different use cases. These endpoints are: (1) The Public Streams contain streams of public tweets and they are suitable to follow specific topics or specific users, as well as they are recommended for data mining, (2) The User Streams contain streams that relate to all information associated with a particular Twitter user, (3) The Site Streams are targeted to applications connected to Twitter by multiple users. In the dissertation is selected to use a Public Stream endpoint.

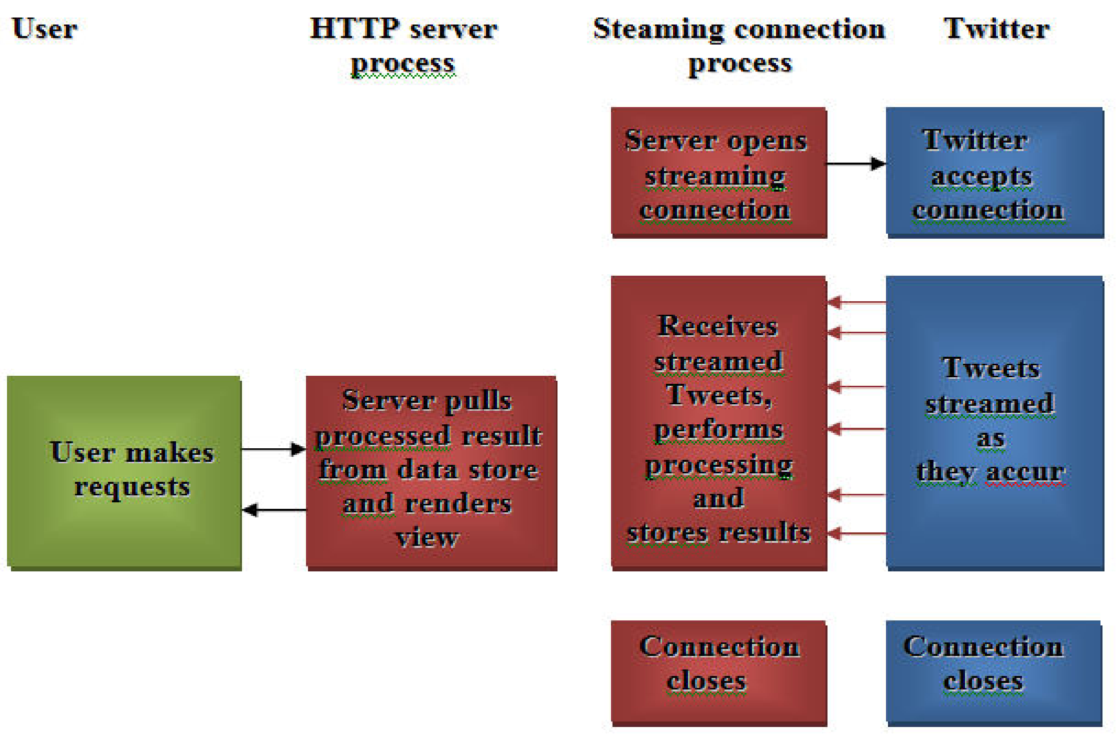

User
HTTP server process
Steaming connection process
Twitter
Server opens streaming connection
Twitter accepts connection
Receives streamed Tweets, performs processing and stores results
Tweets streamed as they accur
User makes requests
Server pulls processed result from data store and renders view
Connection closes
Connection closes

Figure 2.3: Twitter API connection process via Streaming API.

The connection to any of the Twitter API can only be established by requests which must be authenticated by Twitter using Open Authentication (OAuth), which is an authorization protocol that allows secure authenticated requests to be sent to the Twitter API [Twitter Developer, OAuth with the Twitter API]. As the passwords are easy to violate, OAuth provides a safer way to connect to Twitter APIs. In this way, a user's password is not displayed anywhere and is not shared with other applications. The user is needed to create an app on the relevant Twitter page through which he will eventually communicate with the API. The process is represented in Chapter 3.

Chapter 3 Methodology

The chapter presents the necessary procedure to be followed, with the main purpose of collecting the tweets. Initially, the steps followed to build a Twitter application are described in order to be gained access to Twitter data. Then, the codes are being described that used for the communication with the API, the collection of particular tweets and the storing of them. Furthermore, the programs and the dataset are being presented that were utilized in this thesis.

3.1 Twitter Application Setup

In order to get access to real-time data from Twitter, is needed the building of an application that interacts with the Twitter API. The steps below feature this procedure. The first prerequisite is that it is required a Twitter account where the registration is realized at the website https://twitter.com/. Next, with the Twitter login ID and password, at the website, https://apps.twitter.com/ is being realized the building of the Twitter application. By clicking on the "Create New App" button, a new application can be created. Figure 3.1 indicates the described process.

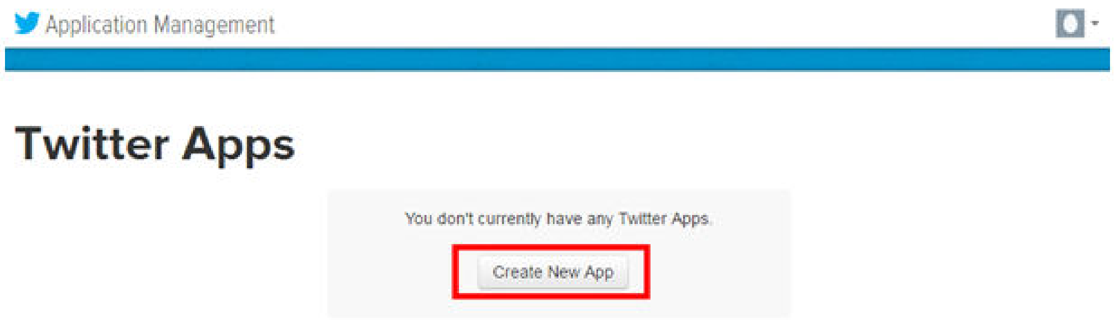
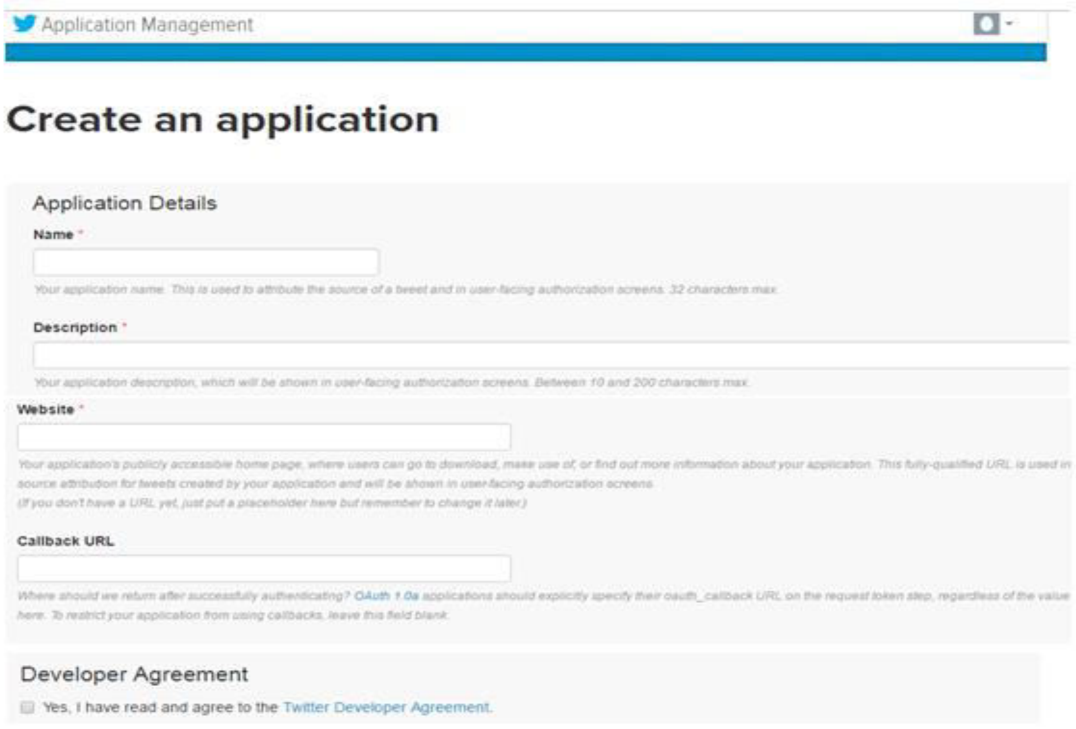


Figure 3.1: Creating of a new application.

In Figure 3.2 all the required fields must be filled in and then submit the form.



Application Management

Create an application

Application Details

Name *

Your application name. This is used to attribute the source of a tweet and in user-facing authorization screens. 32 characters max.

Description *

Your application description, which will be shown in user-facing authorization screens. Between 10 and 200 characters max.

Website *

Your application's publicly accessible home page, where users can go to download, make use of, or find out more information about your application. This fully-qualified URL is used in source attribution for tweets created by your application and will be shown in user-facing authorization screens.
(If you don't have a URL yet, just put a placeholder here but remember to change it later.)

Callback URL

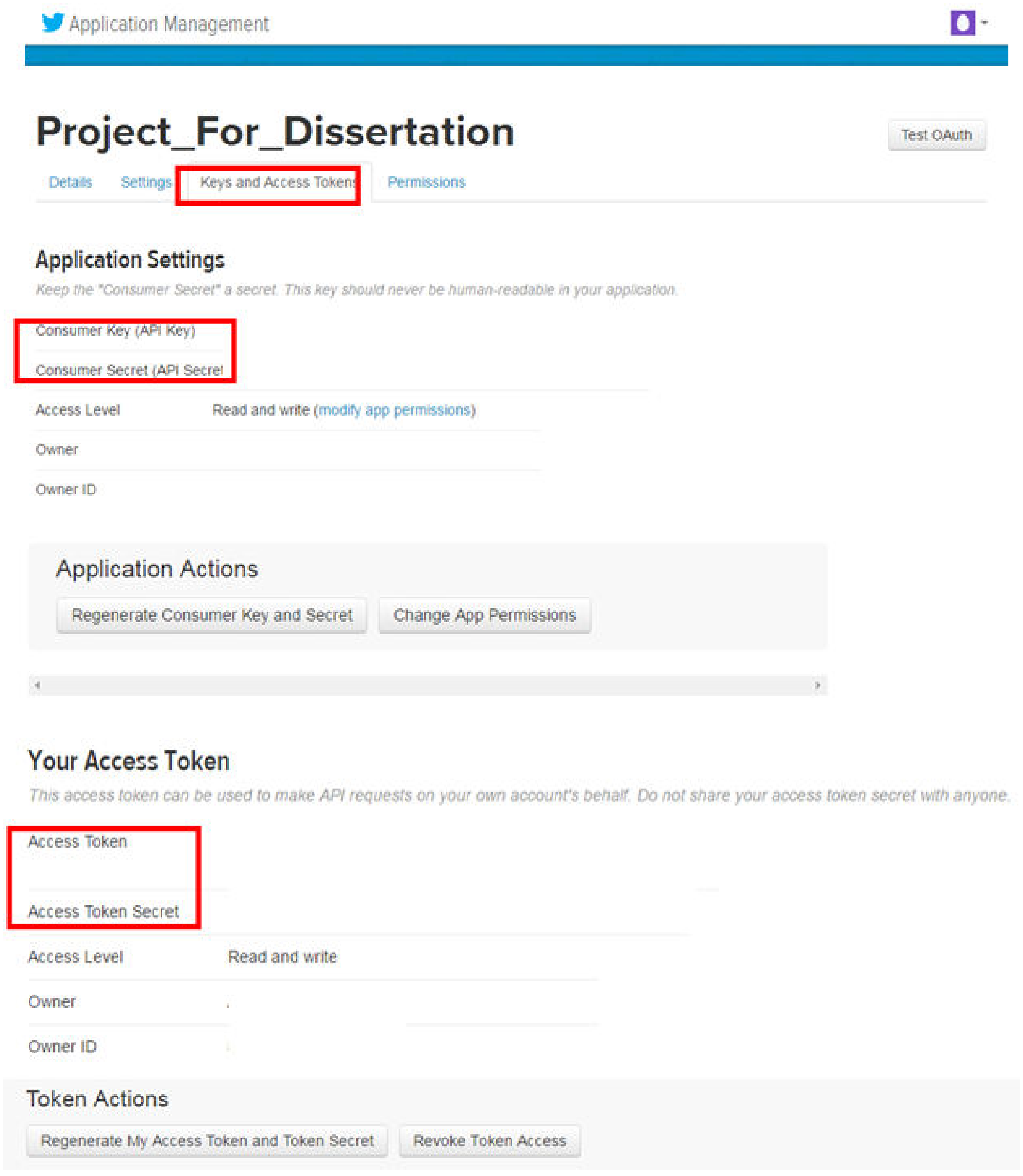
Where should we return after successfully authenticating? OAuth 1.0a applications should explicitly specify their oauth_callback URL on the request token step, regardless of the value here. To restrict your application from using callbacks, leave this field blank.

Developer Agreement

☐ Yes, I have read and agree to the Twitter Developer Agreement.

Figure 3.2: Registration in application form.

In Figure 3.3, by going to the tab ″Keys and Access Tokens″, the keys and access tokens are received which are the Twitter API keys. The consumer and secret keys concern the connection between the application and the user of Twitter who created it, whilst the access token and secret keys authenticate the application with the Twitter API for the user to be able to connect to it and submit their requests. Once the keys have been acquired they can be used in the user's application. In general, these keys are confidential and need to be kept private and provide read-only permissions.


Application Management

Project_For_Dissertation

Test OAuth

Details Settings Keys and Access Tokens Permissions

Application Settings

Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.

Consumer Key (API Key)	
Consumer Secret (API Secret	
Access Level	Read and write (modify app permissions)
Owner	
Owner ID	

Application Actions

Regenerate Consumer Key and Secret | Change App Permissions

Your Access Token

This access token can be used to make API requests on your own account's behalf. Do not share your access token secret with anyone.

Access Token	
Access Token Secret	
Access Level	Read and write
Owner	
Owner ID	

Token Actions

Regenerate My Access Token and Token Secret | Revoke Token Access

Figure 3.3: Acquisition of the Twitter API keys.

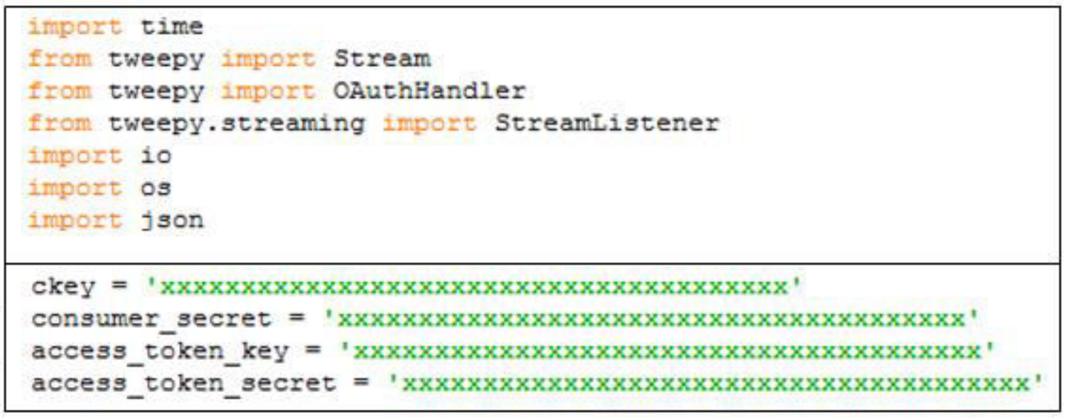
3.2 Creating Streaming Connection and collecting Twitter data

After successfully obtaining credentials for Twitter Streaming API, it is needed to initialize connection for the collection of the tweets. For this implementation, it was chosen to use the Python programming language, as it contains a library called tweepy. This library is an easy-to-use Python library for access to the Twitter API that allows

Python to communicate with the Twitter platform and use its API. The Python scripts follow are based on www.stats.seandolinar.com and the Python version of 2.7.12 is used.

1

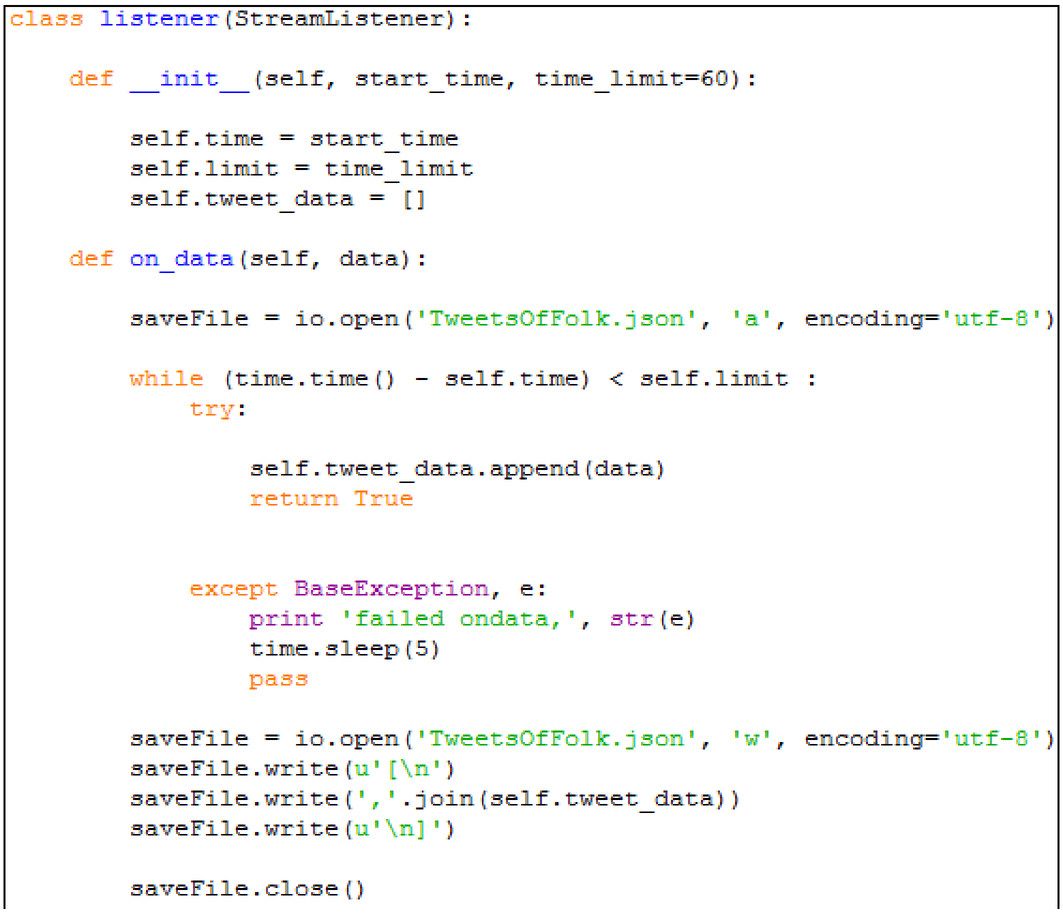
2



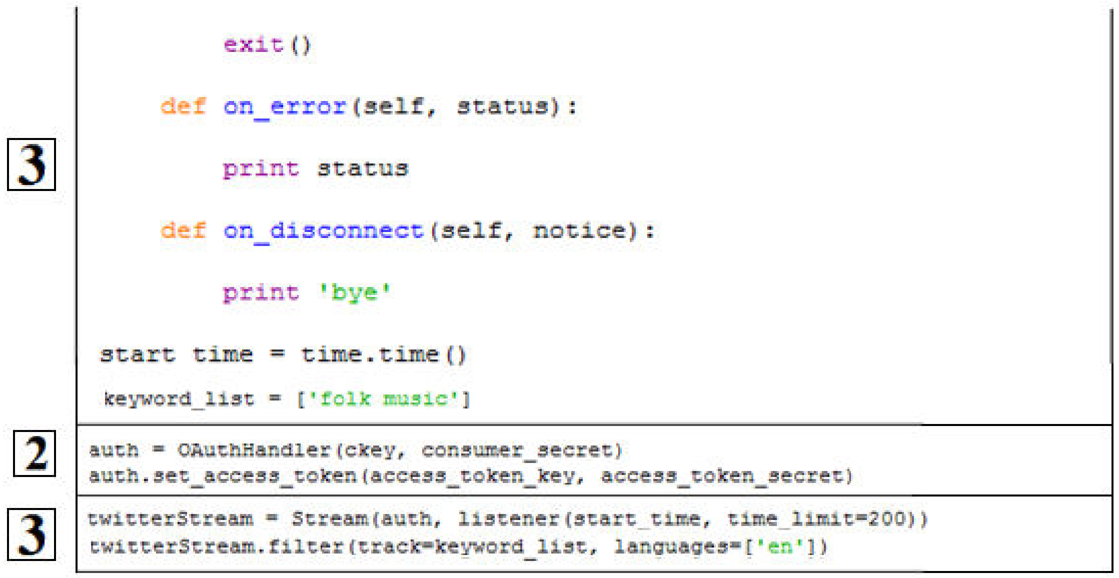
```
import time
from tweepy import Stream
from tweepy import OAuthHandler
from tweepy.streaming import StreamListener
import io
import os
import json
```

```
ckey = 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
consumer_secret = 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
access_token_key = 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
access_token_secret = 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
```

3



```
class listener(StreamListener):

    def __init__(self, start_time, time_limit=60):

        self.time = start_time
        self.limit = time_limit
        self.tweet_data = []

    def on_data(self, data):

        saveFile = io.open('TweetsOfFolk.json', 'a', encoding='utf-8')

        while (time.time() - self.time) < self.limit :
            try:

                self.tweet_data.append(data)
                return True


            except BaseException, e:
                print 'failed ondata,', str(e)
                time.sleep(5)
                pass

        saveFile = io.open('TweetsOfFolk.json', 'w', encoding='utf-8')
        saveFile.write(u'[\n')
        saveFile.write(','.join(self.tweet_data))
        saveFile.write(u'\n]')

        saveFile.close()
```



```
            exit()

        def on_error(self, status):

            print status

        def on_disconnect(self, notice):

            print 'bye'

    start_time = time.time()

    keyword_list = ['folk music']
```

```
auth = OAuthHandler(ckey, consumer_secret)
auth.set_access_token(access_token_key, access_token_secret)
```

```
twitterStream = Stream(auth, listener(start_time, time_limit=200))
twitterStream.filter(track=keyword_list, languages=['en'])
```

1: Initially, the necessary modules are presented.

2: Then, OAuth needs to be used in order to authorize the application to access Twitter on behalf of the created account, thus, this snippet of code presents the code that ensures the access to Twitter using the corresponding credentials.

3: Furthermore, the class is listener instance for Twitter streaming that can gather the upcoming tweets and store them into the file called ″TweetsOfFolks.json" and can inform the user for errors occur with streaming. After auth and listener() instances are passed to Stream object that will now contain the credentials for authentication and information about tweets which have to be streamed, Twitter streaming is ready. Additionally, the filter() method called on Twitter Stream will take input stream word for the term 'folk music' from the user and collect only tweets in the English language. Here are some examples to clarify the matching of term or phrases in a keyword_list of track parameter.

′pop′→ matches tweets containing the term like ′POP′, ′#pop′, ′@pop′

′pop music′→ matches tweets containing the phrase ′pop music′

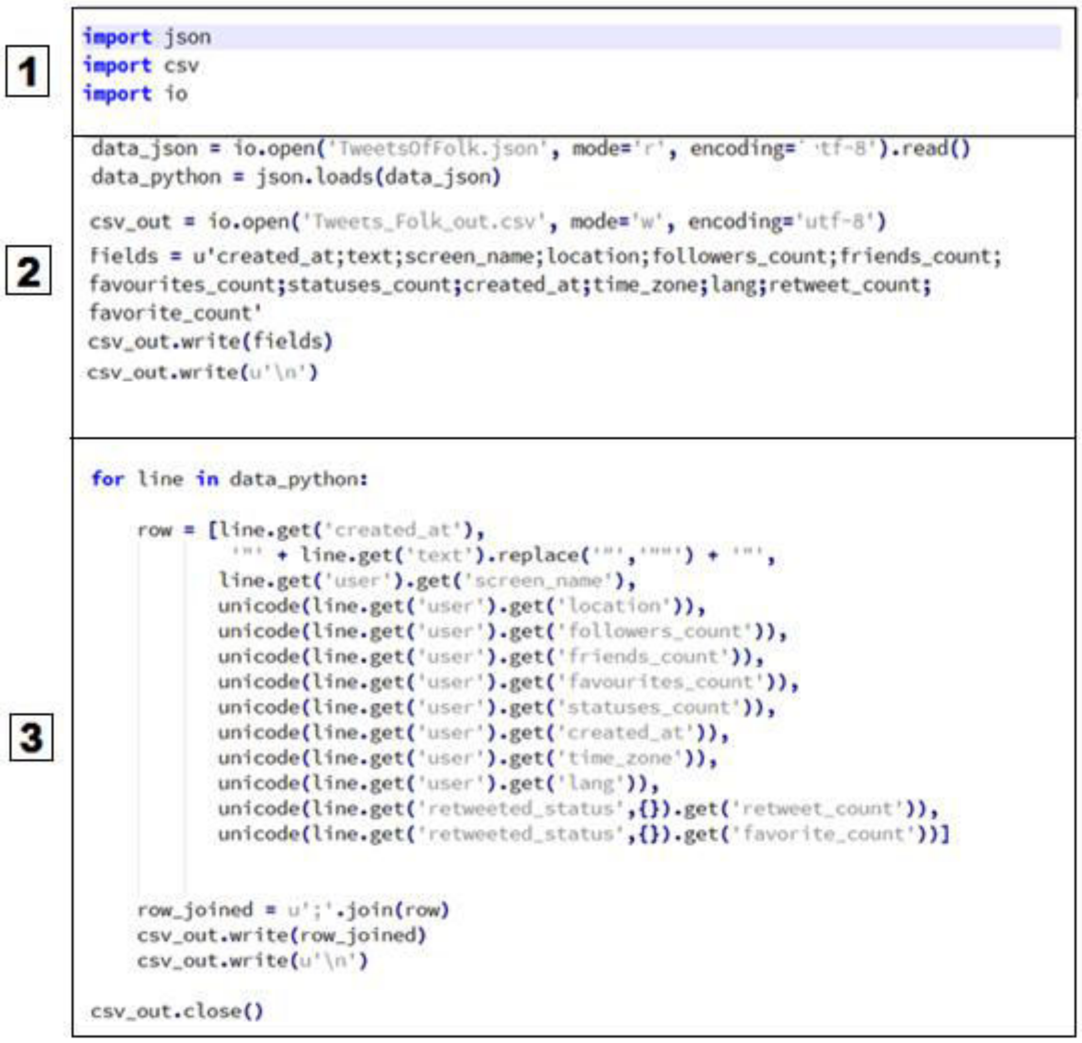
pop music→ matches tweets containing the terms separately in the tweet namely, ′Adele is a famous pop star with amazing voice′

′pop,music′→ matches tweets containing either the term pop or the term music in any order.

3.3 Export and storing the tweets

Tweets are accessed via the Twitter API as "status objects", which are structured JSON formatted objects that contain all the metadata about both the individual tweet and Twitter user. JSON is based on key-value pairs, with named attributes and associated values. These attributes and their values are used to describe objects [Twitter Developer, Introduction to Tweet JSON]. A JSON file has over 160 associated attributes and includes a variety of information, such as username, location, retweets, media, links, geo metadata and many more. An example of a fragment of a JSON tweet is represented in Figure 3.4. The attributes were used in this thesis are those that are tabulated in Table 3.1, as these have valuable insights for analysis. A definition for the JSON is as follows:
JSON (JavaScript Object Notation) is a lightweight data- interchange format [JSON, Introduction to JSON]. The data exchange takes place mostly between a client and a server on the Web. JSON is based on the JavaScript programming language, but it can also be used in most other programming languages that have many libraries which include codes to generate and parse JSON format data [Wikipedia. JSON].

After collecting the incoming tweets and storing them to "TweetsOfFolk.json" file, the following code can convert this file into a more appropriate format for use by exporting the specific attributes we need.

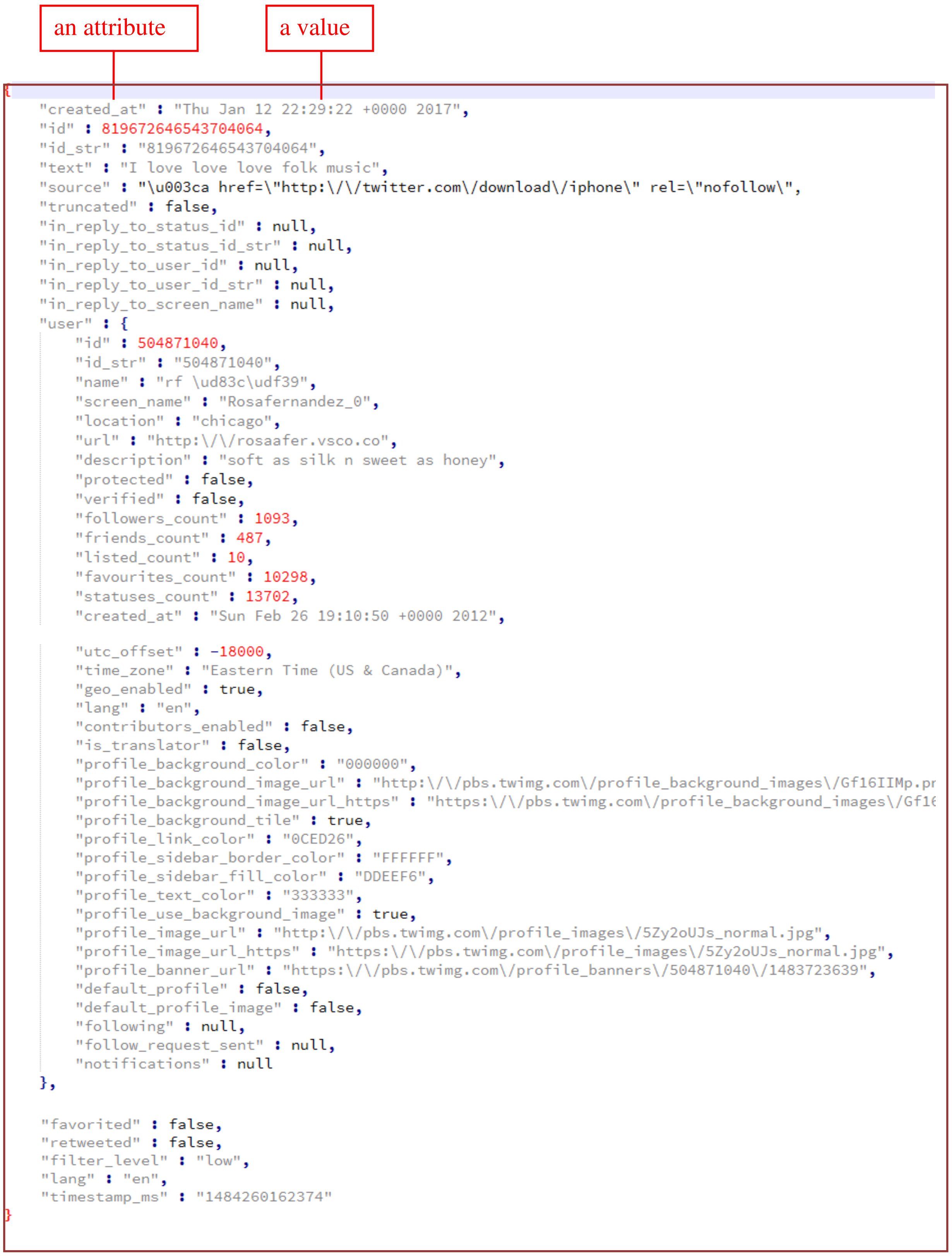


```
import json
import csv
import io

data_json = io.open('TweetsOfFolk.json', mode='r', encoding='utf-8').read()
data_python = json.loads(data_json)

csv_out = io.open('Tweets_Folk_out.csv', mode='w', encoding='utf-8')
fields = u'created_at;text;screen_name;location;followers_count;friends_count;
favourites_count;statuses_count;created_at;time_zone;lang;retweet_count;
favorite_count'
csv_out.write(fields)
csv_out.write(u'\n')

for line in data_python:

    row = [line.get('created_at'),
            '"' + line.get('text').replace('"','""') + '"',
           line.get('user').get('screen_name'),
           unicode(line.get('user').get('location')),
           unicode(line.get('user').get('followers_count')),
           unicode(line.get('user').get('friends_count')),
           unicode(line.get('user').get('favourites_count')),
           unicode(line.get('user').get('statuses_count')),
           unicode(line.get('user').get('created_at')),
           unicode(line.get('user').get('time_zone')),
           unicode(line.get('user').get('lang')),
           unicode(line.get('retweeted_status',{}).get('retweet_count')),
           unicode(line.get('retweeted_status',{}).get('favorite_count'))]


    row_joined = u';'.join(row)
    csv_out.write(row_joined)
    csv_out.write(u'\n')

csv_out.close()
```

1: The necessary Python modules are imported.

2: Then, in the variable called data_json, the downloaded JSON file is opened and read as a string while in the variable data_python, the string is decoded into a JSON Python object, namely a Python dictionary using the json.loads() method. In addition, in the following code snippet, a CSV output file is opened in which we could write. The write() method requires Unicode objects thus, the necessary attributes, are written in the appropriate way.

3: This block of code presents a CSV parser that gets the different attributes we are interested in from a JSON Python object. The line.get('*attribute*')retrieves the relevant information from the tweet. Additionally, the semicolon (;) is used to separate the different attributes. The get() method is used the second time for helping to go to a second nesting level and to get deeper into this level the ({}) must be used.



```
{
    "created_at" : "Thu Jan 12 22:29:22 +0000 2017",
    "id" : 819672646543704064,
    "id_str" : "819672646543704064",
    "text" : "I love love love folk music",
    "source" : "\u003ca href=\"http:\/\/twitter.com\/download\/iphone\" rel=\"nofollow\",
    "truncated" : false,
    "in_reply_to_status_id" : null,
    "in_reply_to_status_id_str" : null,
    "in_reply_to_user_id" : null,
    "in_reply_to_user_id_str" : null,
    "in_reply_to_screen_name" : null,
    "user" : {
        "id" : 504871040,
        "id_str" : "504871040",
        "name" : "rf \ud83c\udf39",
        "screen_name" : "Rosafernandez_0",
        "location" : "chicago",
        "url" : "http:\/\/rosaafer.vsco.co",
        "description" : "soft as silk n sweet as honey",
        "protected" : false,
        "verified" : false,
        "followers_count" : 1093,
        "friends_count" : 487,
        "listed_count" : 10,
        "favourites_count" : 10298,
        "statuses_count" : 13702,
        "created_at" : "Sun Feb 26 19:10:50 +0000 2012",

        "utc_offset" : -18000,
        "time_zone" : "Eastern Time (US & Canada)",
        "geo_enabled" : true,
        "lang" : "en",
        "contributors_enabled" : false,
        "is_translator" : false,
        "profile_background_color" : "000000",
        "profile_background_image_url" : "http:\/\/pbs.twimg.com\/profile_background_images\/Gf16IIMp.pr
        "profile_background_image_url_https" : "https:\/\/pbs.twimg.com\/profile_background_images\/Gf16
        "profile_background_tile" : true,
        "profile_link_color" : "0CED26",
        "profile_sidebar_border_color" : "FFFFFF",
        "profile_sidebar_fill_color" : "DDEEF6",
        "profile_text_color" : "333333",
        "profile_use_background_image" : true,
        "profile_image_url" : "http:\/\/pbs.twimg.com\/profile_images\/5Zy2oUJs_normal.jpg",
        "profile_image_url_https" : "https:\/\/pbs.twimg.com\/profile_images\/5Zy2oUJs_normal.jpg",
        "profile_banner_url" : "https:\/\/pbs.twimg.com\/profile_banners\/504871040\/1483723639",
        "default_profile" : false,
        "default_profile_image" : false,
        "following" : null,
        "follow_request_sent" : null,
        "notifications" : null
    },

    "favorited" : false,
    "retweeted" : false,
    "filter_level" : "low",
    "lang" : "en",
    "timestamp_ms" : "1484260162374"
}
```

Figure 3.4: The structure of JSON of a tweet.

Table 3.1: Attributes overview.

Atributes	Description
Created_at	The UTM time when this tweet was created
Text	The actual UTF-8 text of the status update
Screen_name	The screen name or alias that this user identifies themselves with
Location	The user-defined location for this account's profile
Followers_count	The number of users this account is following
Friends_count	The number of users this account is following, also known as their "followings"
Favourites_count	The number of Tweets this user has liked in the account's lifetime
Statuses_count	The number of Tweets (including retweets) issued by the user
Time_zone	A string describing the Time Zone this user declares
Lang	Indicates a BCP 47 language identifier corresponding to the machine-detected language of the Tweet text
Retweet_count	The number of times this tweet has been retweeted
Favorite_count	The number of times this tweet has been liked by Twitter user's

3.4 Technologies and Packages

Two software programs are utilized in this thesis are Python and R. This chapter will cover their descriptions and necessary information of these programs.

3.4. 1 Python programming language

Python was conceived in the late 1980s and its implementation was started in December 1989 by Guido van Rossum at CWI (Centrum Wiskunde & Informatica) in the Netherlands as a successor to the ABC programming language capable of exception handling and interfacing with the Amoeba operating system [Wikipedia. Python (programming language)]. Python is a dynamic and high-level yet simple programming language with important features and incredible possibilities, having uses in sectors such as, web development, data analysis, machine learning and game development among many others. Python distinguishes from other programming languages because [Dave Kuhlman, 2011]:

1. it combines significant power with very clear syntax using multiple levels of organizational structures such as modules, classes, functions, packages as well as very high-level data types such as strings, lists, dictionaries.
2. it borrows elements from imperative, object-oriented, functional and procedural programming.
3. it extends in C and C++ adding in Python new modules and types written in these languages.
4. it comes standard with many libraries, including those for mathematical function, XML(Extensible Markup Language), parsing downloading web pages, cryptography, GUI(Graphical User Interface) and so on.
5. it automatically compiles the code to bytecode and executes it, characterizing Python as a scripting language.

3.4.2 R programming language

R is a system for statistical analyses and graphics created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand, and is currently developed by the R Development Core Team [Wikipedia. R(programming_language)]. It is a GNU project, i.e. anyone can make improvements to its code, which is similar to the language and environment S language created at the AT&T Bell Laboratories by John Chambers and colleagues [The R Project for Statistical Computing, Introduction to R]. R can be considered as a different implementation of S. The core of R is an interpreted computer language which allows branching and looping as well as modular programming using functions [what is R?,FAQs on R]. Many of R's standard functions are written in R itself, which makes it easy for users to follow the algorithmic choices made. In addition, it allows the user to interact with other languages (C / C ++, Java, Python) with data files (Excel, Access) and other statistical packages (SAS, Stata, SPSS, Minitab). R with its libraries (more than 15,000 libraries) implements a wide variety of statistical and graphical techniques, including linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, and is highly extensible [wikipedia. R(programming_language)]. R is available as Free Software under the terms of the Free Software Foundation's GNU General Public License in source form. Also, it compiles and runs on a wide variety of UNIX platforms and similar systems including Linux, Windows and MacOS [The R Project for Statistical Computing, Introduction to R]. The R language is widely used among statisticians and data miners for developing statistical software and data analysis.

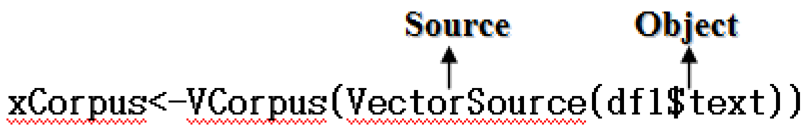
R stand out from other programming languages for some key advantages [Roger.,D, 2015; Mizumoto.,A et al., 2016]. First, R is an open-source programming language and software environment, free and the most popular platform mainly for statistical programming and applied machine learning. Second, today R runs on almost any standard computing platform and operating system thus, it has been reported to be running on modern tablets, phones, PDAs, and game consoles. Third, R has sophisticated graphics capabilities where its base graphics system allows for very fine control over essentially every aspect of a plot or graph, an example of which is the ggplot2 library that allows complex and sophisticated visualizations of high-dimensional data. Fourth, R provides an elegant and powerful web framework called Shiny that it used to build interactive web apps straight from R. Fifth, R makes the range of possible uses and functions nearly unlimited. Its flexibility enables the ideas suggested by a community of professional statisticians and computer scientists to be integrated almost instantaneously.

3.4.3 Tm package

R has obtained explicit support for text mining through the *tm* package originally presented by Ingo Feinerer. Until recently, R had lacked a framework for text mining purposes, thus the *tm* package allows R users to work efficiently with texts and transform them into structured representations where existing R methods can be applied for clustering or classification. Furthermore, tm package provides native support for handling a number of standard file formats such as plain text, PDF files or XML files.

Data structures and algorithms can be expanded to meet custom requirements since the package is designed in a modular way to allow easy integration of new files forms, transformations, and filtering operations. *Tm* package provides easy access to pre-processing and manipulation mechanisms such as whitespace removal or stemming between file formats. In addition, a general architecture is available for filtering documents for certain criteria, or for perform full-text search.

At this point, it is explained how the tm package is structured, specifically, the procedure and the functions that were used. Initially, a corpus should be constructed to implement the tm package. Corpus is the main structure for document management, namely, a collection of documents that contain text in natural language. Like most things in R, the corpus has specific attributes that enable certain types of analysis. Corpora in R exist in two ways: a) the Volitile Corpus (VCorpus) is a temporary object within R and is the default when assigning documents to a corpus, b) Permanent Corpus (PCorpus) is a permanent object that can be stored outside R. Once the type of corpus is defined, it is then necessary to identify a source. The source that was used in the thesis is the VectorSource , that is, a vector of characters that treat each component as a document. Therefore, to create a corpus is needed to define the source and the object. The example below indicates the function that was used for creating the corpus from our code.



```
xCorpus<-VCorpus(VectorSource(df1$text))
```

Since the corpus has been created, then is needed to modify the documents (texts). This is accomplished by applying a series of transformations, such as removing stop words, whitespace, punctuations, etc, to clean all corpus documents in order to obtain a convenient representation for later analysis. The transformations the *tm* package offers are:

- clean.corpus <- tm_map(xCorpus,content_transformer(tolower)), the function that converts the text to lower case letters.
- clean.corpus<-tm_map(clean.corpus,content_transformer(removeWords)), the function that removes stop words.
- clean.corpus <- tm_map(clean.corpus, removeNumbers), the function that removes the numbers.

- lean.corpus <- tm_map(clean.corpus, removePunctuation), the function that removes punctuation marks.
- clean.corpus <- tm_map(clean.corpus, stemDoc), the function that stems the text documents.
- clean.corpus <- tm_map(clean.corpus, stripWhitespace), this function removes extra whitespace.

The next in the process includes the creation of the document term matrix (DTM), that is, a matrix that lists all occurrences of words in the corpus, by a document. The DTM is a common format for representing a text corpus in a bag-of-words format, in which rows are documents, columns are terms, and cells indicate how often each term occurred in each document. The *tm* package permits a corpus to be created by assigning *tf* and *tf-idf* weights upon creation DocumentTermMatrix() function. More specifically,

- clean.corpus.dtm <- DocumentTermMatrix(clean.corpus): with this way is adjusted the weighting to *tf*.
- clean.corpus.dtm<-DocumentTermMatrix (clean.corpus,control=list(weighting=function(x)weightTfIdf(x,normalize= FALSE))): with this way is adjusted the weighting to *tf-idf*.

3.4.4 Ggplot2 package

ggplot2 is a powerful and a flexible data visualization package for the statistical programming language R, created by Hadley Wickham in 2005. Ggplot2 is an implementation of Leland Wilkinson's Grammar of Graphics. The concept behind ggplot2 divides a plot into three different fundamental parts [Alboukadel Kassambara, 2013]:

Plot = data + Aesthetics + Geometry. The principal components of every plot can be defined as follow:

- data: contains a data frame
- Aesthetics: is used to indicate x and y variables. It can also be used to control the color, the size or the shape of points, the height of bars, etc.

- Geometry: defines the type of graphics such as: histogram, box plot, line plot, density plot, dot plot.

An instance for understanding the above could be the:

plot= ggplot(data=mpg,aes(x=displ,y=hwy,colour=class,fill=class))+geom_histogram()

3.5 Dataset Description

The tweets of Twitter were used as a data source and data were collected through the Twitter Streaming API and the tweepy package. Data were collected from time December 30, 2016 to Feb 04, 2017.The collected data is in JSON format, meaning that each line of the output stream is a tweet encoded as a JSON object. The subject of interest of the project is five different music genres: pop, rock, rap, classical and folk. Each music genre has the following tweets: classical music has 3415 tweets, folk 2090 tweets, pop 4243 tweets, rap 3606 tweets and rock 3446 tweets, hence the total raw tweets are 16800(81.1MB). Table 3.2 presents the elements as well as the rates of occurrence of the five different music genres in the first and second datasets. The Twitter search query for finding the tweets of interest includes keywords. The keywords were used are “classical music”, “folk music”, match the tweets containing the exact phrase "classical music". No hash-tag sign (#) was used in keywords. Furthermore, only English tweets were extracted. The dataset consists of 16800 tweets (observations) and 12 attributes that include both numerical and categorical attributes. In particular, the attributes utilized are: created_at, lang, time_zone, followers_count, friends_count, favourites_count, statuses_count, retweet_count, favorite_count, creating thus the dataset with the name file tweetsOfAllMusicGenres, which was used to explore those attributes.

In addition, a second dataset was generated with the name file tweetsOfTexts, which was used for applying text classification. More specifically, out of 16800 thousand tweets that were collected, only the unique tweets were used, while there were 6217 deduplicates, accounting for 37% of the total number of tweets and they were removed since they do not help for the analysis. Thus, tweetsOfTexts dataset includes unique tweets, which are automatically tagged with “music_genres” label. The dataset, therefore, has two columns: the Text, which contains the 10583 tweets and the Music_genres ("classical_music", "folk_music", "pop_music", "rap_music", "rock_music"), among which 2474 tweets were automatically labeled based on keywords retrieved as "classical_music", 1500 tweets were automatically labeled as "folk_music", 2441 tweets were automatically labeled as "pop_music", 2319 tweets were automatically labeled as "rap_music" and 1849 tweets were automatically labeled as "rock_music".

A third dataset was generated with the name file tweetsOfTexts2, which was used for applied text clustering. The file contains only one column with the text tweets, that is, the 10583 tweets. The frequency of the received data was almost daily. Specifically, for half an hour a day, the received tweets for the pop, rock and rap music genres approached approximately the two hundred tweets, while for the classical and folk music genres the received tweets were about sixty tweets.

Table 3.2: Distribution of the tweets of tweetsOfAllMusicGenres and tweetsOfTexts datasets.

Music_genres	**# of Tweets**	**%**	**# of Tweets**	**%**
Classical_music	3415	20.33%	2474	23.38
Folk_music	2090	12.45%	1500	14.17
Pop_music	4243	25.25%	2441	23.07
Rap_music	3606	21.45%	2319	21.91
Rock_music	3446	20.52%	1849	17.47
	16800	**100%**	**10583**	**100%**

Chapter 4 Theoretical Background

The chapter describes in detail the theoretical background regarding the key components of our work. It is being presented the area of the text mining and their tasks as well as the algorithms for the implementation of them. In addition, were described the different techniques that were used in the text attribute as well as the evaluation measures that were utilized in the algorithms.

4.1 Text Mining

The flood of social networks from various forms of data creates a basic need for the exploitation, mining, processing and analyzing them so that they can be converted into information that will allow us to discover the knowledge that lies behind them. The bulk of the information disseminated on the internet is in the form of text. For this reason, text mining is very important nowadays as it is an important step in knowledge discovery.
Text mining is an interdisciplinary field that draws on information retrieval, data mining, machine learning, statistics and computation linguistics. The goal of text mining is to derive high-quality information from a text. Specifically, text mining is a process that employs a set of an algorithm for converting unstructured text into structured data objects and the quantitative methods used to analyze these data objects [Tarsem Singh,2016]. In other words, text mining is the process that turns texts into data that can be analyzed. The extracting knowledge from any unstructured text is a difficult task, so a process must be followed which helps in this task. Figure 4.1 depicts the text mining process that is followed. Modern text mining techniques require the integration of natural language processing operations with several advanced machine learning techniques to understand the meaning of text documents.

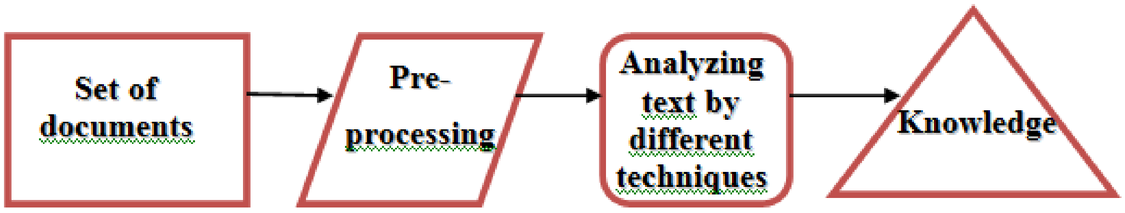


Figure 4.1: Text mining process.

4.2 Text Mining Tasks

Two most typical text mining tasks are text classification and text clustering. Text classification, also known as categorization is the task of assigning a document to a set of two or more pre-defined categories (or classes), based on learning models that have been trained using labeled data (i.e., documents with known class information) [Fragkiskos Malliaros, &Konstantinos Skianis.,2012]. There is a broad range of learning models for text classification, but the Naïve Bayes and Support Vector Machine algorithms are widely used. Text clustering, also known as document clustering is the task of grouping a set of unlabeled text documents into representative groups (categories) called clusters so that documents within a cluster should be as similar as possible and dissimilar from documents in other clusters. [Wikipedia. Cluster analysis]. Some of the most common text clustering algorithms are K-means and Latent Dirichlet Allocation The algorithms of both text mining tasks are described extensively in their sections.

4.3 Algorithms

The chapter describes the text classification algorithms were used in the thesis.

4.3.1 Naïve Bayes algorithm [wikizero. Naïve Bayes Classifier]

NB is a type of statistical probabilistic classifiers. These classifiers calculate class probabilities based on the feature values of the training data and the vector that is being classified. NB is based on Bayes' theorem (4.1), where *P(c|D)* states the conditional probability of *c* given *D*, where *c* is the class label and *D* a feature.

$$P\left(c|D\right) = \frac{P\left(D|c\right)P(c)}{P(D)} \quad (4.1)$$

Essentially the Bayesian approach is to provide a mathematical rule explaining how, our existing beliefs, denoted by *P(c)*, should change in the light of new evidence, denoted by *D*. It also allows calculating unknown conditional probabilities from a known conditional probability together with the prior probabilities.
Considering *c* as a dependent class variable, and D_1 through D_n as features variables, the probability model for a classifier is a conditional model is thus:

$$p(c|D_1,...,D_n) \quad (4.2)$$

The model (4.2) is reformulated to make it more accessible using the Bayes theorem (4.1), so:

$$p(c|D_1,...,D_n) = \frac{p(c)p(D_1,...,D_n|c)}{p(D_1,...,D_n)}$$

The numerator of that fraction is in our interested since the denominator does not depend on *c* and the values of the features D_i are given so that the denominator is effectively constant. The numerator is equivalent to the joint probability model, namely:

$$p(c)p(D_1,...,D_n|c) = p(c,D_1,...,D_n)$$

which can be rewritten as follows, using the chain rule for repeated applications of the definition of conditional probability:

$$
\begin{aligned}
&p(c, D_1, ..., D_n) \\
&= p(c)p(D_1, ..., D_n | c) \\
&= p(c)p(D_1 | c)p(D_2, ..., D_n | c, D_1) \\
&= p(c)p(D_1 | c)p(D_2 | c, D_1)p(D_3, ..., D_n | c, D_1, D_2) \\
&= p(c)p(D_1 | c)p(D_2 | c, D_1)p(D_3 | c, D_1, D_2)p(D_4, ..., D_n | c, D_1, D_2, D_3) \\
&= p(c)p(D_1 | c)p(D_2 | c, D_1)p(D_3 | c, D_1, D_2)...p(D_n | c, D_1, D_2, D_3, ..., D_{n-1})
\end{aligned}
$$

Naive Bayes assumes that the features are independent from each other within each class, that is, each feature D_i is conditionally independent of every other feature D_j for j ≠ i. The assumption means that:

$$
\begin{aligned}
&p(D_i | c, D_j) = p(D_i | c) \\
&p(D_i | c, D_j, D_k) = p(D_i | c) \\
&p(D_i | c, D_j, D_k, D_l) = p(D_i | c) \\
&\cdots\cdots
\end{aligned}
$$

The joint model can be expressed as:

$$p(c, D_1, ..., D_n) = p(c)p(D_1 | c)p(D_2 | c)p(D_3 | c) = ... = p(c)\prod_{i=1}^{n} p(D_i | c) \quad (4.3)$$

This means that under the above independence assumptions, the conditional distribution over the class variable c can be expressed like this:

$$p(c, D_1, ..., D_n) = \frac{p(c)p(D_1, ..., D_n | c)}{Z} = \frac{p(c, D_1, ..., D_n)}{Z} = \frac{1}{Z} p(c)\prod_{i=1}^{n} p(D_i | c) \quad (4.4)$$

where the evidence $Z = p(D) = \sum_k p(c)p(D | c)$ (4.5) is a scaling factor dependent only on $D_1, ..., D_n$,that is, a constant if the values of the feature variables are known.

Naïve Bayes classifier has two models for representing documents and calculating probabilities, the Multivariate Bernoulli model and the Multinomial model [Hetal,Doshi., and Maruti, Zalte.,2011; Mehdi, Allahyari., et al.,2017]. These models compute the posterior probability of a class, based on the distribution of the words in the document, namely, they ignore the actual position of the words in the document and work with the “bag of words” assumption.

- Multivariate Bernoulli model: In this model a document is represented by a vector of binary features denoting the presence or absence of the words in the document. Thus, the frequency of words are not used for the modeling a document.
- Multinomial model: In this model a document is represented by vector of integer features, where the frequencies of terms captured in a document by representing a document with a bag of words. The documents in each class can then be modeled as samples drawn from a multinomial word distribution.

4.3.2 Support Vector Machines

SVM is a classification algorithm applicable to both linear and nonlinear classifiers. SVM goal is to find the optimal line (or hyperplane) that best separates two classes, where best separation means finding the widest gap between classes.
Initially, in the case of exists two classes and therefore a two-dimensional space with a hyperplane, which the hyperplane is a line, the problem is which one of them is the best separating hyperplane? In the ideal situation, which the classes are linearly separable, a line splits the two classes perfectly. Nevertheless, not only one line splits the dataset perfectly, but a whole bunch of lines does. In Figure 4.2, there are many lines that can separate the data, but there is only one that maximizes the margin, namely maximizes the distance between it and the nearest data point of each class. This line is termed the optimal separating hyperplane.

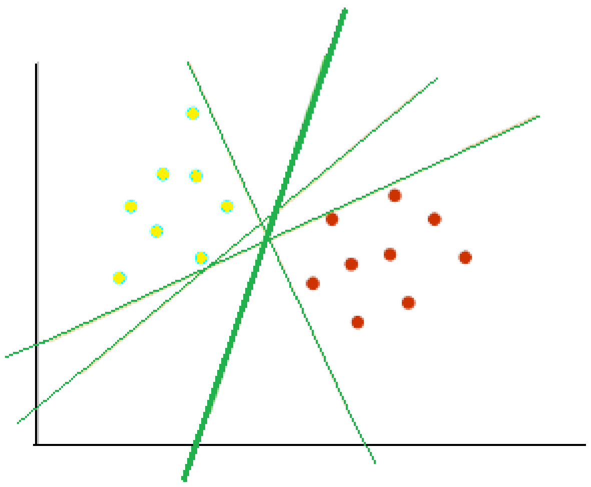


Figure 4.2: Optimal Separating Hyperplane

The mathematical approach of the SVM is represented as follows:

In the binary classification problem, the training set *x* is such that [Luis, Torgo., 2017]:

$$x = \{(x_1, y_1), \ldots, (x_n, y_n)\}, x_n \in \mathbb{R}^K, y_n \in \{-1, 1\}$$

where x_i are the feature vectors of each instances (i.e. observations), and $y_n \in \{-1, 1\}$ is class label. The separating hyperplane can be described by $w \cdot x = b$ (4.6), where *w* is a vector of coefficients and *b* is perpendicular distance from the hyperplane to the origin. These parameters define a separating hyperplane that divides the input space into two half-spaces corresponding to the two classes. Considering that the points of one of the classes have $Y = +1$ while the others have $Y = -1$, then the separating hyperplane is such that:

$$\begin{aligned} w \cdot x_i + b &\geq 0, \forall_i : y_i = +1 \\ w \cdot x_i + b &\leq 0, \forall_i : y_i = -1 \end{aligned} \quad (4.7)$$

Assuming that the separating hyperplane is in the center of the two maximum margin hyperplanes, H_1 and H_2, as illustrated in Figure 10, and assuming that *m* is the distance from the maximum margin hyperplane to both H_1 and H_2, the equations defining these two hyperplanes are as follows:

$$\begin{aligned} w \cdot x_i + b &= m \\ w \cdot x_i + b &= -m \end{aligned} \quad (4.8)$$

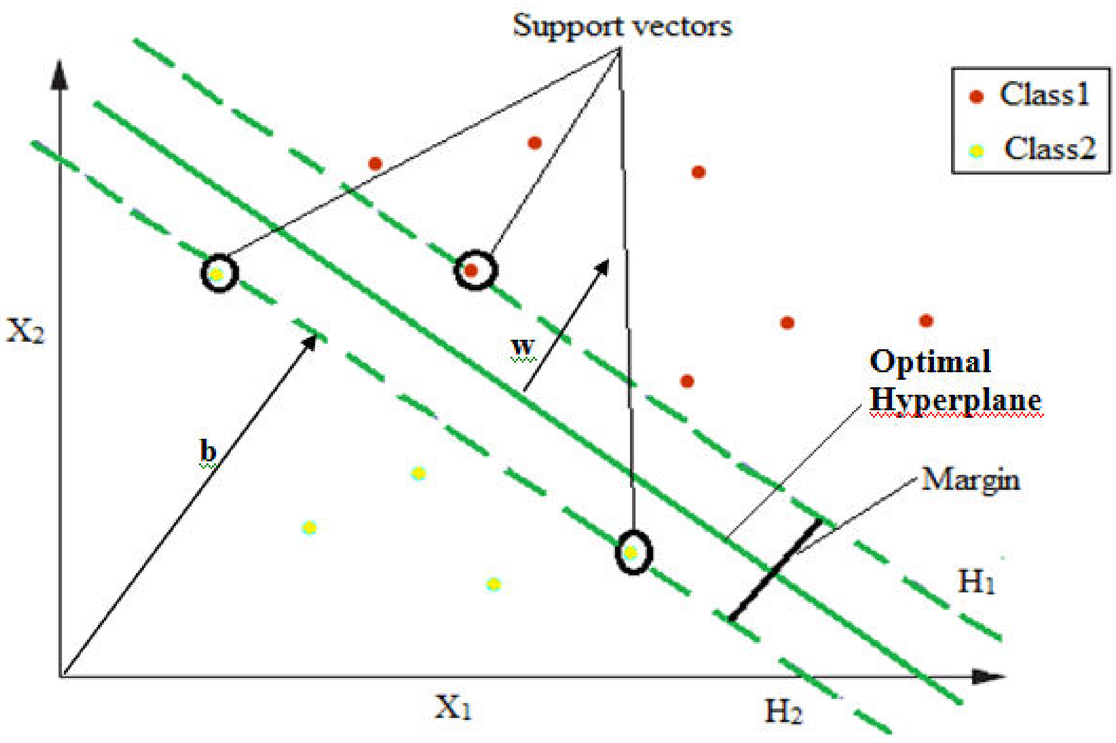


Figure 4.3: A linear SVM with support vectors and a maximum distance to both classes. The support vectors are the points that lie on the margin.

The points in these two hyperplanes are known as the support vectors .
Changing the variables so that m=1, the equations 4.8 can be described by:

$$w \cdot x_i + b \geq +1, \forall_i : y_i = +1$$
$$w \cdot x_i + b \leq -1, \forall_i : y_i = -1 \quad (4.9)$$

These equations can be re-written as:

$$y_i \left(w \cdot x_i + b \right) \geq +1, \forall_i \quad (4.10)$$

SVMs try to maximize the margin between H_1 and H_2, thus using linear algebra it can be proved that the distance between these two hyperplanes is the normalized difference between their constant terms, where the normalization factor is the L_2-norm, $\|w\|$, of the coefficients. The distance between H_1 and H_2 is: $m = \frac{|1-b|}{\|w\|} - \frac{|-1-b|}{\|w\|} = \frac{2}{\|w\|}$, which is the margin that is needed to maximize. This maximization problem is inconvenient because calculating the L_2-norm involves a square root. This can be overcome because maximizing the m= $\frac{2}{\|w\|}$ is equivalent to minimizing $\frac{1}{2}\|w\|^2$ and taking into account the constraints of Equations 4.9, this minimization is a Quadratic Programming (QP) problem. In order to cater for the constraints in this problem is needed the help of Lagrange multipliers. The Lagragian form of this problem is given by:

$$L_p = \frac{\|w\|^2}{2} - \sum_{i=1}^{N} \lambda_i \left[y_i \left(w \cdot x_i + b \right) - 1 \right] \quad (4.11)$$

where $\lambda_1, \ldots, \lambda_N \geq 0$ are the Lagrangian multipliers.

SVM as presented so far assumed that the classes are linearly separable, but in practice, the classes are not usually linearly separable. Thus in such cases, a higher order function can split the classes. To accomplish this, the kernel trick is applied. In particular, Kernels can easily modify linear methods such that they yield non-linear decision functions in the input space. The purpose is to transform a non-linearly separable dataset into a Hilbert space with a higher or even infinite dimension, where the dataset can be separated by a hyperplane. This transformation is performed directly through kernel evaluations, without explicit computation of any feature map. This strategy is called kernel trick. SVM algorithm was evaluated using different kernels functions. The most common types of kernel functions are [Sapna, N.Gaikwad.,D, S. Bormane.,2014]:

- Lineal kernel: $K(x, y) = x^T y, x, y \in \mathbb{R}^d$
- Polynomial kernel: $K(x, y) = \left(x^T y + r \right)^n, x, y \in \mathbb{R}^d, r \geq 0$
- Gaussian kernel(RBF kernel): $K(x, y) = e^{-\frac{\|x-y\|^2}{2\sigma^2}}, x, y \in \mathbb{R}^d, \sigma > 0$
- Sigmoid kernel: $K(x, y) = \tanh(\alpha x^T y + c)$

SVMs are inherently for two-class classifiers, but in the case of involving of SVM into more than two classes (K > 2), there are two main approaches [Jonathan, Milgram et al., 2006]: (1) one-versus-rest classifiers (commonly referred to as "one-versus-all" or OVA classification), and (2) one-versus-one classifiers or OVO classification. In particular, the "one-versus-all" approach consists of constructing one SVM per class, which is trained to distinguish the samples in a single class from the samples in all remaining classes whilst the "one-versus-one" approach consists in constructing one SVM for each pair of classes, thus, for a problem with *k* classes, *k(k-1)/2* SVMs trained to distinguish the samples of one class from the samples of another class.

4.3.3 K-means

Partitional clustering is a type of clustering where all observations in the data are partitioned into *k* different clusters. Many clustering algorithms for their implementations require the number of clusters *k* as an input parameter in order to return a clustering, K-means is one of them. The number of cluster *k* is an unknown parameter, which is a key feature for the starting off of the implementation of the algorithm. Thus, determining the "optimal" number of cluster is a fundamental issue which has a deterministic effect on the clustering results. For this reason, many methods are used for estimating the optimal number of clusters. A very extensive comparative evaluation was conducted by [Milligan, G.W., & Cooper,1985] presents thirty proposed methods.

In addition, a major computational burden, as it can affect the clustering quality while performing document clustering is the calculation of similarity measure between a pair of documents. The similarity measure is a function that assigns a real number between 0 and 1 to a pair of documents, depending upon the degree of similarity between them. The most popular document similarity measures, according to [Gerard, Salton., 1988], are Cosine, Dice, Jaccard. In this thesis was used the Jaccard distance measure. The Jaccard index, also known as the Jaccard similarity coefficient, is a statistic used for comparing the similarity and diversity of sample sets. The Jaccard coefficient measures similarity between sample sets (A and B) and is defined as the difference of the sizes of the union and the intersection of two sets divided by the size of the union of the sets [Mushfeq-Us-Saleheen, Shameem., & Raihana, Ferdous.,2009].

$$J(A,B) = 1 - J(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}$$

K-means clustering is the most widely used partitional clustering algorithm. Its popularity is due to its simplicity and efficiency in clustering large textual datasets. The K-Means clusters a set of documents into *k* clusters based on the attributes of each document, where k is a predefined constant determined upfront by user. The objective of this algorithm is to minimize the average squared distance of documents from their cluster centers. According to Equation 4.13, a cluster center is defined as the centroid $\vec{\mu}$ of the documents in a cluster ω [Christopher,D., Manning., et al.,2008].

$$\vec{\mu}(\omega) = \frac{1}{|\omega|} \sum_{\vec{x} \in \omega} \vec{x} \quad (4.13)$$

The ideal cluster in K-means is a sphere with the centroid as its center of gravity where ideally, the clusters should not overlap. A measure of how well the centroids represent the members of their clusters is the Residual Sum of Squares. RSS is defined as "the squared distance of each vector from its centroid summed over all vectors" and can be calculated with Equations 4.14 and 4.15 [Christopher,D., Manning., et al.,2008]:

$$\text{RSS}_k = \sum_{\vec{x} \in \omega_\kappa} \left| \vec{x} - \vec{\mu}(\omega_\kappa) \right|^2 \quad (4.14)$$

$$\text{RSS} = \sum_{k=1}^{\text{K}} RRS_k \quad (4.15)$$

where in Equation 4.14, ω_k is document cluster *k*, $\vec{\mu}$ is centroid of the document in cluster ω_k and $\vec{x}$ is document vector in cluster *k*. The objective of K-means algorithm is to minimize RSS through modifying the center of clusters.

The basic steps of the algorithm are as follows [Sanjivani, Tushar Deokar., 2013]:

1. Randomly Select K points as the initial centroids (cluster centers)
2. **Repeat**
3. Assign each point to the closest cluster centre

4. Recompute the cluster centres of each cluster
5. **Until** convergence criterion is met

4.3.4 Topic Modeling-Latent Dirichlet Allocation

Topic Modeling is a type of statistical model for discovering the abstract "topics" that occur in a collection of documents [Wikipedia, Topic Model]. A topic consists of clusters of words that frequently occur together. A topic modeling can connect words with similar meanings and distinguish between uses of words with multiple meanings. The main purpose of topic modeling is to discover patterns of word-use and how to connect documents that shared similar patterns [Himanshu, Sharma., & et. al, 2017]. Topic Modeling has four topic models, such as Latent Semantic Analysis (LSA), Probabilistic Latent Semantic Analysis (PLSA), Latent Dirichlet Allocation (LDA), Correlated Topic Model (CTM) [Rubayyi, Alghamdi., & Khalid, Alfalqi., 2015]. The topic modeling algorithm used in this thesis was Latent Pirichlet Allocation (LDA). In general, topic models can help us find a group of words from a collection of documents that best represents the information of the collection. LDA is a generative probabilistic model for collections of discrete data such as text corpora [David,M., Blei, & et al, 2008] where the basic idea is that each document is modeled as a mixture of topics and each topic is a discrete probability distribution that defines how likely each word is to appear in a given topic. These topic probabilities provide a concise representation of a document [Rubayyi, Alghamdi., & Khalid, Alfalqi., 2015].

4.4 Text pre-processing Techniques

Text Pre-processing is one of the key problems in the process of text classification, as the text contains several syntactic features that may not be useful for analysis. So the text is needed to be subjected to certain techniques for its cleaning. The applied techniques are tokenization, filtering, and stemming. These techniques are useful as they increase the performance and the accuracy of the later classification system significantly. In the following we briefly describe them.

Tokenization: Tokenization is the process of splitting up a string into units called tokens each of which is either a word or number or punctuation mark [Christopher,Manning., and Hinrich, Schiitze., 1999].

Filtering: A common filtering is stop-words removal. Stop words are the words frequently appear in the text without having much content information (e.g. prepositions, conjunctions, pronouns). Examples of stop-words are: a, about, an, are, as, at, be, by, for, from, how, in, is, of, on, or, that, the, these, this, too, also, was, what, when, where, who, will, etc.

Stemming: The stemming is a technique to detect different inflections and derivations of morphological variants of words in order to reduce them to one particular root called stem, namely, to the original root form. A stem is the portion of a word that is left after removing its prefixes and suffixes. For instance, the reduction of “liked” to “lik” and “like” to “lik” [Wahiba Ben Abdessalem Karaa, 2013].

4.4.1 Feature Weighting [Ahmad, Mazyad., et al.,2017]

After the process of text preprocessing has been applied, the next action is to represent the text documents in a suitable format that can be recognized by the classifier. The most common way of representation is called "Vector Space Model"(VSM). In particular, in VSM each text document is represented as a vector of index terms in which each term is associated with a weight (score) that measure how informative/discriminative the correspondent term is. The method which assigns a weight to a term is called Term Weighting Scheme (TWS). There are two main term weighting schemes: the Term Frequency (TF) and Term Frequency - Inverse Document Frequency (TF-IDF).

Term Frequency: TF is the simplest approach to assign the weight to be equal to the number of occurrences of term t in document d.

Term-Inverse document frequency: Tf-Idf is a statistical metric used to measure how important a term is across a set of documents and it is calculated as the product of tf and inverse document frequency (idf). Idf measures how informative a word is, namely, if the term is common or rare across all documents. Idf is calculated as follows:

$$idf_i = \log \frac{|D|}{1 + |\{d : t_i \in d\}|}$$

where $|D|$ is the total number of documents and $|\{d : t_i \in d\}|$ is the number of documents that contains the term t . Then, *tf-idf* will be given by: $(tf - idf)_{i,j} = tf_{i,j} \times idf_i$.

A high weight in *tf-idf* is reached by a high term frequency in the given document and a low document frequency of the term in the whole collection of documents. The *tf-idf* value for a term will always be greater than or equal to zero.

4.5 Evaluation Metrics

This chapter describes measures and methods to evaluate the performance of classifiers.

4.5.1 Cross-validation

The Cross-validation sometimes called rotation estimation is probably the most common model evaluation technique used to assess how well the model generalizes to new independent data [Wikipedia. Cross-validation statistics]. For a machine learning problem, the data is divided into two segments: One training set which is used to train or learn the model and one test set which is used to evaluate the accuracy/performance of the model. The basic form of cross-validation is k-fold Cross-Validation, where the data are first partitioned into *k* equally sized segments or folds and then k iterations of training and validation are performed so that into each iteration a different fold of the data is held out for validation while the remaining *k-1* folds are used for learning [Payam, Refaeilzadeh., et al., 2009]. The process is repeated *k* times, meaning that each sequement will be used as a test set once. The results from each fold can then be summed together into a final estimation. The widely used k-fold Cross-validation is the 10 fold Cross-validation (k=10) where the dataset is randomly divided into 10 equal sized subsamples. Of the 10 subsamples, 9 subsamples are used as training data, and the remaining one subsample is used as the validation data for testing the model

[Ziheng,Wang., et al.,2019]. The cross-validation process is then repeated for10 times, with each of the 10 subsamples used once as the validation data. The process results in 10 estimates which then are averaged to produce a single estimation.

4.5.2 Confusion Matrix

A confusion matrix also known as a contingency table or an error matrix, is a specific table layout that allows visualization of the performance of an algorithm. In information retrieval system, the format of a confusion matrix is as follows [Christopher,D., Manning., et al.,2008]:

Table 4.1: The confusion matrix for information retrieval.

	Relevant	Non-relevant
Retrieved	true positives(tp)	false positives(fp)
Not retrieved	false negatives(fn)	true negatives(tn)

The two most frequent and basic measures for information retrieval effectiveness are precision and recall. Precision is the measure of exactness or fidelity, whereas Recall is a measure of completeness. In particular, Precision (P) is the fraction of retrieved documents that are relevant and can be expressed by dividing the number of relevant documents by the total retrieved items.

$$\Pr ecision = \frac{\#\left(relevant\, items\, retrieved\right)}{\#(retrieved\, items)} = P(relevant \mid retrieved)$$

Recall (R) is the fraction of relevant documents that are retrieved and can be expressed by dividing the retrieved relevant items by the total existing relevant items.

$$\operatorname{Re} call = \frac{\#\left(relevant\, items\, retrieved\right)}{\#(relevant\, items)} = P(retrieved \mid relevant)$$

These notions can be made clear by examining the Table 3. Precision and Recall can also be expressed by Equations follows:

$$\Pr ecision = \frac{tp}{(tp + fp)}$$

$$\operatorname{Re} call = \frac{tp}{(tp + fn)}$$

Given the numbers from the confusion matrix, several performance measures can be calculated, such as:

$$Accuracy = \frac{correctly\,identified\,documents}{all\,documents} = \frac{tp + tn}{tp + fp + fn + tn}$$

$$TNRate = Specificity = \frac{tn}{fp + tn}$$

$$FPRate = 1 - Specificity = \frac{fp}{fp + tn}$$

$$kappa = \frac{accuracy - \left[\frac{\left(tn + fp\right)(tn + fn) + (fn + tp)(fp + tp)}{(tp + tn + fp + fn)^2} \right]}{1 - \left[\frac{\left(tn + fp\right)(tn + fn) + (fn + tp)(fp + tp)}{(tp + tn + fp + fn)^2} \right]}$$

4.5.3 F1 score

F1 score also called F-score or F-measure is a measure of a test's accuracy and is defined as the weighted harmonic mean of the precision and recall of the test [wikipedia. F1 score]. The formula is: $F = 2 \cdot \left(\frac{precision \cdot recall}{precision + recall} \right)$.

4.5.4 Area Under the Curve

AUC provides an aggregate measure of performance across all possible classification thresholds [Machine Learning]. The range of possible AUC values is 0.5 to 1, where a perfect classifier has AUC = 1 and a completely random classifier, that is, a bad classifier has AUC = 0.5.

Chapter 5 Experiments and Results

This chapter presents the principal findings of the research by explaining them with the experimental approaches. The findings can be divided into two sections. The first section presents an exploratory data analysis, aiming to obtain a general view of the data, while the second section presents the text mining tasks such as text classification and text clustering aiming to extract meaningful insights. The instrumentation that was utilized in the research is the programming language R. For the experiments, the version was employed is R 3.4.3. In carrying out of the experiments was used an Intel Core i5-7200 2.50GHz-2.70GHz processor with 8GB of RAM running Windows 10.

5.1 Dataset Representation

The chapter performs a data exploration which involves visualizing and summarizing the data considering that it is vitally important to obtain further information on the dataset.

5.1.1 Understanding and summarizing of lang attribute

Table 5.1 depicts the top 10 languages, ordered by decreasing number of tweets. The top 10 languages accounted for 97% of all the tweets. In Table 5.1, it is apparent that the most commonly occurring languages are English, Spanish and British English, with the English language constituting 82.8% of the total. The large percentage is confirmed by the fact that we had a limitation of downloading tweets only in the English language. In the dataset, 38 languages were found in the 16800 tweets. In addition, Table 5.1 shows the distribution of users by the interface languages they choose. The anticipated evidence is that the English language dominates with 10740 users constituting 84% of the total. Moreover, the last column shows the number of tweets per the number of users of each language separately.

Table 5.1: Distribution of the top 10 languages in Twitter.

Lang	Tweets	%	Users	Tweets/user
en-English	13923	82.875	10740	1
es-Spanish	793	4.720	527	1
en-GB-British English (United Kingdom)	338	2.011	280	1
it-Italian	294	1.75	112	3
fr-French	292	1.738	217	1
pt-Portuguese	233	1.386	220	1
de-German	168	1	117	1
ru-Russian	153	0.910	99	1
ja-Japanese	136	0.809	103	1
nl-Dutch	82	0.488	51	2

5.1.2 Understanding and summarizing of time_zone attribute

With regard to Table 5.2, what applies to all tweets is that most of them were declared without a time zone, that is, 6272 tweets are empty, less than 38% of the total. The fields without a time zone probably due to both the lack of explicit user consent and the unavailability of localization services at the moment in which tweets are published. The remaining 62% include time-zone information, where 25% include a city name, such as London, whilst 37% include the name of time zone, such as Pacific Time. In the dataset, 140 time zones were found in the 16800 tweets. Table 5.2 also shows the distribution of users by the time zone they choose. Most users do not choose a time zone, that is, 5195 users that constitute 48% of the total. The cities with the most users are: London that constitutes 6% of the total, Quito constitutes 1.9%, Amsterdam constitutes 1.8%, Rome constitutes 0.3% and Casablanca constitutes 0.69%.

Table 5.2: Distribution of the 10 top time zones in Twitter.

Time zone	Tweets	%	Users
None	6272	37.366	5195
Pacific Time(US & Canada)	3286	19.577	2221
Eastern Time(US & Canada)	1614	9.615	1157
Central Time(Us & Canada)	906	5.397	673
London	771	4.593	652
Quito	345	2.055	207
Atlantic Time(Canada)	261	1.554	249
Amsterdam	232	1.382	201
Rome	184	1.096	36
Casablanca	177	1.054	75

5.1.3 Understanding, summarizing and visualizing of continuous attributes

Online social networks such as Twitter, Facebook, are now a popular way for users to connect, communicate, and share content. Due to the massive popularity of these sites, data about the users and their communication offers unprecedented opportunities to examine how human society functions at scale. In the Twitter blogosphere, data important to systematically investigate are the profile characteristics of users, such as, the number of followers, friends, favorites. The number of followers is probably the most basic and succinct quantity for measuring the popularity of users, namely, it is an indicator of how active a user is where users with many followers are likely to be more active on Twitter. In addition, features that indicate a substantial measure of activeness on Twitter is the statuses, favorites, retweets, favorites.

Statistics is a branch of mathematics that deals with the collection, organization, analysis, interpretation, and presentation of data. Specifically, within statistics, there is a main category, the descriptive statistics, which is a subset of mathematical statistics that is invaluable for summarizing, consolidating, and describing research data in a logical, meaningful, and efficient way. Descriptive statistics are broken down into three major categories that help compare the data: the Measures of Location, the Measures of Dispersion and the Measures of Shapes. The Measures of Location are used in this thesis. Specifically, the measures of location can broadly be defined as the measures that portray the central location of varying data values.

Table 5.3 depicts the descriptive statistical measures of the attributes regarding the information on Twitter users of our dataset. Examining the users' information, it appears that the median number of the users' tweets is 6670 and the mode is 4, which means that 6670 users posted a tweet four times. In addition, the median count of the users' followers is 531 with the mode equal to 0, which means that 531 users are not followed by other users, while the median count of the users' favourites is 1445 and the mode is 0, which means that 1445 users that did not do a "like" in a tweet. From Table 5.3 also seems that the median count of the users' friends is 484 and the mode equal to 0, which means that 484 users do not follow anyone, while the median count of the users' retweets is 83 with the mode equal to 1, meaning that 83 users retweeted a tweet. Furthermore, the median count of the users' favorites is 90 and the mode is 0, which means that in 90 users did not like tweets.

Table 5.3: Descriptive statistical measures of the attributes.

Measures of Location					
	Min.	**Median**	**Mean**	**Max.**	**Mode**
Followers_count	0	531	11940.13	28715815	0
Friends_count	0	484	4090.18	5430083	0
Favourites_count	0	1445	10428.19	3555686	0
Statuses_count	1	6670	78040.9	164504281	4
Retweet_count	1	83	1951.56	76915	1
Favorite_count	0	90	2834.23	139460	0

In addition, all the attributes are tabulated in Table 5.3 are skewed to the right, which means that there are few users with very high numbers of followers, friends, etc, whilst most have few numbers of followers, friends, etc in the dataset. The plots below also confirm this ascertainment. In particular, the distributions are characterized as right-skewed distributions and the box plots as a positive skew box plot, where, as a result of those characteristics the plots have their mean greater than the median. Those box plots do not include extreme values.

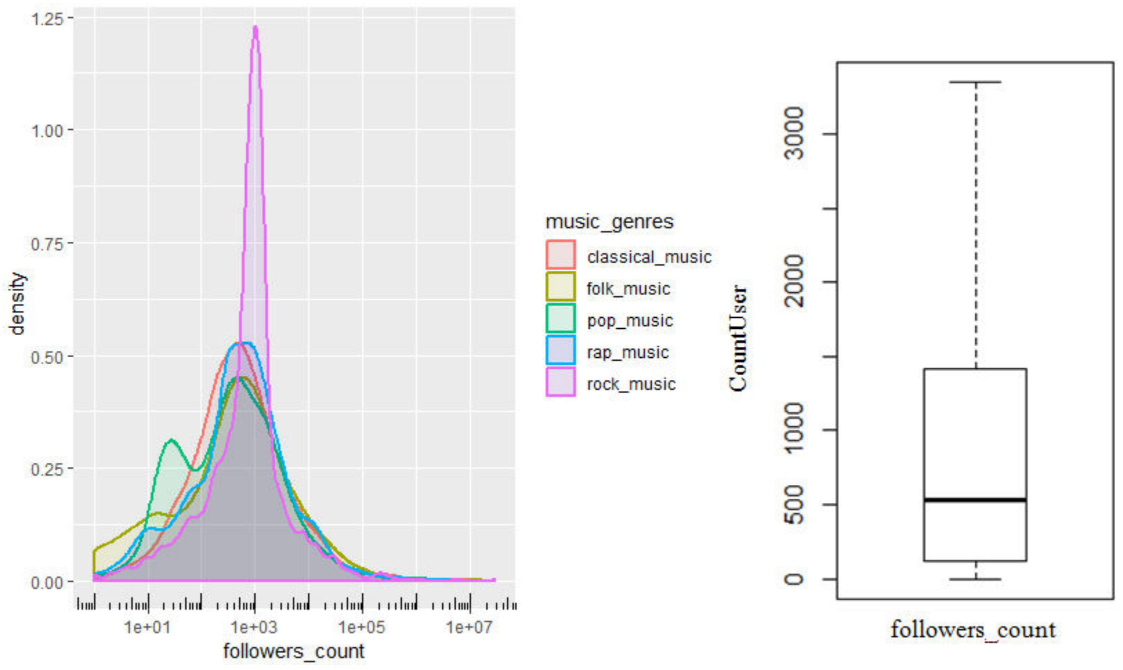


Figure 5.1a: Distributions of followers_count.

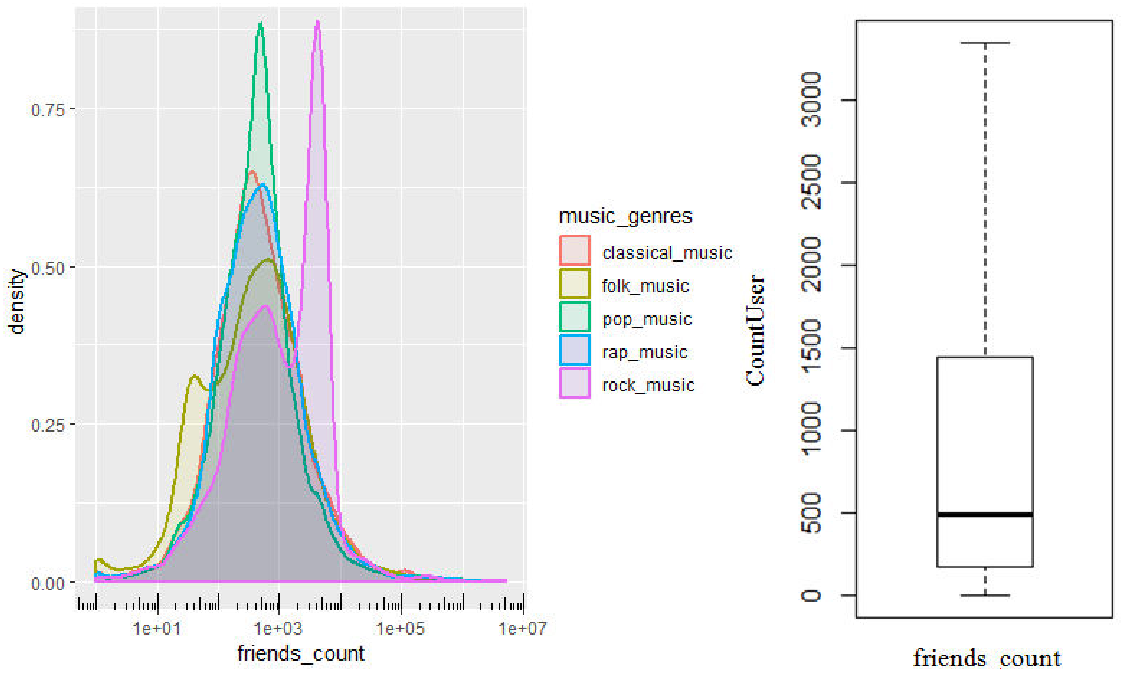


Figure 5.1b: Distributions of friends_count.

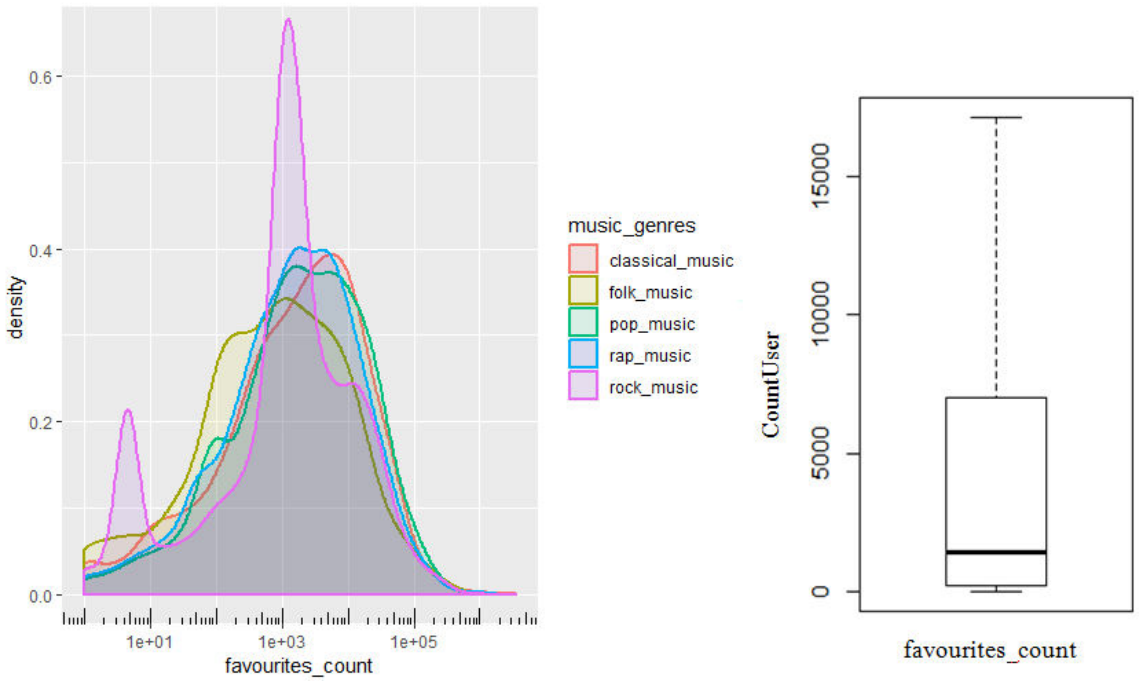


Figure 5.1c: Distributions of favourites_count.

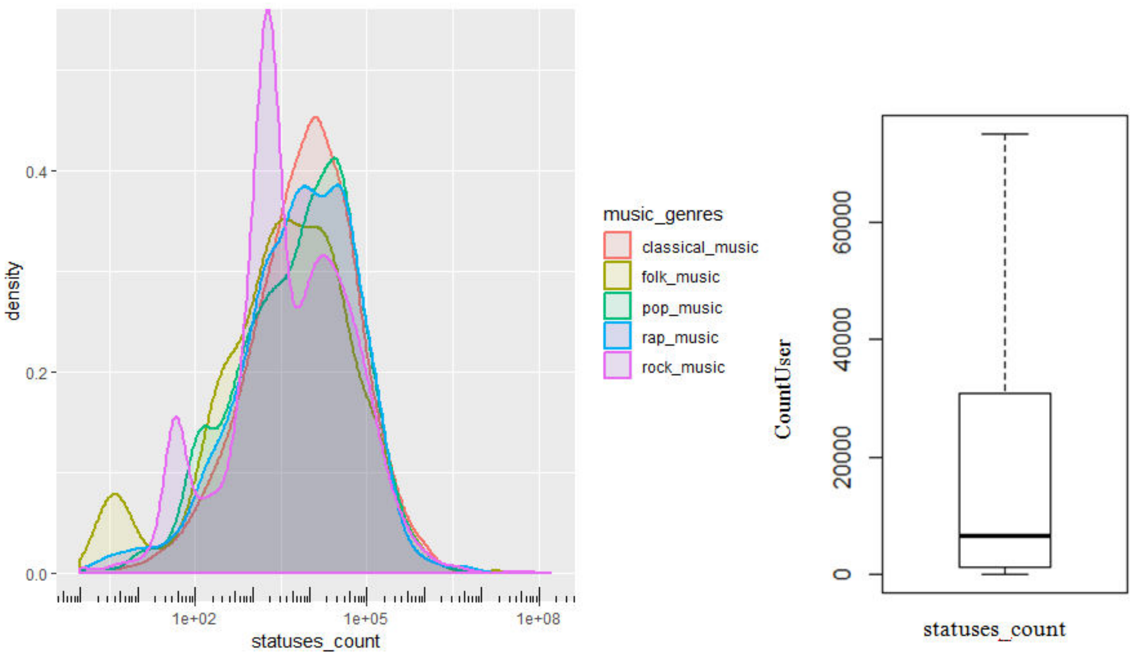


Figure 5.1d: Distribution of statuses_count.

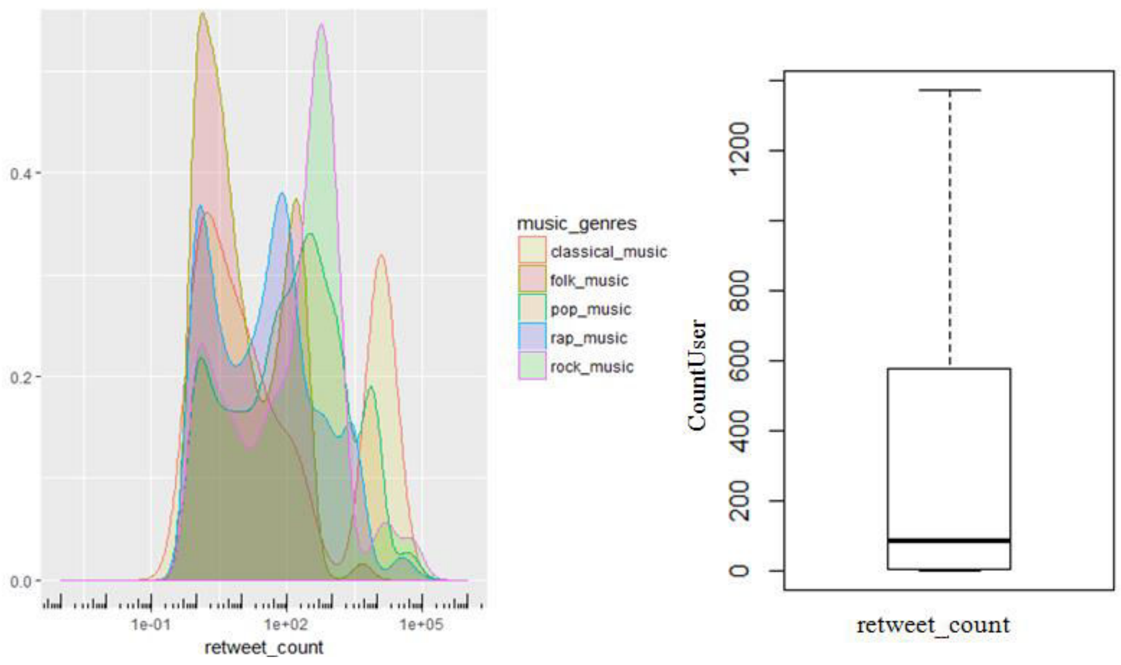


Figure 5.1e: Distributions of retweet_count.

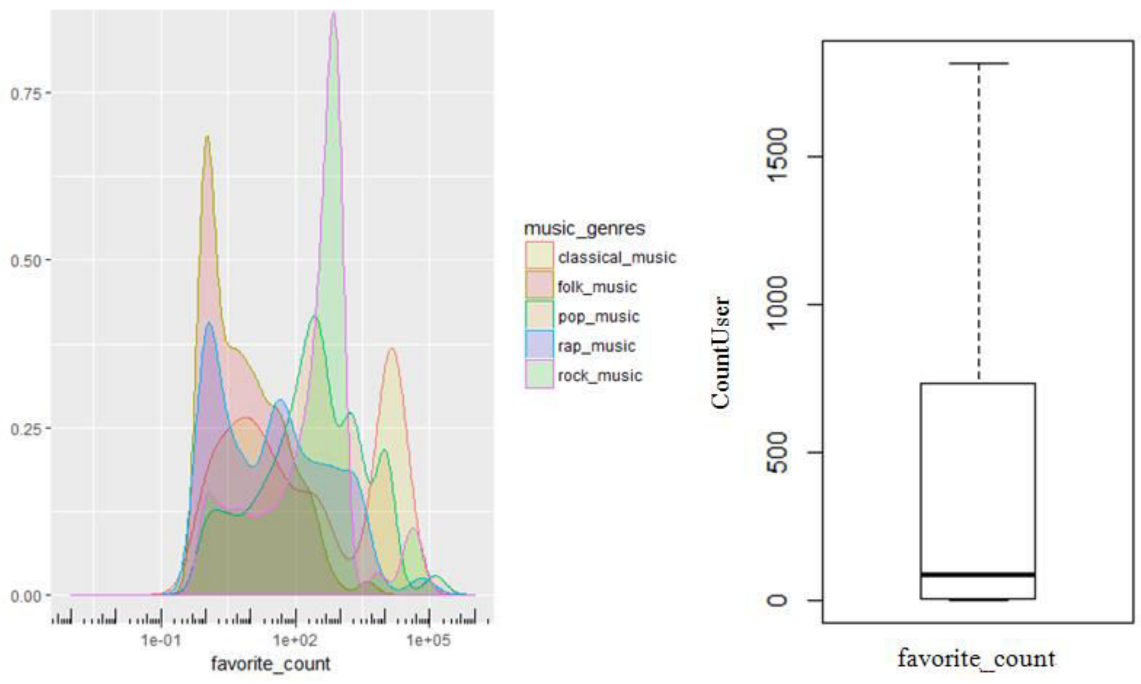


Figure 5.1f: Distributions of favorite_count.

5.2 Number of followers, friends and statuses of the users on the dataset

An expected question that can be raised is, how many followers do users have? Every tweet has some impact for the user, and hence the number of followers is of great importance to every user with regard to the direct, faster and greater transmission of information for each tweet. In the current dataset, there are 12781 users (unique users) with an average of 6.10 tweets per user, where 2.75%, namely 351 users are no followed by other users as they have 0 followers. 48.82% of Twitter users have less than 500 followers, 16.76% have less than 1000, 24.24% have less than 5000,4.04% have less than 10000, 2.67% have less than 20000 and 3.45% have more than 20000. Figure 5.2

illustrates the distribution of Twitter users' followers. In addition, Table 5.4 reports the top 5 users with the most followers.

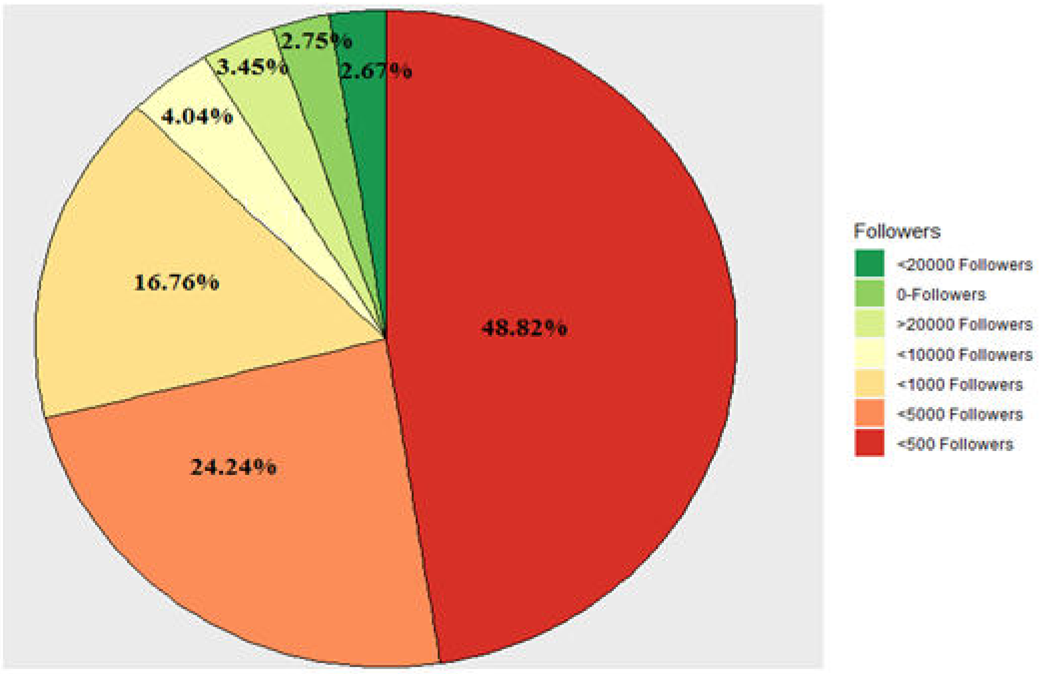


Figure 5.2: Distribution of users by follower count.

Table 5.4:Users with most followers.

Nickname Of user	No.of Followers
User1	28715815
User2	12586176
User3	8447507
User4	5446443
User5	5102585

In the case of number of friends that users have, 1.48%, that is 190 users have 0 friends. 14.86% of Twitter users have less than 500 friends, 17.44% have less than 1000, 25.84% have less than 5000, 2.80% have less than 10000,1.34% have less than 20000 and 1.54% have more than 20000. Figure 5.3 illustrates the distribution of Twitter users' friends. In addition, Table 5.5 reports the top 5 users with the most friends.

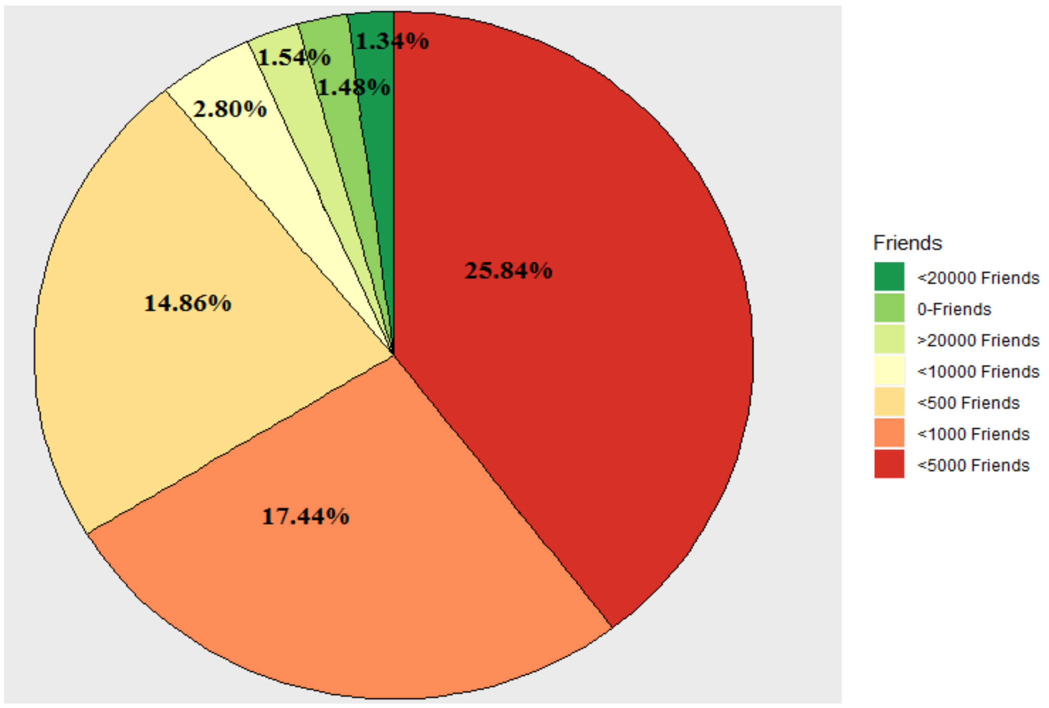


Figure 5.3: Distribution of users by friend count.

Table 5.5: Users with most friends.

Nickname Of user	No.of Friends
User1	5430083
User2	4601143
User3	2504270
User4	1442901
User5	1422981

As regards Figure 5.4, this depicts the distribution of Twitter users by the number of tweets they have sent. The numbers that are shown are quite astounding as well about 100% of all Twitter users have tweeted at least once. 15.10% of Twitter users have made less than 500 tweets, 6.63%% have made less than 1000 tweets, 24.02% have made less than 5000 tweets, 10.57% have made less than 10000 tweets, 11.62% have made less than 20000 tweets and 32.03% have made over than 20000 tweets.

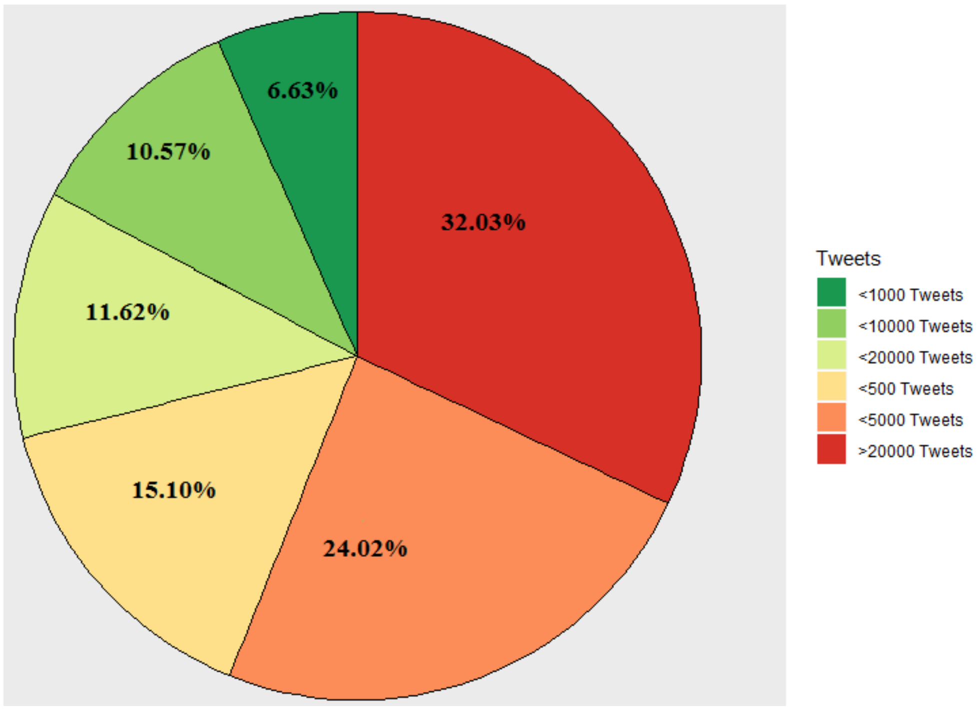


Figure 5.4: Distribution of users by statuses count.

It is apparent that in the vast majority of users, 48.82%, that is 6239 users, have less than 500 followers, 25.84%, that is 3302 users, have less than 5000 friends and 32.03%, that is 4094 users, have posted over 20000 tweets. Thus, it is concluded that a high number of users have few followers and friends, in contrast to the tweets they post. Furthermore, according to the data, the average Twitter user has 1,070 followers 3,124 friends and 6,106 tweets.

5.3 Visually overview of a text

Useful information can be extracted and depicted from the text attribute of the dataset. Thus, using the wordcloud() function the frequency of the words was counted and depicted. The creating of the word clouds are presented in the code of Chapter 5.5.1 (lines:32-38). The word cloud of all music genres is illustrated in Figure 5.5 and the most used words in each music genre are illustrated in Figure 5.6. The word cloud of all music genres clearly shows that ″music″ is the top word with the ″classical″, ″pop″, ″rap″, ″rock″, and ″folk″ words to follow. So the tweets present information on music, classical, pop, rap, rock and folk. Some other important words are ″love″, ″sad″, ″good″, ″great″, ″happy″, and ″like″, which are both positive and negative sentiments of the users. Another set of frequent words, ″reggae″, ″indie″, ″alternative″, ″jazz″ and ″metal″ are tweets about other different music genres. Some tweets characterize the music genres such as ″songs″, ″live″, ″artist″, ″band″, and ″tribute″. Finally, it is obvious that the word ″music″ must have appeared in all music genres.

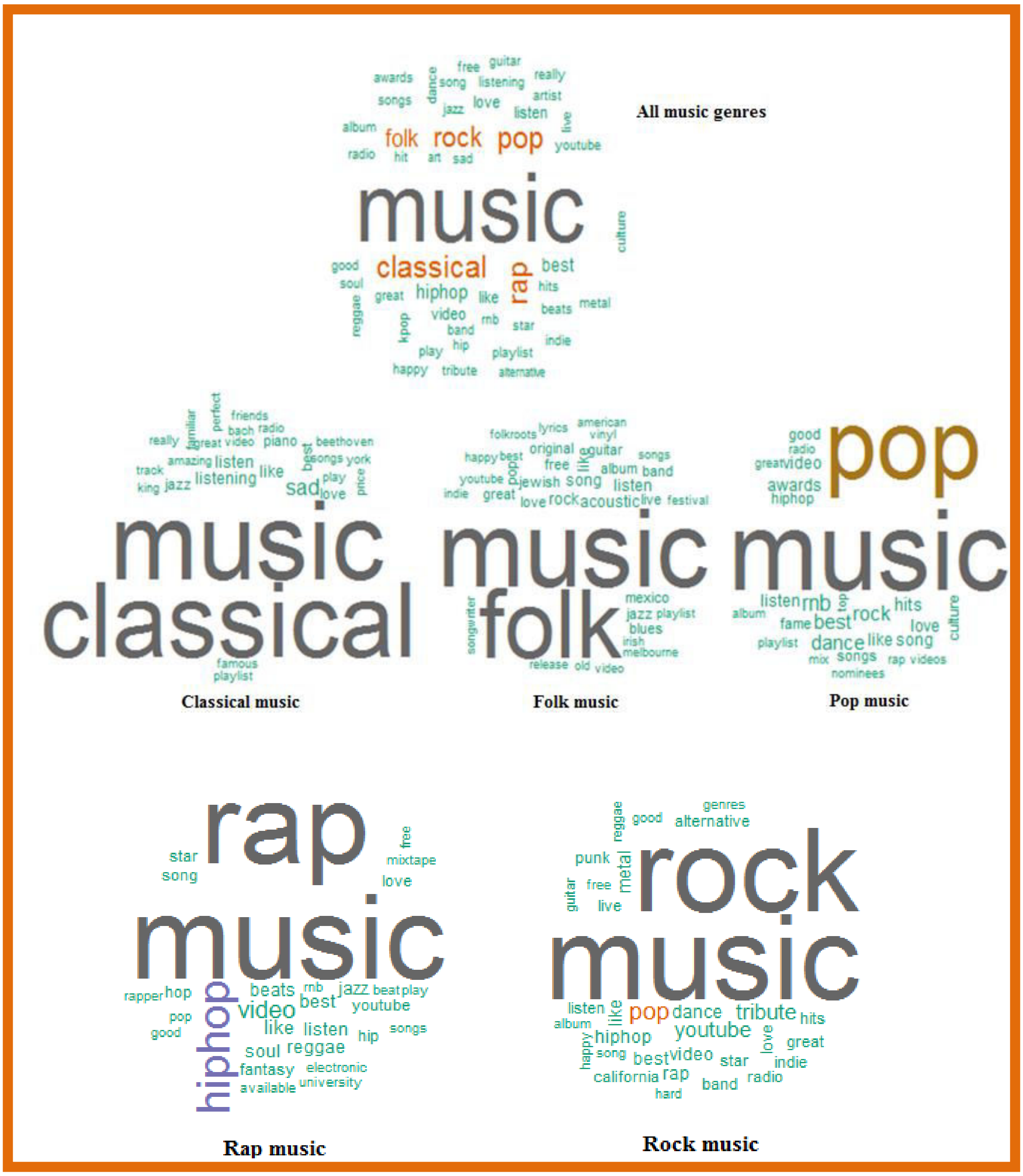


Figure 5.5: Word clouds of all music genres.

All music genres

Word	Frequency
music	9950
classical	2356
pop	2256
rap	2211
rock	2001
folk	1350

Classical music

Word	Frequency
music	2428
classical	2304
sad	252
like	129
listen	119

Folk music

Word	Frequency
music	1456
folk	1276
acoustic	84
rock	82
song	78

Pop music

Word	Frequency
music	2162
pop	1870
rnb	228
best	211
rock	206

Rap music

Word	Frequency
music	2070
rap	2008
hiphop	665
video	256
best	163

Rock music

Word	Frequency
music	1637
rock	1607
pop	211
youtube	152
tribute	145

Figure 5.6: Frequent terms for all music genres.

An approach for finding associations for a given term, which is a further form of count-based evaluation methods, can be realized with the findAssocs() function (line 42, Chapter 5.5.1), that computes associations between terms in a Term Document Matrix or Document Term Matrix. Table5.7 presents the results of applying the function on frequently occurring terms, with at least 0.01 correlation, that is a minimum threshold which has been set. From Table 5.7, it is apparent that the term "song" is highly associated with the term "songs", which demonstrates a strong pairwise term association. In particular, the term "songs" was found in 10% of all the tweets

.

Table 5.7: Words associations.

album													
the	**folk**	**music**	**new**	**best**	**good**	**this**	**dance**	**just**					
0.08	0.04	0.04	0.04	0.03	0.03	0.02	0.01	0.01					
song													
songs	**rnb**	**love**	**pop**	**jazz**	**folk**	**hiphop**	**new**	**year**	**dance**	**good**	**listen**	**rap**	**this**
0.10	0.09	0.08	0.08	0.07	0.06	0.05	0.05	0.05	0.02	0.02	0.02	0.01	0.01
great													
will	**songs**	**make**	**live**	**year**	**time**	**band**	**folk**	**this**					
0.08	0.06	0.04	0.03	0.03	0.02	0.01	0.01	0.01					

5.4 Implementations of the text mining tasks with the algorithms

The chapter presents the implementations of the algorithms in the two text mining tasks the text classification and text clustering. Those tasks were implemented with tm package of R programming language. The tm package provides a framework for flexible integration of premier statistical methods from R, it interfaces to well known open source text mining infrastructure and methods and it has a sophisticated modularized extension mechanism for text mining purposes [Ingo, Feinerer et al., 2008]. Detailed description of the tm package is included in Chapter 3.4.3.

5.4.1 Text Classification with Naïve Bayes algorithm

The chapter presents the implementation of the Naïve Bayes algorithm. The algorithm utilizes the text mining framework of R provided by the *tm* package.

```r
#Load the necessary libraries
library(tm)
library(wordcloud)
library(Rgraphviz)
library(e1071)
library(caret)
library(SnowballC)
#Step 1-Data preparation- Importing of the file
df1<- read.csv("C:/Users/pc/Desktop/tweetsOfTexts.csv",sep=";",header=TRUE,
    stringsAsFactor=FALSE)
attach(df1)
names(df1)
str(df1)
#Shuffle rows in the dataframe
set.seed(1986)
df1<-df1[sample(nrow(df1)),]
str(df)
df1$music_genres<-as.factor(df1$music_genres)
str(df1)
```

```r
#Step 2-Data Preparation- Create the corpus- Preprocessing of the text
 xCorpus<-VCorpus(VectorSource(df1$text))
 clean.corpus <- tm_map(xCorpus,content_transformer(tolower))
 clean.corpus <- tm_map(clean.corpus, removeNumbers)
myStopList=read.table("C:/Users/pc/Desktop/stop.txt",header=FALSE,sep="\n",strip.white=TRUE)
 clean.corpus<-tm_map(clean.corpus,removeWords,myStopList[,1])
 clean.corpus<-tm_map(clean.corpus,removeWords,stopwords("english"))
removeUsersNames<-(function(x)gsub("@\\w+", "", x))
clean.corpus<- tm_map(clean.corpus,content_transformer(removeUsersNames))
removeURL0<- (function(x) gsub("(f|ht)tp(s?)://\\S+", "", x, perl=T))
clean.corpus<-tm_map(clean.corpus,content_transformer(removeURL0))
removeStrangeCharac<- (function(x) gsub("[^[:alpha:][:space:]]*","",x))
clean.corpus<-
tm_map(clean.corpus,content_transformer(removeStrangeCharac), lazy = TRUE)
clean.corpus <- tm_map(clean.corpus, removePunctuation)
clean.corpus <- tm_map(clean.corpus, stemDocument)
clean.corpus <- tm_map(clean.corpus, stripWhitespace)
#Step 3- Data preparation - Create the DTM
  clean.corpus.dtm <- DocumentTermMatrix(clean.corpus)
#Visualizing textual data- Create the WordClouds
```

```
   #dtm<-TermDocumentMatrix(clean.corpus)
   #dtmMat<-removeSparseTerms(dtm,sparse=0.985)
   #dtm_matrix<-as.matrix(dtmMat)
   #v<-sort(rowSums(dtm_matrix),decreasing=TRUE)
  #d<-data.frame(word = names(v),freq=v)
   #head(d, 6)
  #wordcloud(words=d$word,freq=d$freq,min.freq=1,random.order=FALSE,
#colors=brewer.pal(8, "Dark2"),scale = c(5, 0.5))
#Word association
#dtm<-TermDocumentMatrix(clean.corpus)
#dtmMat<-removeSparseTerms(dtm,sparse=0.97)
#findFreqTerms(dtmMat, lowfreq=15)
#Find words associations
#findAssocs(dtmMat,"song", corlimit=0.01)
# Step 4-Data preparation- Creating training and test datasets
df1.train<-df1[1:7937,]
df1.test<-df1[7938:10583,]
dtm.train<-clean.corpus.dtm[1:7937,]
dtm.test<-clean.corpus.dtm[7938:10583,]
clean.corpus.train<-clean.corpus[1:7937]
clean.corpus.test<-clean.corpus[7938:10583]
```

```
#Step 5- Data preparation-creating indicator features for frequent words
dim(dtm.train)
frequent_terms <- findFreqTerms(dtm.train,5)
dtm.train.nb<-DocumentTermMatrix(clean.corpus.train,control=list(dictionary=
frequent_terms))
dtm.test.nb<-DocumentTermMatrix(clean.corpus.test,control=list(dictionary=
frequent_terms))
#Function to convert the word frequencies to yes (presence) and no (absence) labels
 convert_count<-function(x){
 y<-ifelse(x>0,1,0)
y<-factor(y,levels=c(0,1),labels=c("No","Yes"))
y
}
trainNB<-apply(dtm.train.nb,MARGIN=2,convert_count)
testNB<-apply(dtm.test.nb,MARGIN=2,convert_count)
#Constructing model and making prediction
naiveBayesModel<-
train(trainNB,df1.train$music_genres,method=”nb”,trControl=trainControl(method
=”cv”,number=10))
naiveBayes.pred <- predict(naiveBayesModel, newdata=testNB)
#Creating of Confusion Matrix
naiveBayesConfMat<-confusionMatrix(naiveBayes.pred, df1.test$music_genres)
```

After loading the necessary libraries (lines: 1-6), the first step is to import the textual data into R. The file of textual data is loaded from CSV format into the R in the form of a data frame (line: 7). Then, in order to have a random order in the data frame, the rows are shuffled, so that there is an equal number of tweets in both sets in the splitting of the training set with the test set (lines: 11-12). The action that follows is the preprocessing of textual data that includes creating a corpus, the VCorpus() function is used for this purpose (line: 16). For the preprocessing of textual data, a series of transformations are applied, which are done via the tm_map() function. (lines: 17-30). Table 5.8 indicates examples from the content of the text corpus before and after the cleaning process.

Table 5.8: The textual data before and after the cleaning process.

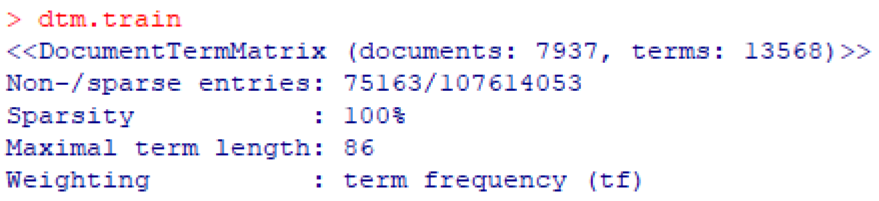
Text before cleaning	Text after cleaning
I always enjoy discovering new music that ain't pop	always enjoy discovering new music aint pop
love this song https://t.co/yfSKnmUQjH	love song
@IAmLindsayJones the best classical music	best classical music

After the corpus is pre-processed and cleaned up, it has to be transformed into a matrix called document term matrix, the DocumentTermMatrix() function is used for this purpose(line: 31). Table 5.9 represents a portion of the Document Term Matrix of this project in which the value of 0 in the matrix is considered a sparse entry whereas a nonzero value is a nonsparse entry.

Table 5.9: The depiction of a portion of the DTM.

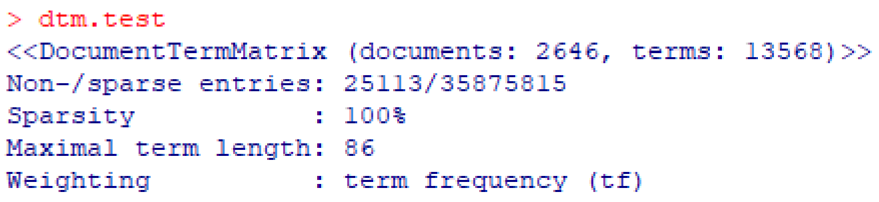
Terms					
Docs	**listening**	**love**	**music**	**play**	**pop**
550	1	0	2	1	1
551	0	0	1	0	1
552	0	0	1	0	1
553	0	0	1	0	1
554	0	0	1	0	1
555	0	1	1	0	1
556	0	0	1	0	1
557	0	0	0	0	0
558	0	0	0	0	0

The step that follows after the creating of the DTM, is the splitting of the training and test dataset (lines: 43-48). Thus, the data is divided into two portions: 75 percent for training and 25 percent for testing, namely the first 7937 tweets are used for training and the remaining 2646 for testing. Figure 5.7 and Figure 5.8 indicate the statistics of the train set and test set of DTM of the Naïve Bayes algorithm.



```
> dtm.train
<<DocumentTermMatrix (documents: 7937, terms: 13568)>>
Non-/sparse entries: 75163/107614053
Sparsity           : 100%
Maximal term length: 86
Weighting          : term frequency (tf)
```

Figure 5.7: The train set of DTM for Naïve Bayes algorithm.



```
> dtm.test
<<DocumentTermMatrix (documents: 2646, terms: 13568)>>
Non-/sparse entries: 25113/35875815
Sparsity           : 100%
Maximal term length: 86
Weighting          : term frequency (tf)
```

Figure 5.8: The test set of DTM for Naïve Bayes algorithm.

As it can be seen from Figure 5.8 the DTM includes 13568 features, which are not all useful for classification. In order to reduce the number of features, any words that appear in less than five texts in the training dataset are eliminated (line: 50). The findFreqTerms() function is used to identify the frequent words and then the DTM is restricted to use only the frequent words using the 'dictionary' option (lines: 51-52). The results of the function indicate that there are 20.31 terms appearing in at least five texts. Therefore, the training and testing datasets include 20.31 features, which correspond to words appearing in at least five texts.

Then, a variation of Naive Bayes algorithm is used known as binarized (boolean feature) Naive Bayes, in which method, the term frequencies are replaced by the Boolean yes or no features through the convert_count() function (line: 53). At this point, the training of the Naive Bayes classifier is started. The Naive Bayes implementation is in the e1071 package. In the model construction function, the classifier object contains a Naïve Bayes classifier that can be used to make predictions (line: 56). For evaluation of the classifier, its predictions must be tested on unseen data in the test data. The predict() function is used to make the predictions (line: 57).

For comparing the predicted outcomes to the true values the confusionMatrix() function is used (line: 58). The results are depicted in Table 5.10.

Table 5.10: Confusion matrix of Naïve Bayes classifier.

Predicted \ Actual	classical music	folk music	pop music	rap music	rock music	Row total
classical music	566	15	19	8	11	619
folk music	11	339	18	5	15	388
pop music	10	9	510	29	34	592
rap music	10	1	30	524	13	578
rock music	11	12	38	14	394	469
Column total	608	376	615	580	467	2626

Table 5.11 reports the evaluation results using k-fold cross-validation method with k=10. The execution time of the algorithm was 77 seconds. The model has achieved an accuracy of 88.1%, which is high accuracy. The high values appear in all evaluation metrics, which proves that the Naive Bayes classifier is effective for text classification.

Table 5.11: Evaluation Results of Naïve Bayes classifier.

CA	Precision	Recall	F1	Kappa	Specificity	AUC
0.881	0.879	0.881	0.878	0.851	0.894	0.887

Table 5.12 presents the results of recall and precision measures of each music genre for the Naïve Bayes classifier. As it is apparent, all the values of both measures are high, but in classical music it's higher. High recall and precision values indicate low false negatives and low false positives. Specifically, this means that in the classical music class most items are labeled correctly.

Table 5.12: Recall and Precision of each music genre of Naïve Bayes classifier.

Naïve Rayes

	Recall	Precision
classical_music	**0.930**	**0.914**
folk_music	0.901	0.873
pop_music	0.829	0.861
rap_music	0.903	0.906
rock_music	0.843	0.840

5.4.2 Text Classification with Support Vector Machine algorithm

The chapter presents the implementation of the Support Vector Machine algorithm. In R, the svm() function provides an interface to libsvm in the e1071 package complemented by visualization and tuning functions. Libsvm is a fast and easy-to-use implementation of the most popular SVM formulations such as, C-SVM, nu-SVM classification, and epsilon-SVM regression. It also provides the most common kernels, including linear, polynomial, RBF, and sigmoid, only extensible by changing the C++ source code. Furthermore, libsvm is used in multiclass classification, as it uses the one against one technique by fitting all binary subclassifiers and finding the correct class by a voting mechanism. Compared to the Naïve Bayes algorithm code, the SVM code remains almost the same in the most part. One difference between them is that the Naïve Bayes algorithm uses the *tf* method unlike to SVM which uses the *tf-idf*.

```
#Load the necessary libraries
library(tm)
library(e1071)
library(caret)
library(dplyr)
library(SnowballC)

#Importing of the file
df1<-read.csv("C:/Users/pc/Desktop/tweetsOfTexts.csv",sep=";",header=TRUE,
stringsAsFactor=FALSE)
attach(df1)
names(df1)
str(df1)
#Shuffle rows in the dataframe
set.seed(1986)
df1<-df1[sample(nrow(df1)),]
str(df)
df1$music_genres<-as.factor(df1$music_genres)
str(df1)
#Creating of corpus- Preprocessing of the text
xCorpus<-VCorpus(VectorSource(df1$text))
clean.corpus <- tm_map(xCorpus,content_transformer(tolower))
clean.corpus <- tm_map(clean.corpus, removeNumbers)
myStopList<-read.table("C:/Users/pc/Desktop/stop.txt",header=FALSE,sep="\n",
strip.white=TRUE)
clean.corpus<-tm_map(clean.corpus,removeWords,myStopList[,1])
clean.corpus<-tm_map(clean.corpus,removeWords,stopwords("english"))
clean.corpus <- tm_map(clean.corpus, removePunctuation)
clean.corpus <- tm_map(clean.corpus, stemDocument)
clean.corpus <- tm_map(clean.corpus, stripWhitespace)
#Splitting text documents into words
clean.corpus.dtm<-
DocumentTermMatrix(clean.corpus,control=list(weighting=function(x)weightTfIdf(
x,normalize=FALSE)))

#Creating indicator features for frequent words
frequent_terms<-findFreqTerms(clean.corpus.dtm,10)
clean.corpus.dtm2   <-   DocumentTermMatrix(clean.corpus,   list(global   =   c(2,
Inf),dictionary = frequent_terms))
#Creating training and test datasets
train_idx<-createDataPartition(df1$music_genres, p=0.75, list=FALSE)
trainSet <- df1[train_idx,]
testSet <- df1[-train_idx,]
trainSet2 <-clean.corpus[train_idx]
testSet2 <- clean.corpus[-train_idx]
#Creating indicator features for frequent words
frequent_terms2<- findFreqTerms(clean.corpus.dtm2, lowfreq=10)
#merging training and testing datasets with the dictionary of those frequent terms
trainSVM <- DocumentTermMatrix(trainSet2, list(dictionary=frequent_terms2))
testSVM <- DocumentTermMatrix(testSet2, list(dictionary=frequent_terms2))
#converts the DTM into a categorical form for modeling
```

```
convert_counts <- function(x) {
x <- ifelse(x > 0, 1, 0)
# x <- factor(x, levels = c(0, 1), labels = c("Absent", "Present"))
}
trainSVM <- trainSVM %>% apply(MARGIN=2, FUN=convert_counts)
testSVM <- testSVM %>% apply(MARGIN=2, FUN=convert_counts)
trainSVM <- as.data.frame(trainSVM)
testSVM <- as.data.frame(testSVM)
trainSVM1 <- cbind(music_genres=as.factor(trainSet$music_genres), trainSVM)
testSVM1 <- cbind(music_genres=as.factor(testSet$music_genres), testSVM)
trainSVM1<-as.data.frame(trainSVM1)
testSVM1<-as.data.frame(testSVM1)


#Constructing model and making prediction
svmModel<-svm(music_genres~.,data=trainSVM1,method=”C-
classification”,kernel=”radial”)
svm.pred <- predict(svmModel, na.omit(testSVM1))
#Creating of Confusion Matrix
svmConfMat<-confusionMatrix(svm.pred,testSVM1$music_genres)
```

Table 5.13: Confusion matrix of SVM classifier.

		Predicted					
		classical music	folk music	pop music	rap music	rock music	Row total
Actual	classical music	567	23	12	22	31	655
	folk music	7	300	4	0	2	313
	pop music	27	33	537	51	58	706
	rap music	8	10	31	500	20	559
	rock music	9	9	26	6	351	411
	Column total	618	375	610	579	462	2644

Table 5.14 reports the evaluation results using the k-fold cross-validation method with k=10 and as a kernel parameter the radial. The execution time of the algorithm was 103 seconds. Unlike to Naïve Bayes classifier, the SVM classifier has achieved an accuracy of 85.2%, which is also high accuracy. The high values appear in all evaluation metrics, which proves that the SVM classifier is also suitable for text classification.

Table 5.14: Evaluation Results of SVM classifier.

CA	Precision	Recall	F1	Kappa	Specificity	AUC
0.852	0.867	0.833	0.852	0.813	0.962	0.897

Table 5.15 presents the results of recall and precision measures of each music genre for the SVM classifier. It seems that all the values of both measures are high, but the classical music class has the highest recall and a lower precision while the folk music class has the highest precision and a lower recall. Low recall and precision values indicate many false negatives and false positives. In particular, the classical music class with low precision but high recall indicates that most of its predicted labels are incorrect

compared to the training labels while the folk music class with low recall but high precision indicates that most of its predicted labels are correct compared to the training labels.

Table 5.15: Recall and Precision of each music genre of SVM classifier.

SVM

	Recall	Precision
classical_music	**0.917**	0.865
folk_music	0.80	**0.958**
pop_music	0.880	0.760
rap_music	0.863	0.894
rock_music	0.759	0.854

5.4.3 Text Clustering with K-means algorithm

This chapter presents the implementation of K-means clustering algorithm. K-means is the most commonly used unsupervised machine learning algorithm and it is a heuristic method of constructing clusters of documents.

```
#Load the necessary libraries
library(NLP)
library(tm)
library(cluster)
library(fpc)
library(SnowballC)
library (slam)
library(philentropy)
library(factoextra)
library(ggsci)
#Importing the file
df1=read.csv("C:/Users/user/Desktop/tweetsOfTexts2.csv",sep=";",header=
 TRUE, stringsAsFactor=FALSE)
attach(df1)
names(df1)
#Creating of corpus- Preprocessing of the text
xCorpus=VCorpus(VectorSource(df1$text))
clean.corpus <- tm_map(xCorpus,content_transformer(tolower))
clean.corpus <- tm_map(clean.corpus, removeNumbers)
myStopList=read.table("C:/Users/user/Desktop/stop.txt",header=FALSE,sep="\n"
,
strip.white=TRUE)
clean.corpus<-tm_map(clean.corpus,removeWords,myStopList[,1])
clean.corpus<-tm_map(clean.corpus,removeWords,stopwords("english"))
removeUsersNames<-(function(x)gsub("@\\w+", "", x))
clean.corpus<- tm_map(clean.corpus,content_transformer(removeUsersNames))
removeURL0<- (function(x) gsub("(f|ht)tp(s?)://\\S+", "", x, perl=T))
clean.corpus<-tm_map(clean.corpus,content_transformer(removeURL0))
removeStrangeCharac<- (function(x) gsub("[^[:alpha:][:space:]]*","",x))
```

```
24   clean.corpus<-
     tm_map(clean.corpus,content_transformer(removeStrangeCharac),     lazy     =
     TRUE)
25   clean.corpus <- tm_map(clean.corpus, removePunctuation)
26   clean.corpus <- tm_map(clean.corpus, stemDocument)
27   clean.corpus <- tm_map(clean.corpus, stripWhitespace)
#Splitting text documents into words
28   dtm1<-DocumentTermMatrix(clean.corpus,control=list(weighting=weightTfIdf))
29   rowTotals<-slam::row_sums(dtm1)
30   dtm1Final<-dtm1[rowTotals>0, ]
31   dtmsFinal<-removeSparseTerms(dtm1Final,0.98)
32   matrDtmsFinal<-as.matrix(dtmsFinal)
#K-means plot with five clusters
33   dis<-distance(matrDtmsFinal,method="jaccard")
34   kfit <- kmeans(dis, centers=5,iter.max=15,nstart=50)
35   fviz_cluster(kfit,data=dtmsFinal,ellipse.type="norm",geom="point",show.clust.cen
     t=
     TRUE,ellipse.alpha=0.8,palette="uchicago")
```

Initially, K-means algorithm is going to run with five clusters, fifteen maximum iterations per run and fifty random starts (line: 34). The k = 5 is selected as it corresponds to the 5 categories of our project, as well as, we test k=6.

Consequently, the observations are partitioned into five clusters, the resulting plot of which is illustrated in Figure 5.9. The plot of clustering result is achieved with the fviz_cluster() function (line 35)and the distance metric used is Jaccard (line: 33). The fviz_cluster() is obtained based on PCA(Principal Component Analysis), which reduces the number of dimensions to two in such a way that the reduced dimensions capture as much of the variability between the clusters as possible, thus enabling better visualization.

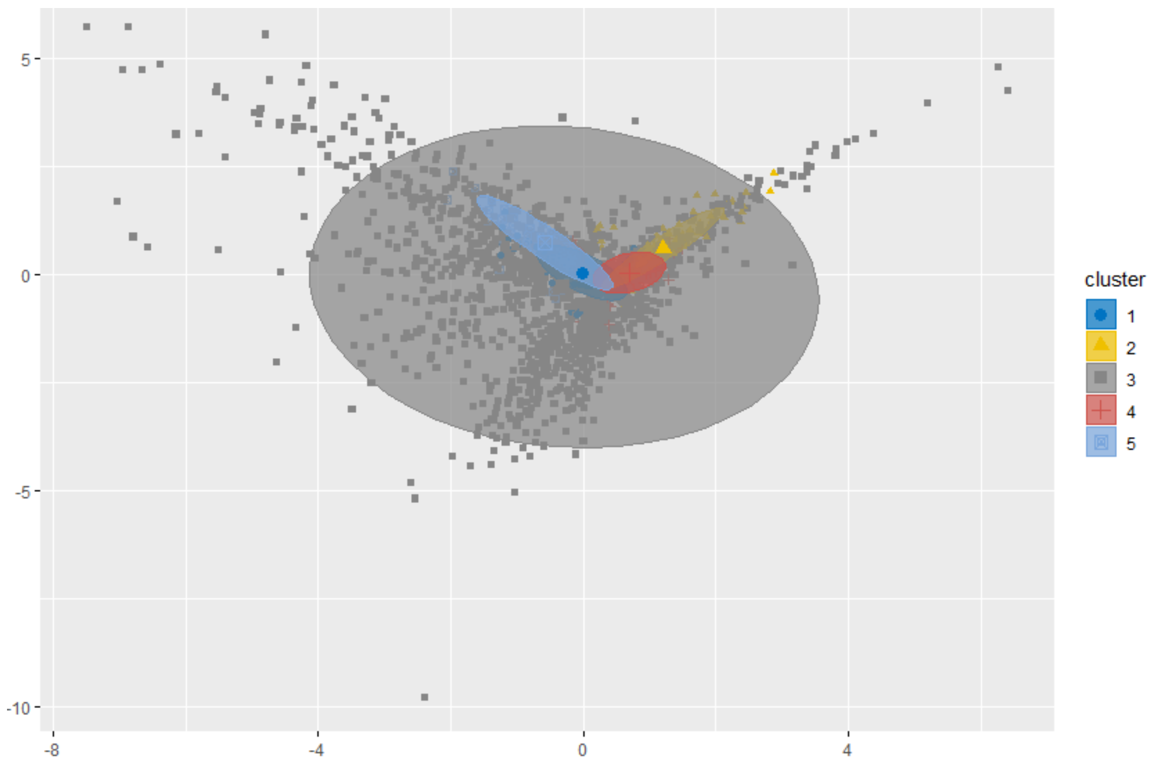


Figure 5.9: Cluster plot with k=5 and without the stemming technique.

The execution time of K-means without the stemming technique was 1707 seconds. As it appears from Figure 5.9, the algorithm groups the dataset into five clusters with each of five music genres. In particular, in cluster 1 there are 923 documents of which 37 belongs to classical music, 0 belongs to folk music, 50 belongs to rap music,630 belongs to rock music and 206 belongs to pop music, while in cluster 2 there are 1074 documents of which 834 belongs to classical music, 67 belongs to folk music,55 belongs to rap music, 52 belongs to rock music and 66 belongs to pop music. Additionally, in cluster 3 there are 6164 documents of which 1991 belongs to pop music, 1078 belongs to rock music, 2569 belongs to rap music, 556 belongs to folk music and 0 belongs to classical music. In cluster 4 there are 979 documents of which 101 belongs to classical music, 659 belongs to folk music, 50 belongs to rap music, 93 belongs to rock music and 76 belongs to pop music, while in cluster 5 there are 1399 documents of which 980 belongs to pop music, 128 belongs to rock music, 126 belongs to rap music,81 belongs to classical music and 84 belongs to folk music.

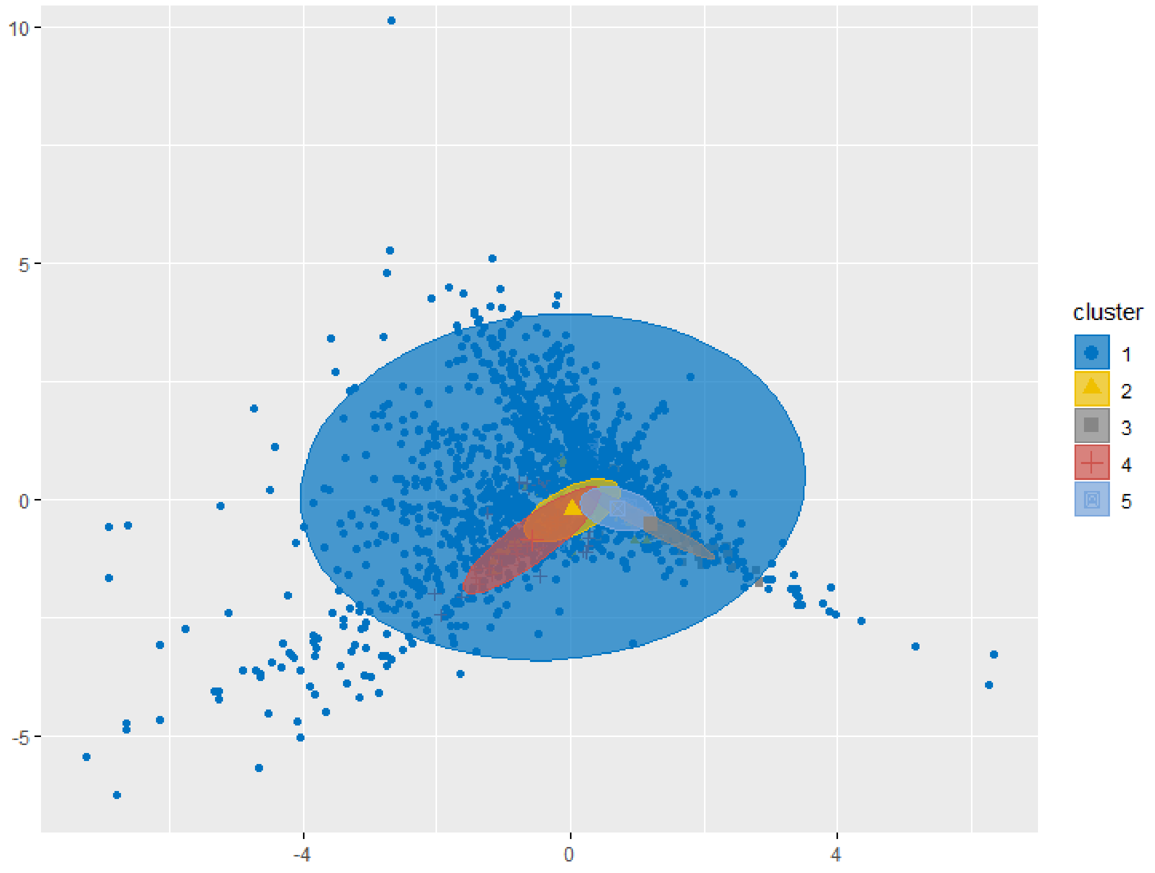


Figure 5.10: Cluster plot with k=5 and the stemming technique.

The execution time of K-means with the stemming technique was 1801 seconds. In Figure 5.10, the implementation of the algorithm returns the following results: in cluster 1 there are 6468 documents of which 1205 belongs to classical music, 622 belongs to folk music, 1972 belongs to rap music, 1120 belongs to rock music and 1449 belongs to pop music, while in cluster 2 there are 1007 documents of which 38 belongs to classical music, 36 belongs to folk music, 59 belongs to rap music, 675 belongs to rock music and 199 belongs to pop music. Furthermore, in cluster 3 there are 1299 documents of which 1055 belongs to classical music,65 belongs to folk music, 61 belongs to rap music, 56 belongs to rock music and 62 belongs to pop music. In cluster 4 there are 885 documents of which 0 belongs to classical music, 33 belongs to folk music, 77 belongs to rap music, 74 belongs to rock music and 701 belongs to pop music, while in cluster 5 there are 910 documents of which 93 belongs to classical music, 607 belongs to folk music, 48 belongs to rap music, 86 belongs to rock music and 76 belongs to pop music.

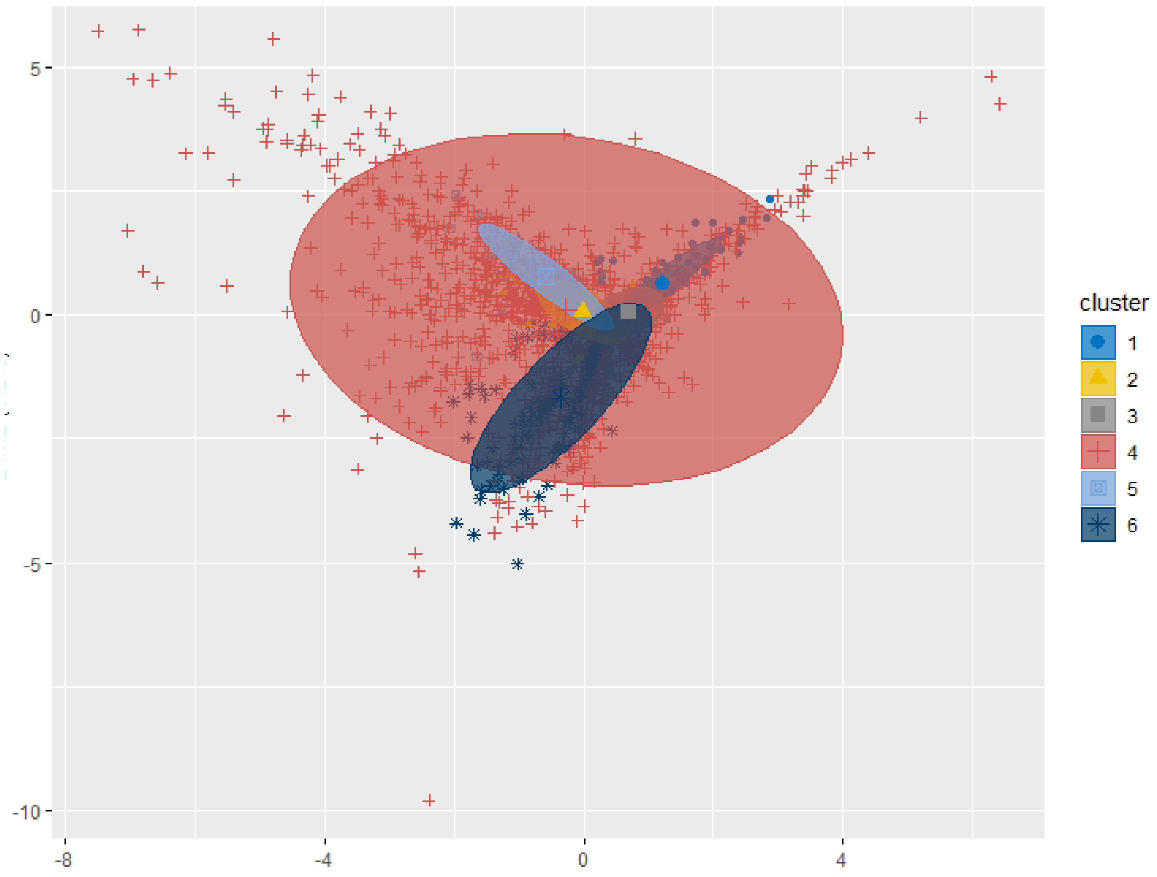


Figure 5.11: Cluster plot with k=6 and without the stemming technique.

The execution time of K-means without stemming was 1935 seconds. In Figure 5.11, the implementation of the algorithm returns the following results: in cluster 1 there are 1393 documents of which 1133 belongs to classical music, 69 belongs to folk music, 57 belongs to rap music, 59 belongs to rock music and 75 belongs to pop music, while in cluster 2 there are 1079 documents of which 41 belongs to classical music, 43 belongs to folk music, 63 belongs to rap music, 746 belongs to rock music and 186 belongs to pop music. In addition, in cluster 3 there are 980 documents of which 99 belongs to classical music, 664 belongs to folk music, 50 belongs to rap music, 91 belongs to rock music and 76 belongs to pop music, while in cluster 4 there are 4901 documents of which 1020 belongs to classical music, 525 belongs to folk music, 1056 belongs to rap music, 992 belongs to rock music and 1308 belongs to pop music. In cluster 5 there are 915 documents of which 31 belongs to classical music, 34 belongs to folk music, 75 belongs to rap music, 77 belongs to rock music and 698 belongs to pop music, while in cluster 6 there are 1301documents of which 42 belongs to classical music, 30 belongs to folk music, 1020 belongs to rap music, 70 belongs to rock music and 139 belongs to pop music.

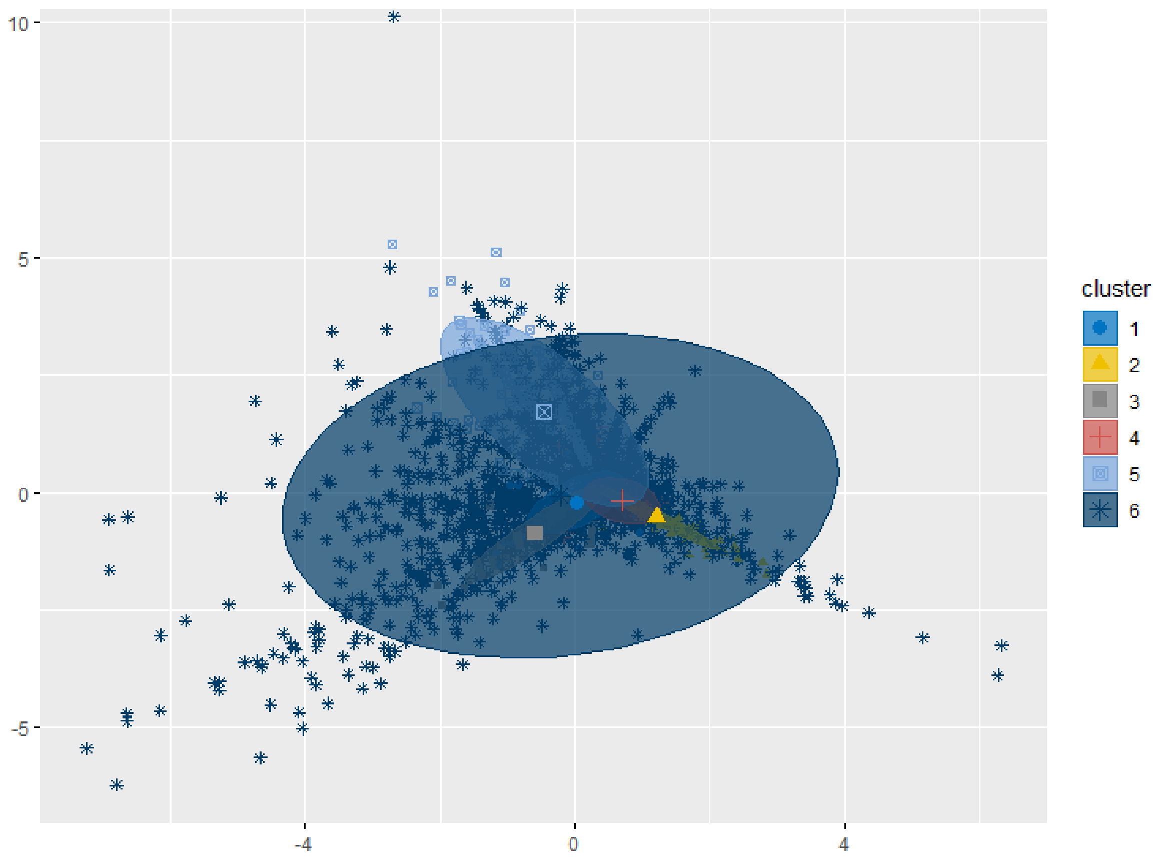


Figure 5.12: Cluster plot with k=6 and the stemming technique.

The execution time of K-means with the stemming technique was 1894 seconds.

In Figure 5.12, the implementation of the algorithm returns the following results: in cluster 1 there are 1008 documents of which 38 belongs to classical music, 36 belongs to folk music, 58 belongs to rap music, 661 belongs to rock music and 215 belongs to pop music, while in cluster 2 there are 1292 documents of which 1058 belongs to classical music, 65 belongs to folk music, 56 belongs to rap music, 56 belongs to rock music and 62 belongs to pop music. Furthermore, in cluster 3 there are 863 documents of which 31 belongs to classical music, 31 belongs to folk music, 70 belongs to rap music, 71 belongs to rock music and 660 belongs to pop music, while in cluster 4 there are 910 documents of which 91 belongs to classical music, 609 belongs to folk music, 48 belongs to rap music, 86 belongs to rock music and 76 belongs to pop music. Additionally, in cluster 5 there are 1250 documents of which 40 belongs to classical music, 49 belongs to folk music, 922 belongs to rap music, 97 belongs to rock music and 142 belongs to pop music, while in cluster 6 there are 5246 documents of which 1119 belongs to classical music, 596 belongs to folk music, 1039 belongs to rap music, 1126 belongs to rock music and 1366 belongs to pop music.

Regarding the above resulting plots, it is concluded that the K-means algorithm either using the stemming technique or not, using *tf-idf* method and stop words list, does not give the desired results. This is explained by the fact that using words with similar content in data-selection filters, the algorithm cannot perform proper grouping of clusters.

5.4.4 Text Clustering with Latent Dirichlet Allocation algorithm

This chapter presents the implementation of LDA algorithm. LDA is a commonly used technique in topic modeling that facilitates the automatic discovery of themes in a collection of documents.

```
# Load the necessary libraries
library(NLP)
library(tm)
library(topicmodels)
library(tidytext)
```

```
#Importing the file
df1=read.csv("C:/Users/user/Desktop/tweetsOfTexts2.csv",sep=";",header= TRUE, stringsAsFactor=FALSE)
attach(df1)
names(df1)
#Creating of corpus- Preprocessing of the text
xCorpus=VCorpus(VectorSource(df1$text))
clean.corpus <- tm_map(xCorpus,content_transformer(tolower))
clean.corpus <- tm_map(clean.corpus, removeNumbers)
myStopList=read.table("C:/Users/user/Desktop/stop.txt",header=FALSE,sep="\n",
strip.white=TRUE)
clean.corpus<-tm_map(clean.corpus,removeWords,myStopList[,1])
clean.corpus<-tm_map(clean.corpus,removeWords,stopwords("english"))
removeUsersNames<-(function(x)gsub("@\\w+", "", x))
clean.corpus<- tm_map(clean.corpus,content_transformer(removeUsersNames))
removeURL0<- (function(x) gsub("(f|ht)tp(s?)://\\S+", "", x, perl=T))
clean.corpus<-tm_map(clean.corpus,content_transformer(removeURL0))
removeStrangeCharac<- (function(x) gsub("[^[:alpha:][:space:]]*","",x))
clean.corpus<-
tm_map(clean.corpus,content_transformer(removeStrangeCharac), lazy =
TRUE)
clean.corpus <- tm_map(clean.corpus, removePunctuation)
```

```
21  clean.corpus <- tm_map(clean.corpus, stripWhitespace)
#Splitting text documents into words
22  dtm1<-DocumentTermMatrix(clean.corpus)
23  rowTotals<-apply(dtm1,1,sum)
24  dtm1.new<-dtm1[rowTotals>0, ]
#Compute the LDA model using Gibbs sampling
25  text_lda<-
    LDA(dtm1.new,k=5,method=”Gibbs”,control=list(nstart=5,seed=c(2003,5,63,100
    001,765),burnin=4000,iter=2000,thin=500,best=TRUE))
#Find the top 10 terms in each topic
26  theResults<-posterior(text_lda)
27  topTenTermsEachTopic<-terms(text_lda,10)
#Find the documents topic probabilities
28  text_docs<-tidy(text_lda, matrix=”gamma”)
```

LDA topic model is applied to analyze the topics hidden in the tweets. The LDA() function is going to run with the Gibbs sampling technique and K = 5 as the number of topics (line:25). The time execution of LDA algorithm is 188 seconds. The top 10 frequent words in each topic are tabulated in Table 5.16 using the terms () function (line: 27). The results lead to some observations, such as Topic 1 mainly concerns about the rock music category while Topic 2 corresponds mainly to the folk music category. Topic 3 mainly describes the classical music category, while Topics 4 and 5 mainly concern the rap music category and pop music category respectively. More specifically, Topic 1 consists of 2763 documents and Topic 2 of 2233 documents. Topic 3 consists of 2043 documents, Topic 4 of 1857 documents and Topic 5 of 1672 documents. However, as we saw earlier with the implementation of k-Means, each cluster can provide information about more than one topic. Thus, with the implementation of the LDA classifier, it can be seen that this algorithm can separate better the categories. This was achieved after a good cleaning of the text of the tweets and mainly using a list of the correct stop words.

Table 5.16: Top 10 terms of each topic.

Topic 1	**Topic 2**	**Topic 3**	**Topic 4**	**Topic 5**
rock	music	music	rap	pop
music	folk	classical	music	music
like	free	sad	hiphop	best
great	live	listening	video	love
radio	album	art	youtube	dance
good	awards	piano	star	rnb
band	guitar	track	playlist	jazz
tribute	acoustic	king	play	songs
indie	concert	amazing	beats	hits
metal	blues	friends	artist	soul

The LDA algorithm calculates the topic probabilities associated with each document, and the tidy() function is used to portray that information (line:28). Table 5.17 lists some topic probabilities of each document. As it appears from Table 5.17, LDA estimates that only 35.93% of the terms in the document 9437 are generated from Topic 1 while document 7750 is drawn almost from Topic 3 with 3% probability. In Table 5.17, the highest probability in each row is bold and the second highest probability is highlighted in red.

Table 5.17: Topic probabilities by document

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
9437	**0.359**	0.156	0.156	0.156	0.171
7750	0.142	0.142	0.171	**0.2**	**0.3**
7077	0.190	**0.333**	0.158	0.158	0.158
10333	**0.241**	0.151	0.181	**0.242**	0.181
8380	0.161	0.161	**0.354**	0.161	0.161
10575	**0.272**	0.196	0.151	**0.212**	0.166
2932	0.159	0.159	0.159	**0.376**	0.144
7095	**0.242**	**0.227**	0.196	0.151	0.181
7067	0.158	0.174	0.158	0.158	**0.349**
2932	0.159	0.159	0.159	**0.376**	0.144

Chapter 6 Conclusion and Future work

The purpose of this thesis was, in principle, to make an introduction to its field data analysis with machine learning techniques. More specific, we get deeper in data analysis and text mining sectors. Many programming languages allow us to work on those sectors, mainly R and Python. R was chosen as it is particularly suited for applications that contain statistical analysis and as well as for text mining tasks. The Twitter by extension gives us the ability to experiment and try any approach to realistic data as it provides free access to many of them. The greatest interest beyond the theoretical analysis of the models is the comparison of these with real data. Thus, it is possible to compare the models in the context of the work, and the evaluation of the results compared to the theoretical approaches from other papers to extract a final conclusion. My main concern is the detailed description of all the steps towards the creating of a complete text mining process from theory to practice, which will be dynamic and customizable and therefore can be used by someone else to extend the experiment or some possible future work to be proposed then. The structure of the work introduces scientific terms that require a basic background in statistics, and programming, but it does not cease to be simple so that each potential reader can reproduce the analysis presented and understand basic concepts and structures of R.

A basic conclusion is a fact that one of the most important steps in the whole process is the collection and pre-processing of the data. A lot of emphasis has to be given to the data cleaning and filtering phase as these will be the final data that will be the features according to which training will be done and therefore the prediction of the requested instances. Several papers indicate that best results were achieved with the SVM model in contrast to the Naïve Bayes, but there are others who support the opposite. According to our approach, we proved that naive Bayes is the best text classifier. We can conclude that we have reached several meaningful models with great performance measures, but there is definitely room for improvement. Some improvements could be the use of the Laplace smoothing parameter in the Naive Bayes model and the use of the cost and gamma parameters in the SVM model. Regarding the clustering algorithms, we have a poor clustering in the k-means algorithm as it did not make a good distinction between classes. It has been observed that if words with similar content are used in data-selection filters, the algorithm cannot perform proper grouping of clusters. As for the LDA algorithm, it gives good results, as it can separate the classes compared to the k-means algorithm. This was achieved by preprocessing the text using a stopword list to improving the quality of text, as it contains many syntactic features that may not be useful for analysis.

A suggestion for future work is the creating of a web-based application appropriate for text classification or text clustering using the algorithms that we studied. For this purpose, this application could be used as a core and with some modern tools that provide the R framework, create an online one tool with a simple user interface (UI) so any user can extract valuable insights from the text mining tasks. This feature is provided by the Shiny framework (http://shiny.rstudio.com) provided by RStudio and facilitates the creating of both an application server.R and a ui.R, which together can be easily installed on a server via of shiny apps (https://www.shinyapps.io) and be accessible from the internet.

Chapter 7 References

Ahmad, Mazyad., Fabien, Teytaud., and Cyril Fonlupt., ″A Comparative Study on Term Weighting Schemes for Text Classification″. The Third International Conference on Machine Learning, Optimization and Big Data, Tuscany, Italy, hal-01662131, 2017.

Akshay, Java., Xiaodan, Song., Tim, Finin., and Belle, Tseng. ″Why We Twitter: Understanding Microblogging Usage and Communities″, 9th WEBKDD and 1st SNA-KDD Workshop, 2007.

Alboukadel kassambara. ″Guide to Create Beautiful Graphics in R″, STHDA, 2013.

Bourdieu, P. ″The social space and the genesis of groups. Social Science Information″, 24(2), 195-220. doi:10.1177/053901885024002001,1985.

Bing, Liu., ″ Web Data Mining., Exploring Hyperlinks, Contents, and Usage Data″, Springer-Verlag Berlin Heidelberg ,Second Edition,2007, 2011.

Boyd, D., and Ellison, N., ″Social Network Sites: Definition, History, and Scholarship″, Journal of Computer-Mediated Communication, Volume 13, Issue 1, Oxford Academic, 2007.

Burt, R.S. ″Structural holes: The social structure of competition″. Cambridge, MA: Harvard University Press, 1995.

Claudine Beaumont, Twitter users send 50 million tweets per day, The Telegraph, 2010, http://www.telegraph.co.uk/technology/twitter/7297541/Twitter-users-send-50-million-tweets-per-day.html

Charanjeet, Kaur., ″Association Rule Mining using Apriori Algorithm: A Survey″, International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 2, Issue 6, June 2013.

Christopher,Manning.,and Hinrich, Schiitze., ″Foundations of Statistical Natural Language Processing″, The MIT Press, 1999.

Christopher,D., Manning.,Prabhakar, Raghavan.,Hinrich,Schutze., ″An Introduction to Information Retrieval″, Cambridge University Press,2008.

Dave Kuhlman, ″A Python Book: Beginning Python, Advanced Python and Python Exercises″, Platypus Global Media, 2011.

David, M., Blei., Andrew, Y., Ng., Michael, I,. Jordan., ″Latent Dirichlet Allocation ″, Journal of Machine Learning Research 3,993-1022, 2003.

Fragkiskos,Malliaros., and Konstantinos, Skianis., ″Graph-Based Term Weighting for Text Categorization″, Europe Fellowship in Graph Mining,2012.

Geraldine, Lo, Siou., Yutaka, Yasui., Ilona, Csizmadi., S., Elizabeth, McGregor., and Paula, J., Robson., ″Exploring Statistical Approaches to Diminish Subjectivity of Cluster Analysis to Derive Dietary Patterns, The Tomorrow Project″, American Journal of Epidemiology, Published by Oxford University Press on behalf of the Johns Hopkins Bloomberg School of Public Health,Vol. 173, No. 8 DOI: 10.1093/aje/kwq458, 2011.

Gerard, Salton.,Automatic text processing, Addison-Wesley Longman Publishing Co., Inc.,Boston, MA, 1988.

Hetal,Doshi., and Maruti, Zalte., ″Performance of Naïve Bayes Classifier – Multinomial Model on Different Categories of Documents″, National Conference on Emerging Trends in Computer Science and Information Technology (ETCSIT), Proceedings published in International Journal of Computer Applications (IJCA),2011.

Himanshu, Sharma., Arvind K., Sharma., ″Study and Analysis of Topic Modelling Methods and Tools –A Survey″, American Journal of Mathematical and Computer Modelling. Vol. 2, No. 2, 2017, pp. 84-87. doi: 10.11648/j.ajmcm.20170202.15, 2017.

Michael, Hahsler., and Sudheer, Chelluboina., ″Visualizing Association Rules:Introduction to the R-extension Package arulesViz″, The R Journal Vol. 9/2, December, 2017.

Ingo, Feinerer., Kurt, Hornik., and David, Meyer., ″Text Mining Infrastructure in R″, Journal of Statistical Software, Volume 25, Issue 5, 2008.

Jacobson, D., Brail, G., and Woods, D. ″Apis: A Strategy Guide.", O'Reilly Media, 2011.

Luis,Torgo., ″Data Minining With R, Learning with case studies ,2nd Edition″, CRC Press Taylor & Francis Group, 2017.

Jonathan, Milgram., Mohamed, Cherie.t, Robert, Sabourin., “One Against One” or “One Against All”: Which One is Better for Handwriting Recognition with SVMs?. Tenth International Workshop on Frontiers in Handwriting Recognition, Université de Rennes, La Baule (France). inria00103955, 2006.

Zi, Chu., Steven, Gianvecchio., Haining, Wang., and Sushil, Jajodia.″Detecting

Automation of Twitter Accounts: Are You a Human, Bot, or Cyborg?″,IEEE Transactions on dependable and secure computing, 2012.

JSON, Introduction to JSON, http://www.json.org

Machine Learning, Crash course, 2019, https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

Matthew, A. Russell., ″ Mining the Social Web, 2nd Ed.″, O'Reilly Media, 2014.

Mehdi, Allahyari., Seyedamin, Pouriyeh., Mehdi, Assefi., Saied, Safaei., Elizabeth, D. Trippe., Juan, B. Gutierrez., Krys, Kochut., ″A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques ″, In Proceedings of KDD Bigdas, Halifax, Canada, 2017.

Mid-day, Technology: From 2006 To 2017 - 11 Years Of Twitter Facts, Science Technology News, 2017, http://www.mid-day.com/articles/twitter-founding-anniversary-twitter-birthday-twitter-facts-twitter-trivia/16369196 .

Milligan, G.W., Cooper, M.C., An examination of procedures for determining the number of clusters in a data set. The Psychometric Society, vol.50, NO.2, pp 159-179, 1985.

Mizumoto, A., and Plonsky, L. ″R as a lingua franca: Advantages of using R for quantitative research in applied linguistics″. Applied Linguistics, 37, 284-291, doi:10.1093/applin/amv025, 2016.

Mushfeq-Us-Saleheen Shameem, Raihana Ferdous., ″An efficient K-Means Algorithm integrated with Jaccard Distance Measure for Document Clustering″, IEEE, 2009.

Namratha, M, Prajwala, T, R., A Comprehensive Overview of Clustering Algorithms in Pattern Recognition, IOSR Journal of Computer Engineering (IOSRJCE) ISSN: 2278-0661 Volume 4, Issue 6, PP 23-30, 2012.

Payam, Refaeilzadeh., Lei, Tang., and Huan, Liu., ″Cross-Validation″, Encyclopedia of Database Systems, Springer Science+Business Media, LLC, 2009.

Portes,A. and Sensenbrenner,J. ″Embeddedness and Immigration: Notes on the Social Determinants of Economic Action″. American Journal of Socialogy, 98(6), 1320-1350.doi:10.1086/230191, 1993.

Priyanka, Rawat., Surya, Kant., Bhaskar, Pant., Ankur, Chaudhary., and Shashi, Kumar Sharma., ″A Better Approach for Multilevel Association Rule Mining″, Proceedings of Fifth International Conference on Soft Computing for Problem Solving, Advances in Intelligent Systems and Computing 437, DOI 10.1007/978-981-10-0451-3_53, Springer Science+Business Media Singapore, 2016

Roger D. Peng, ″R Programming for Data Science″, Leanpub, 2015.

Sanjivani, Tushar, Deokar., ″Text Documents clustering using K Means Algorithm″, International Journal of Technology and Engineering Science Vol 1 (4), pp 282 – 286, 2013.

Rubayyi, Alghamdi., and Khalid, Alfalqi., ″A Survey of Topic Modeling in Text Mining″, International Journal of Advanced Computer Science and Applications (IJACSA), Vol. 6, No. 1, 2015.

Sapna, N.Gaikwad.,D, S. Bormane., ″Brain MR Image Classification Using Least Squares Support Vector Machine″, International Journal of Scientific & Engineering Research, Volume 5, Issue 3, 248, 2014.

Sledgianowski, D., Kulviwat, S. ″Using social network sites: the effects of playfulness, critical mass and trust in a hedonic context″. J. Comput. Inf. Syst. 49(4), 74–83, 2009.

Statista, Number of monthly active Twitter users worldwide from 1st quarter 2010 to 3rd quarter 2016 (in millions), The Statistics Portal, https://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users.

Tarsem Singh, ″A Comprehensive Review of Text Mining″, International Journal of Computer Science and Information Technologies(IJCSIT), Vol. 7 (1), 167-169, 2016.

The R Project for Statistical Computing, Introduction to R, https://www.r-project.org/about.html

Twitter Developer, https://developer.twitter.com/en/docs/ads/general/guides/getting-started.

Twitter Developer, Introduction to Tweet JSON, https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/intro-to-tweet-json.html .

Twitter Developer, Oauth with the Twitter API, https://developer.twitter.com/en/docs/basics/authentication/overview/oauth .

Twitter Help Center. FAQs about Retweets, https://support.twitter.com/articles/77606 what is R?, FAQs on R, https://cran.r-project.org/doc/FAQ/R-FAQ.html#What-is-R_003f

Wahiba Ben Abdessalem Karaa, ″A new Stemmer to improve Information Retrieval″, International Journal of Network Security & Its Applications, Vol.5, No.4, 2013.

wikipedia. Cluster analysis, https://en.wikipedia.org/wiki/Cluster_analysis

wikipedia. Cross-validation statistics, https://en.wikipedia.org/wiki/Crossvalidation_(statistics)

wikipedia. F1 score, https://en.wikipedia.org/wiki/F1_score

wikipedia. JSON, https://en.wikipedia.org/wiki/JSON

wikipedia. Python(programming language), https://en.wikipedia.org/wiki/Python_(programming_language)

wikipedia. Topic Model (https://en.wikipedia.org/wiki/Topic_model)

wikipedia. R(programming_language), https://en.wikipedia.org/wiki/R_(programming_language)#cite_ref-6

wikipedia. Social network, https://en.wikipedia.org/wiki/Social_network

wikipedia. Social networking service, https://en.wikipedia.org/wiki/Social_networking_service

wikipedia. Twitter, https://en.wikipedia.org/wiki/Twitter

wikizero. Naïve Bayes Classifier, https://www.wikizero.com/en/Naive_Bayes_classifier Ziheng,Wang.,Su,Wu.,Chang,Liu., and Shaozhi, Wu., ″The Regresssion of MNIST Dataset Based on Convolutional Neural Network″, The Internaltional Conference on Advanced Machine Learning Technologies and Applications, Advances in Intelligent Systems and Computing, Springer Nature Switzerland, 2019.