



University of Thessaly
Department of Electrical and Computer Engineering

DIPLOMA THESIS

Text Sentiment Analysis
Ανάλυση Συναισθημάτων από Κείμενα

Student
Eleni Siampali

Supervisor
Assoc. Prof. Michail Vassilakopoulos

Volos, 2018

University of Thessaly
Department of Electrical and Computer Engineering

DIPLOMA THESIS

Text Sentiment Analysis
Ανάλυση Συναισθημάτων από Κείμενα

Student
Eleni Siampali

Supervisor
Michail Vassilakopoulos
Associate professor

Cosupervisor
Aspassia Daskalopulu
Assistant professor

Volos, 2018

To my family and my friends

Acknowledgments

I would first like to thank my thesis advisor Associate Professor Michail Vassilakopoulos of the Department of Electrical and Computer Engineering at the University of Thessaly for his motivation, immense knowledge and mentoring. With our frequent communication and cooperation I managed to overcome any hurdles that came in my way towards concluding my research.

I could not also forget, to express my sincere thanks to the rest of my professors for their support, assistance and patience the whole time.

Finally, I would like to express my profound gratitude to my parents and my friends for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis.

Abstract

In this diploma thesis, we investigate the utility of linguistic features for detecting the sentiment of Twitter messages. We evaluate the usefulness of existing lexical resources as well as features that capture information about the informal and creative language used in micro-blogging. Thus, we conduct in-depth research to assess what is the best approach for a successful sentiment analysis. Moving in this direction, we present a hybrid model that combines the Lexicon and the Rule Based approaches.

In addition, by building an application we ask the users to type a movie of their preference. Subsequently, we categorize the tweets related to the particular movie (positive, negative, neutral) using the Multinomial Naive-Bayes classifier. In this way, we enable the each user to develop a comprehensive view of the particular movie based on the comments and reactions.

Περίληψη

Σε αυτή τη διπλωματική εργασία, ερευνούμε τη χρησιμότητα των γλωσσικών χαρακτηριστικών για την ανίχνευση του συναισθήματος των μηνυμάτων από το Twitter. Αξιολογούμε τη χρησιμότητα των υπαρχόντων λεξικών πόρων καθώς και τα χαρακτηριστικά που συλλαμβάνουν πληροφορίες σχετικά με την ανεπίσημη και δημιουργική γλώσσα που χρησιμοποιείται στο μικρο-blogging. Οπότε, πραγματοποιούμε μία σε βάθος έρευνα για να αξιολογήσουμε ποια είναι η καλύτερη προσέγγιση για μια επιτυχημένη συναισθηματική ανάλυση. Κινούμενοι σε αυτή την κατεύθυνση, παρουσιάζουμε ένα υβριδικό μοντέλο που συνδυάζει την προσέγγιση βασισμένη στο λεξικό και την προσέγγιση βασισμένη σε κανόνες.

Επιπλέον, χτίζοντας μία εφαρμογή ζητάμε από τους χρήστες να πληκτρολογήσουν μία ταινία της προτίμησής τους. Στη συνέχεια, ταξινομούμε τα tweets που σχετίζονται με τη συγκεκριμένη ταινία (θετικά, αρνητικά, ουδέτερα) χρησιμοποιώντας τον Multinomial Naive-Bayes ταξινομητή. Με τον τρόπο αυτό, δίνουμε τη δυνατότητα στον εκάστοτε χρήστη να σχηματίσει μια ολοκληρωμένη άποψη για τη συγκεκριμένη ταινία με βάση τα σχόλια και τις αντιδράσεις στο Twitter.

Contents

1	Introduction	1
2	Social Media and Data Mining	7
2.1	Today's Social Media	7
2.2	Social Media Analysis	10
2.3	The Social Network Twitter	11
2.3.1	About	11
2.3.2	History	12
2.3.3	Twitter's Popularity	13
2.4	Data Mining	14
2.4.1	Techniques with Supervised Machine Learning	16
2.4.2	Techniques with Unsupervised Machine Learning	17
2.4.3	The KDD Process	19
2.5	Sentiment Analysis	21
2.5.1	Techniques of Sentiment Analysis	22
2.5.2	Sentiment Analysis Tools	24
3	Data Extraction	27

3.1	Tweet Extraction	27
3.2	Analysis of the Extracted Tweets	29
4	Opinion Mining	33
4.1	Extraction and Tokenization	35
4.1.1	N-grams	36
4.2	Word Corporuses	37
4.3	Lexicon Based Approach	39
4.4	Rule Based Approach	40
4.5	Calculation of the Polarity	41
4.5.1	Sentiment Score (Polarity) of a Query	42
4.6	Combinational Approach - Vader	43
4.6.1	Vader: a Great Tool for Sentiment Analysis	45
4.6.2	Vader's Implementation	46
4.7	Sentiment Score of a Movie Data Set	50
5	Classification and Evaluation	53
5.1	Data	55
5.2	Pre-processing	56
5.2.1	Cleaning The Data	57
5.2.2	Word and Text Normalization	59
5.2.3	Data Transformation	61
5.3	The Classification Process	64
5.3.1	The Multinomial Naive-Bayes Clas- sifier	66
5.4	Performance Evaluation	67

CONTENTS

5.5	Different Classifiers	69
6	Twitter Application	77
7	Conclusions	81
7.1	Future Work	82
A	Tweet Extraction and Analysis	91
B	Data Sentiment - Vader	97
C	Pre-processing	101
D	The Classification Process	105
E	Twitter application	113

CONTENTS

List of Figures

2.1	Components of social media [19]	8
2.2	Number of monthly active Twitter users world-wide [1]	14
2.3	Data mining disciplines	15
2.4	Supervised vs Unsupervised [5]	18
2.5	The KDD process [15]	19
2.6	Sentiment analysis [4]	21
3.1	A subset of the data	29
3.2	Lengths along time	31
3.3	Likes vs Retweets	32
3.4	Real-time location	32
4.1	Architecture diagram of the proposed system	34
4.2	Example of tokenization	36
4.3	Unigram (1-gram)	37
4.4	Bigram (2-gram)	37
4.5	Trigram (3-gram)	37
4.6	Example of Lexicon based approach	39
4.7	Example of Rule based approach	40

LIST OF FIGURES

4.8	Calculation of the polarity	42
4.9	Extraction of the metrics	45
4.10	Optus tweet	46
4.11	Sentiment score of a subset	51
4.12	Metrics of a subset	52
5.1	Training data set	55
5.2	Test data set	56
5.3	Lemmatization's form	59
5.4	Example of lemmatization	60
5.5	Sample of term's frequency matrix	64
5.6	Sample of weight matrix	64
5.7	Accuracy vs Classifiers	75
5.8	Precision vs Classifiers	75
5.9	Recall vs Classifiers	76
5.10	F-score vs Classifiers	76
6.1	The Twitter application environment (Clap- perboard)	78
6.2	Sample of the categorized tweets	79
6.3	Summary of the categorized tweets	80
6.4	Pie-chart of the categorized tweets (%)	80
7.1	Architecture diagram of the application	84

List of Tables

4.1	Positive corpus subset	38
4.2	Negative corpus subset	38
4.3	Words' sentiment ratings	43
4.4	Words' sentiment metrics	45
5.1	Classifier's metrics	69
5.2	Metrics of different classifiers	74

LIST OF TABLES

Chapter 1

Introduction

Data mining has become an important part in the world of natural language processing, which deals with the interaction between computers and human languages and also deals with computers to program and easily process huge amount of natural languages [37]. It is a difficult task for viewers to interpret the user generated content in the social media since it might mostly will be large in number and handwritten. Since nowadays, there is a drastic change in the social media sector, from traditional media channels like magazines to social media channels like social networking sites, which has led researchers to study the scope of their exploitation in order to identify hidden knowledge contained within them. Millions of posts are appearing daily in web-sites that provide micro-blogging such as Twitter, Instagram, Facebook. Authors of those posts share opinions on variety of topics and discuss current issues. As more and more users post about products they use, micro-blogging web-sites become valuable

sources of people's sentiments. So, in order to build systems to mine opinions about any given topic, we need a powerful method for quickly identifying data. Recent years have seen an increasing interest in text analysis in which characterizations are expressed as the opinions, feelings, and attitudes expressed in a text, rather than just the facts. Sentimental analysis is the process of identifying positive and negative opinions, emotions, sentiments and evaluations.

This diploma thesis is aimed on giving the ability to the users to get information through commentary and reactions at the Twitter for the movie of their preference that would like to watch. This is accomplished by implementing algorithms of machine learning. More specific, using some classification techniques, we train our classifier with a data set related to movie reviews. Then, acquiring the tweets related to each user's selection movie, we classify them as positive, negative and neutral giving to the user an appropriate recommendation.

The chapters of the thesis are described briefly as follows: **Chapter 1** provides an introduction to the multidisciplinary field of data mining. It discusses the evolutionary path of information technology, which has led to the need for data mining, and the importance of its applications. It

examines the data types to be mined, including relational, transactional, and data warehouse. The chapter presents a general classification of data mining tasks, based on the kinds of knowledge to be mined, the kinds of technologies used and the kinds of applications that are targeted. Finally, major challenges in the field are discussed as well as the the target of this thesis.

Chapter 2 introduces the importance of social media on people's lives as well as their impact on them. Also, the useful knowledge that is observed from social media is described and the role that data mining plays on it. Specifically, in this chapter different machine learning techniques are presented in detail. Finally, emphasis is given both in the theoretical and in the practical part of sentiment analysis, such as its techniques and the tools that are used.

Chapter 3 introduces techniques for data extraction, particularly for tweets extraction. Then, some analysis on these extracted data takes place including general information about them.

Chapter 4 provides firstly a general idea of the way that the unlabeled data are reconstructed to the appropriate form. After that, different approaches are described, like the Lexicon Based and the Rule Based approaches, for the categorization of sentences or a even a document. Lastly, an implementation which combines both these approaches takes place, the Vader, through which the tweets in our

data set are classified.

Chapter 5 illustrates the classification model. Different techniques for data pre-processing are carried out. It first introduces the concept of data quality and then discusses methods for data cleaning, data integration, data reduction, data transformation, and data discretization. After that, the classification process is carried out using the Multinomial Naive-Bayes classifier. At last, the performance of our model is presented through different metrics as well as metrics' comparison of other classifiers.

Chapter 6 presents a common approach on how actually an application could be built. Aiming to help the user learn more about a movie, we constructed a simple interface in which anyone can type a movie of its preference. Then, through this implementation a set of tweets related to the selected movie are retrieved and several examples of tweet classification in positives, negatives and neutrals and what knowledge can be gained from it, are provided.

Chapter 7 is a summary of the results of our process. Furthermore, conclusions arise which are described in this chapter as well as recommendations for further improvement.

Lastly, the appendices represent the developer's code for implementing all the operations and functions that took place for building the model:

Appendix A introduces techniques for extracting and

analyzing tweets.

Appendix B illustrates the tool which was used to extract the sentiment of each sentence.

Appendix C consists of all the pre-processing stages and techniques which took place in order to train our classifier.

Appendix D provides the implementation of the classification process.

Appendix E presents a Twitter application model and how the tweets related to a specific movie, are classified in positives, negatives and neutrals through it.

Chapter 2

Social Media and Data Mining

2.1 Today's Social Media

Social media, social network in general, have gained remarkable attention and they play a leading role in today's society the last decade. People are becoming more and more interested in web technologies and they are relying on social network for information, news and opinion of other users on diverse subject matters. They have made their lives easier as social media allow them to be connected and interact with each other at anytime and anywhere through instant messages, web searching, button likes as shown below (Figure 2.1).

registered members broadcast daily short burst messages to the world, called tweets. In other words, it is a means of discovering interesting people or sites and following their burst messages for as long as they are interesting. In addition to its relative novelty, Twitter's big appeal is how rapid and scan-friendly it is. We can track hundreds of interesting tweets and read their content with a glance.

- Google+ (pronounced Google plus) is a social networking project from Google, designed to replicate the way people interact offline more closely than in any other social networking services. Specifically, it attempts to be differentiated by allowing more transparency in who you share with and how you interact. The project's slogan is "Real-life sharing rethought for the web" [40].
- LinkedIn is a social networking site designed specifically for the business community. The goal of this specific site is to allow registered members to establish and document networks of people they know and trust professionally as well as the chance to find new opportunities to grow their careers. It is very similar to Facebook in terms of its broad feature offering. These features are more specialized because they cater to professionals.

2.2 Social Media Analysis

The heavy reliance on social network sites causes them to generate massive data ceaselessly and rapidly. Many organizations, companies, individuals and even government of countries follow the activities on social media in order to obtain knowledge on the reaction of their audience to postings that they are concerned [2]. Authors of those posts share opinions on variety of topics and discuss current issues. Social media permits the effective collection of large scale data and this gives rise to major computational challenges. Nevertheless, as more and more users post, the efficient mining of the information retrieved from these data helps to discover valuable knowledge in different fields such as banking, marketing, government and even fashion trends. Moreover, these big data can be used as decision support tool by different entities associated with social media contents for many purposes. So, it is not only about the size of information which is much larger but also the fact that information tends to be more varied and extensive in its very nature and content.

2.3 The Social Network Twitter

2.3.1 About

One of the most popular micro-blogging services is considered to be the Twitter, where users create status messages (called tweets). These messages were originally restricted to 140 characters [25], but on November 7, 2017, the limit was doubled to 280 characters for all languages except Japanese, Korean and Chinese. Registered users can post tweets, but those who are unregistered can only read them. Users can access Twitter through its website interface, Short Message Service (SMS) or mobile device application software ("app"). Additionally, tweets contain special features like emoticons and hashtags that may have an analytical value. Hashtags are labels used for search and categorization and they are also included in the text represented by a "#". Emoticons are expressions of emotion and can either be written as a string of characters e.g., ":-)", or as a unicode symbol. If a tweet is a reply or is directed to another Twitter user, mentions can be used by prepending a username with "@" [11].

In the Twitter social media, each user has the option to choose the users who wants to follow as well. These are

the users whose tweets want to see directly on their timeline. Each user has the ability to re-publish a tweet as well as referring to its original creator (retweet). In addition, a tweets can be marked as something likeable. The above are useful metrics if we try to characterize a user about its activity on the Twitter social network. So, it is claimed that Twitter is the easiest way to see what is going on in the world in real time, which is one of the platform's major differentiators from the other networks on this list.

However, because of the rapid and direct communication that characterizes all social media tools, some issues arise as to the correctness of grammar and syntax. The resulting "noise" creates a problem in the part of the automated sense recognition process and its removal is a big bet.

2.3.2 History

Twitter was created in March 2006 by Jack Dorsey, Noah Glass, Biz Stone, and Evan Williams and launched in July of that year [33]. According to research published in April 2014, around 44 percent of user accounts have never tweeted [28]. The first tweet was posted by Jack Dorsey (creator) at 12:50 PM PST on March 21, 2006 and read "just setting up my twittr". On October 21, 2009, the first tweet was sent from space [7]. US astronauts Nicola Stott and Jeff Williams took part in a live 'tweetup' from the In-

ternational Space Station with around 35 members of the public at NASA Headquarters, Washington, DC, USA. The service rapidly gained worldwide popularity. In 2012, more than 100 million users posted 340 million tweets a day and the service handled an average of 1.6 billion search queries per day. In 2013, it was one of the ten most-visited websites and has been described as "the SMS of the Internet". As of 2016, Twitter had more than 319 million monthly active users, which continues to date [39].

2.3.3 Twitter's Popularity

The below statistic (Figure 2.2) shows a timeline with the amount of monthly active Twitter users worldwide. As of the third quarter of 2017, the micro-blogging service averaged at 330 million monthly active users. At the beginning of 2016, Twitter had reached 310 MAU (monthly active users) per quarter.

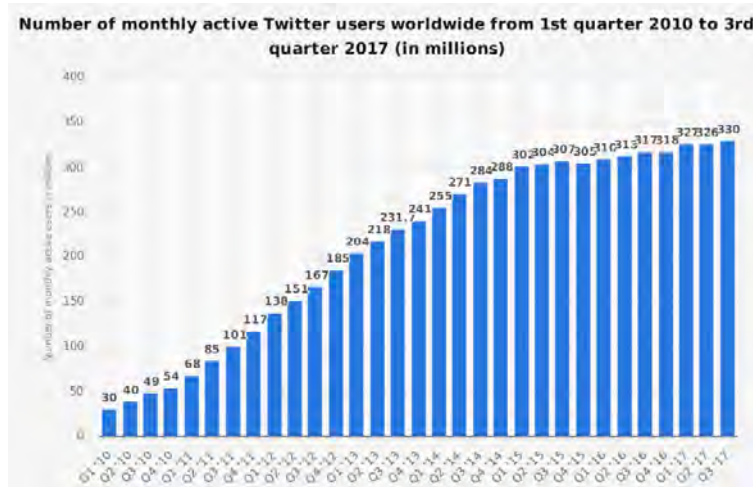


Figure 2.2: Number of monthly active Twitter users worldwide [1]

2.4 Data Mining

From the above, it is created the demand for data mining techniques [18] that is likely to facilitate reforming the unstructured data, place them within a systematic pattern and detect useful knowledge from these massive data sets like trends and rules. But first of all, it is important to explain the meaning of data mining.

What is in fact data mining? Data Mining is defined as extracting information from huge sets of data [34]. In other words, we can say that data mining is the procedure of

mining knowledge from data, known as knowledge discovery in databases (KDD) [13] using data mining techniques. The term KDD was coined at the first KDD workshop in 1989 (Piatetsky-Shapiro 199t) to emphasize that "knowledge" is the end product of a data-driven discovery. It is a discipline, lying at the intersection of statistics, machine learning, pattern recognition and other fields as shown below (Figure 2.3):

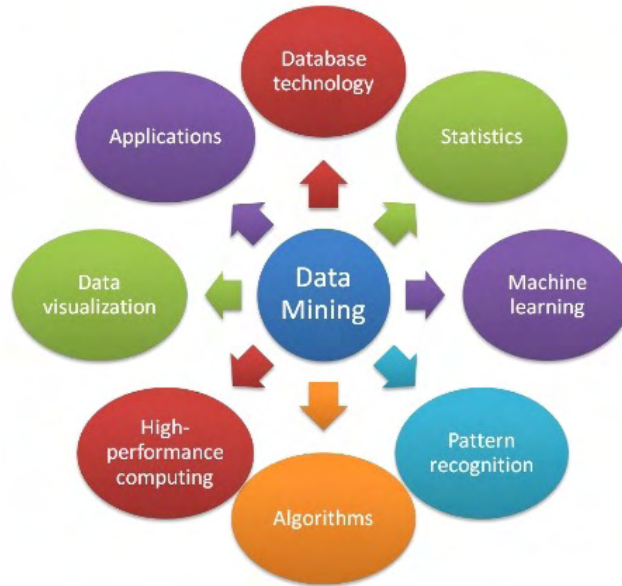


Figure 2.3: Data mining disciplines

All of these are concerned with certain aspects of data analysis. Consequently, they have have much in common, but each one has its own distinct flavor, emphasizing particular problems and types of different solutions.

Regarding the data mining techniques in machine learning

that were mentioned earlier, there are several of them, divided into two categories, developed and used in data mining projects recently [31].

2.4.1 Techniques with Supervised Machine Learning

The data is presented to the algorithm with example inputs and their associated outputs.

- ***Classification*** is a classic data mining technique based on machine learning and it is basically used to find out in which group each item is related within a given data set.
- ***Regression*** is a technique that attempts to determine the statistical relationship between two or more variables where a change in a dependent variable (usually denoted by Y) is associated with, and depends on, a change in one or more independent variables [42].
- ***Prediction*** as its name implied is one of data mining techniques that discovers relationship between independent variables and relationship between dependent and independent variables.
- ***Time Series Analysis*** is a statistical technique that deals with time series data or trend analysis.

Time series data means that data is in a series of particular time periods or intervals [21]. So, it helps us to understand what are the underlying forces leading to a particular trend in the time series data points and it helps us in forecasting and monitoring the data points by fitting appropriate models to it.

2.4.2 Techniques with Unsupervised Machine Learning

No labels are given to the learning algorithm. The goal is to discover groups in the data or to determine the distribution of data within the input space.

- ***Clustering*** is the procedure of dividing objects or data points into groups so that data points in the same group have something more in common than those in other groups. In other words, the aim is to separate groups with similar traits and assign them into groups, called as clusters.
- ***Association Rules*** is a rule-biased machine learning method [23] for finding frequent patterns, correlations, associations or causal structures from data sets found in various kinds of databases such as relational databases, transactional databases and other forms of data repositories.

- **Summarization** is the process of shortening a text document in order to create a summary with the major points of the original document. It creates text summary by definition of the relevance score of each sentence and extracting sentences from the original documents. This summarization method takes into account the weight of each sentence in each specific document [3].
- **Sequential Analysis** is one of data mining techniques that seeks to discover similar patterns in data transaction over a business period. The uncover patterns are used for further business analysis to recognize relationships among data.

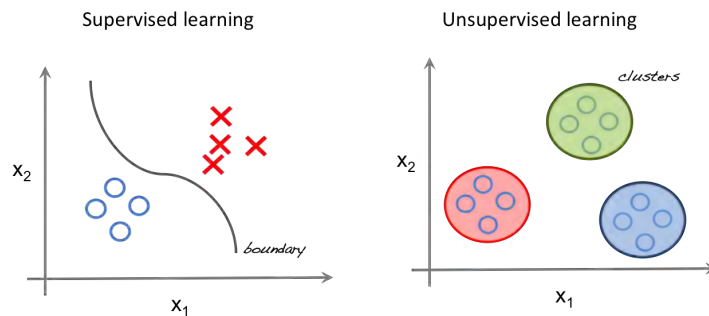


Figure 2.4: Supervised vs Unsupervised [5]

2.4.3 The KDD Process

The overall process of finding and interpreting patterns from data is called KDD process [37] and is shown below (Figure 2.5) as well as the steps that it involves:

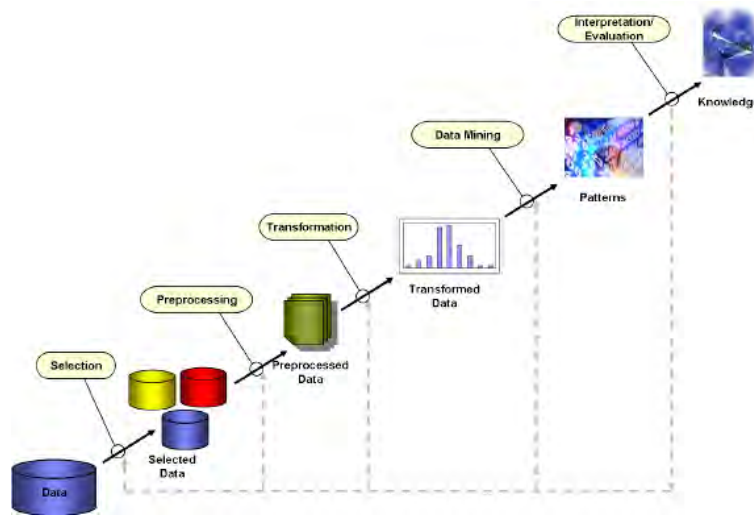


Figure 2.5: The KDD process [15]

1. **Data selection:** data relevant to analysis task are retrieved from db
 - Creation of a selected data set
2. **Pre-processing**
 - Data cleaning: remove noise and inconsistent data

- Data integration: multiple data sources may be combined
3. **Data reduction and transformation:** data are transformed or consolidated into forms appropriate for mining
 - Useful features are found
 - Dimensionality/Variable reduction
 - Invariant representation
 4. **Data mining:** intelligent methods are applied to extract patterns of interest
 - Summarization
 - Classification
 - Regression
 - Association
 - Clustering
 5. **Interpretation/Evaluation:** pattern evaluation and knowledge presentation
 - Visualization
 - Transformation
 - Removing redundant patterns
 6. **Use of discovered knowledge**

2.5 Sentiment Analysis

It is true that the sites on social media are commonly known for information dissemination, opinion/sentiment expression and product reviews. Users' opinions, for example, on social media such as Twitter, Facebook and Youtube are basically positive, negative or neutral (being commonly regarded as no opinion expressed). Consequently, micro-blogging web-sites have become valuable sources of people's sentiments and recent years have seen an increasing interest in text analysis in which characterizations are expressed as the opinions, feelings and attitudes expressed in a text, rather than just the facts.

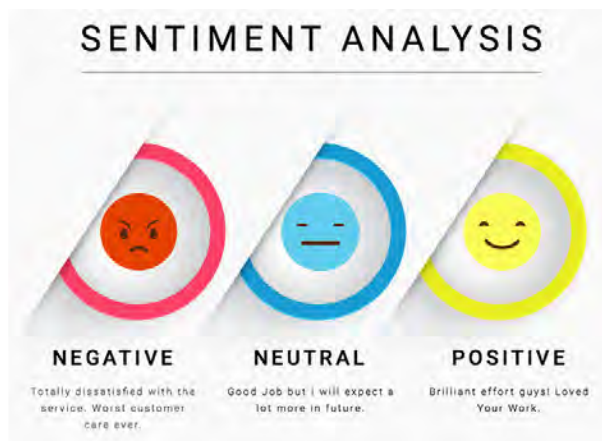


Figure 2.6: Sentiment analysis [4]

Sentiment analysis is the process of identifying these positive and negative opinions, emotions, sentiments and evaluations [32]. More specific, Twitter sentiment analysis of-

fers understanding, on a large scale, of users' sentiments. Many conclusions can be drawn through these sentiments. For example, regarding small businesses, it is easy for them to see what people are saying about business's brand, do market research on how people feel about competitors, market trends, product offerings and be able to analyze the impact of marketing campaigns on Twitter users [19]. Also, in general, people can see how popular is a player, a movie or even a product.

This is a process inextricably linked to the field of Natural Language Processing (NLP). This domain belongs to the fields of Linguistics and Computer Science and deals with the interaction between the native language and the computer language. We are going to describe and examine this particularly in the following sections.

2.5.1 Techniques of Sentiment Analysis

An appropriate approach to sentiment analysis is to start with a lexicon [6][12] of positive, negative and neutral words and sentences. In these lexicons, entries are marked with their prior polarity: out of context, does the word seem to evoke something positive, something negative or something neutral. Most of the present approaches to opinion mining and sentiment analysis are still far from

being able to perfectly extract the effective information associated with natural language texts. Especially, when dealing with social media, in which contents are often very distinct and noisy and the use of a domain-dependent training corpus is just not enough.

The different types of sentiment analysis approaches are machine learning (described in Subsections 2.4.1 and 2.4.2), Lexicon-based, Statistical and Rule-based approaches.

The “Lexicon Based Approach” involves calculating sentiment polarity for a review using the semantic orientation of words or sentences in the review. The “semantic orientation” is a part of subjectivity and opinion in text [10]. In other words, it is a database of lexical units for a language along with their sentiment orientations. This can be indicated as a set of tuples of the pattern (lexical unit, sentiment). Here, the lexical entity may be words, word senses, phrases, etc.

Then the “Rule Based Approach” looks for opinion words in a text and then classifies it based on the number of positive and negative words [43]. It considers different rules for classification such as dictionary polarity, negation words, booster words, idioms, mixed opinions etc.

Lastly, the “Statistical methods” are commonly semanti-

cally weak, meaning that, with the omission of clear affect keywords, other lexical or co-occurrence elements in a statistical model have limited predictive value individually [20]. As a conclusion, statistical text classifiers only work with adequate accuracy when given a very large text input. So, while these methods may be able to effectively classify user's text on the page-level or paragraph-level, they do not perform well on smaller text units such as sentences or clauses.

2.5.2 Sentiment Analysis Tools

Due to the multifaceted importance of sentiment analysis, tools, libraries and interfaces Application Programming Interfaces (APIs) are constantly being developed to help automate this process.

The Natural Language Toolkit (NLTK) (2017) is a Python language platform that makes it easier to manage Natural Language Processing (NLP). This is a free software program with an enormous contribution to Python's natural language processing and application in both education and industry. It refers to many natural languages and not just to English as most of these tools. It includes an Application Programming Interface (API) through which some important work on natural language processing is

being implemented. Some of these are tokenization, classification, and sentiment analysis described in the above section.

Chapter 3

Data Extraction

This diploma thesis focuses on analyzing users' commentaries in Twitter related to movies as well as their reactions to them. Its target is to separate these reactions in positives, negatives and neutrals finding the response of a specific movie to the audience. As a result, anyone will be able to appraise this movie or another movie based on these results. The particular procedure involves several steps which are described in further sections.

3.1 Tweet Extraction

For the purpose of this analysis, it is created the demand for obtaining the appropriate data.

The first step is to extract these data based on movies. Twitter provides the opportunity to retrieve data related to users' tweets by creating a developer account [35]. Through this method, specific codes for each one are retrieved like *the Consumer Key (API Key)*, *the Consumer Secret (API Secret)*, *the Access Token* and *the Access Token Secret*. These codes are then used to access the twitter database. A particular number of tweets about what is defined can be downloaded by selecting a specific name of hashtag.

In this case the name of the movie we choose to deal with, is "The Dark Tower" and the number of tweets we set is 100. A sample of the extracted tweets is the following (Figure 3.1):

	text	len	ID	Date	Source	Likes	RTs
0	RT @DarkTowerMovie: Take a deeper look inside ...	140	920884006362771456	2017-10-19 05:27:10	Twitter for iPad	0	43
1	Get yourself up that bookcase! https://t.co/Mx...	54	916334890559385600	2017-10-06 16:10:37	Twitter for iPad	25	6
2	RT @DarkTowerMovie: Download the #SombraGroup ...	140	897684074810265601	2017-08-16 04:58:56	Twitter for iPad	0	26
3	I can't wait to see the @LEGO_DarkTower movie!...	70	897045335813615616	2017-08-14 10:40:49	Twitter for iPad	27	9
4	RT @LEGO_DarkTower: Wolves of the Calla. (The ...	140	896987254903771136	2017-08-14 06:50:01	Twitter for iPad	0	22
5	RT @JLYNNCHRISTIE: Tom Taylor was great in the...	140	896122802662658048	2017-08-11 21:34:59	Twitter for iPad	0	2
6	RT @LEGO_DarkTower: And if you don't believe t...	140	894588068790980608	2017-08-07 15:56:30	Twitter for iPad	0	7
7	RT @Junkie_XL: Our final @DarkTowerMovie: Stud...	140	894222994486304768	2017-08-06 15:45:50	Twitter for iPad	0	8
8	RT @LEGO_DarkTower: I have 2 versions of Lego ...	140	893818356779941893	2017-08-05 12:57:57	Twitter for iPad	0	10
9	RT @Junkie_XL: 🚩 @DarkTowerMovie Studio Time S...	140	893731592174276608	2017-08-05 07:13:10	Twitter for iPad	0	11

Figure 3.1: A subset of the data

3.2 Analysis of the Extracted Tweets

After the data have been extracted, some analysis is carried out on them and the movie.

1. General information

- Length:
 - The length's average in tweets: *113.4848*
- Likes:
 - The tweet with the most likes is: *Fantastic! An even better trailer! <https://t.co/6Ocyv9QAU6>*
 - Number of likes: *66*
 - Number of characters: *58*
- Retweets:
 - The tweet with the most retweets is:
RT @StephenKing: Pilgrim, there are other worlds than these. Come with us on the adventure. DarkTowerMovie <https://t.co/0h0UnMq39i>
 - Number of retweets: *8974*
 - Number of characters: *131*

2. **Length of tweets along time:** The provided line graph (Figure 3.2) depicts the average length of tweets with the passing of time. It can be seen that the average length of tweets varies at different periods of the past year but it does not exceed the length of 140. However, only 3 times, specifically in January, at the

end of April and in August, the least average length is noted.

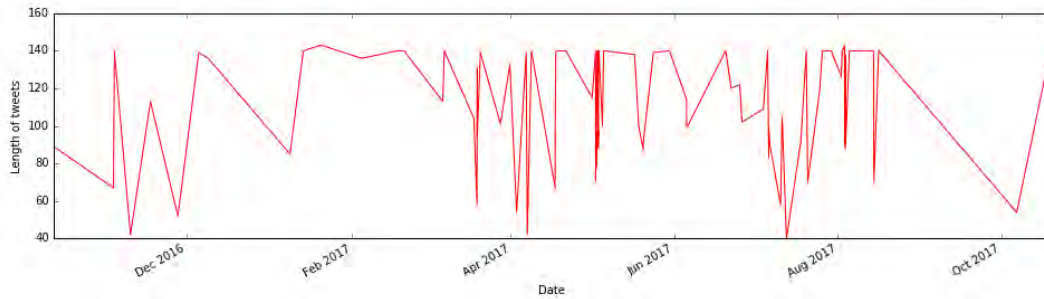


Figure 3.2: Lengths along time

3. Likes vs Retweets: The given graph (Figure 3.3) illustrates the number of likes and the number of retweets along time the last year. As is observed from the graph, the highest number of retweets is noted in April 2017 reaching the maximum 9000. After that, this number is maintained to zero when, from the end of May until the August, rapid fluctuations are marked. Regarding the number of likes, the changes are insignificant, being noted in December, April and May.

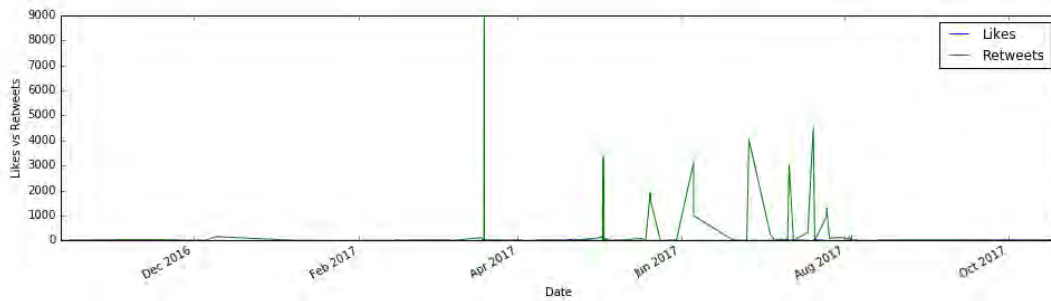


Figure 3.3: Likes vs Retweets

- 4. Real-time location of the tweets:** The below image map (Figure 3.4) shows, with the red dots, the different locations of the twitter users around the world. It can be seen that the most reactions of the movie are coming from the United States and South Africa. So, it offers an opportunity to see the origin of each tweet and other useful information such as in which part of the world the movie is popular.

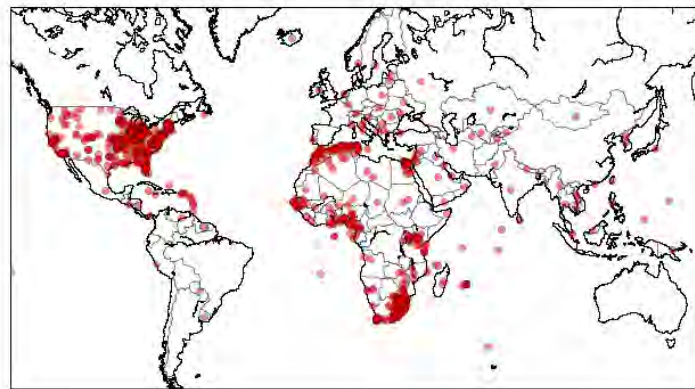


Figure 3.4: Real-time location

Chapter 4

Opinion Mining

After the data have been extracted, the sentiment analysis is carried out using several techniques which lead to conclusions about the views and trends of users for this movie. For the purpose of this study, we propose two approaches, the Lexicon Based and Rule Based approaches. More specific, the Lexicon approach is used to find out the polarity of words. We are classifying these tweets as positives and negatives in order to give sentiment on each one. The following figure (Figure 4.1) shows the entire proposed architecture system.

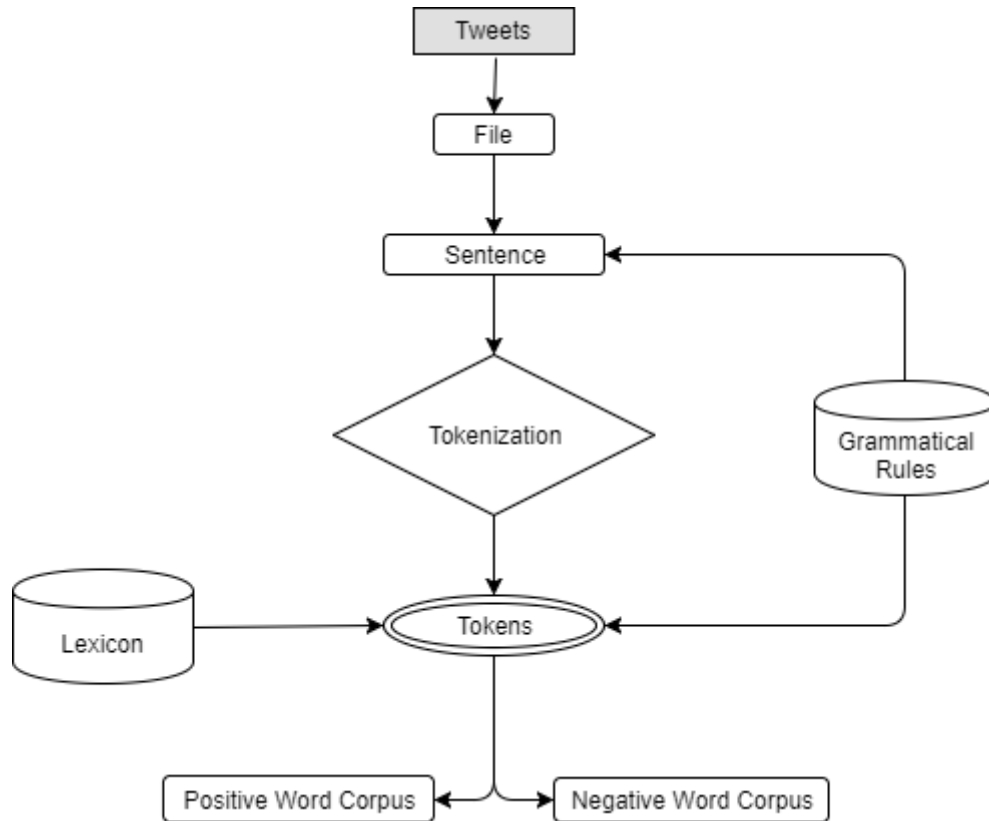


Figure 4.1: Architecture diagram of the proposed system

In the above system different phases of development are proposed [32]. A data set is created using twitter posts of movie reviews. Here sentiment analysis on each word in the tweets is performed. Firstly, we assemble all the tweets in a file. In the next step, we extract sentences based on punctuation marks and we apply grammatical rules to find the sense and context of each sentence. Then, we tokenize sentences based on word boundaries. Subsequently, extracted words are matched against words in the positive and negative word corpus that we have already

built. The polarity of each word is thus found using several algorithms. After that, final polarity of sentence is drawn based on the count. In the final phase, the polarity of entire tweets are found using the count and generate a summary based on that.

4.1 Extraction and Tokenization

In this procedure, a number of tweets are collected for testing and are stored in a file. These are then extracted one by one from the file. A sentence in the file is considered as one tweet by keeping punctuation mark as its base. That is, whenever there is a punctuation mark between any two words, till the previous word before the punctuation mark is taken as one sentence. Likewise, all the sentences are extracted. Each word in the sentence is also extracted by considering space between the words as its base. So, whenever a space between letters in the sentence appears, the letters before the blank space is considered as one word. In the same way, all the words in the sentence are extracted and each word is called as one “token”. The processing of making tokens from sentences is named as tokenization. Usually, it is accompanied by other optional processing steps, such as the removal of stop words and punctuation characters, stemming or lemmatizing, and the construc-

tion of n-grams which are described in further sections. Both extraction and tokenization are performed one by one [36]. In other words, one sentence will be considered at a time and tokenization will be performed in it. And it will be proceeded for further steps. After completion of all the steps in one sentence only, another sentence will be extracted and proceeded for further steps.

Below is an example (Figure 4.2) of a simple but typical tokenization step that splits a sentence into individual words, removes punctuation, and converts all letters to lowercase.

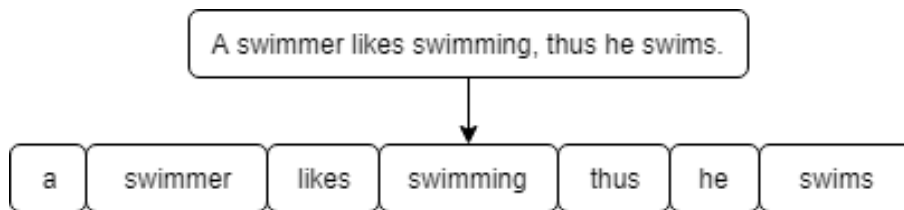


Figure 4.2: Example of tokenization

4.1.1 N-grams

The n-gram model is a variation of tokenization. In the n-gram model, a token can be defined as a sequence of n items. The simplest case is the so-called unigram (1-gram) where each word consists of exactly one word, letter, or symbol. Choosing the optimal number n depends on the

language as well as the particular application. The role of 2-gram model as well as the 3-gram model are shown below (Figure 4.3), (Figure 4.4) and (Figure 4.5) respectively.

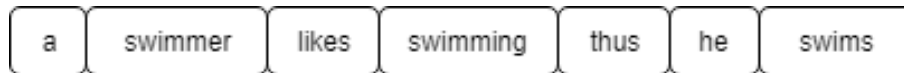


Figure 4.3: Unigram (1-gram)

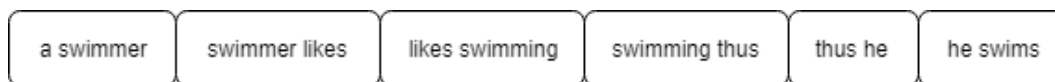


Figure 4.4: Bigram (2-gram)



Figure 4.5: Trigram (3-gram)

4.2 Word Corpora

A positive word corpus with all possible positive words which are usually used in tweets is created (Table 4.1) as well as a negative word corpus (Table 4.2).

Positive Corpus					
admire	ecstatic	impress	majestic	qualified	unquestionably
awesome	excellent	incredible	memorable	quaint	upgraded
amazing	enthusiasm	integral	modesty	quieter	undisputed
beautiful	fascinate	joyful	nicely	recommend	victory
beloved	fun	jubilantly	noble	remarkable	vigilance
bravo	favorite	justly	non-violent	rapt	vivid
charming	glory	kid-friendly	outsmart	satisfy	wow
congratulations	great	kudos	outstanding	splendid	wonderful
cool	grateful	keen	optimal	super	well-balanced
delight	happiness	lovely	pleasant	thankful	yay
darling	hot	lucid	perfect	temptingly	zippy
desirable	humorous	liberty	passionate	thriving	zenith

Table 4.1: Positive corpus subset

Negative Corpus					
ambiguous	exploit	illusion	murky	quibble	unachievable
annoyance	erosion	ignorance	malevolence	quarrellous	unfriendly
ambush	extravagantly	intimidate	mischievously	quash	undesirable
boring	fatuously	jaundiced	naive	rantingly	vulnerable
bias	fake	juddering	notorious	reticent	vociferous
brutal	freakish	jitter	nuisance	racism	violent
contagious	grievous	killer	obstructed	smugly	wrestle
combative	grumpily	kaput	overzealous	sagging	wobble
cannibal	gaudy	kook	outsider	scolded	woeful
disappointing	hawkish	low-rated	peril	truant	yawn
defame	harmful	lascivious	painful	tauntingly	zap
depressingly	heavy-handed	ludicrously	plagiarize	terrible	zealot

Table 4.2: Negative corpus subset

4.3 Lexicon Based Approach

A Lexicon Based Approach is used to match words or sentences in a file against words or sentences in the corpuses [10]. We made words in sentences as individual tokens, as a result to be very easy to compare the tokens with each word in the two corpuses. Here is an example (Figure 4.6) which shows clear idea about word matching:

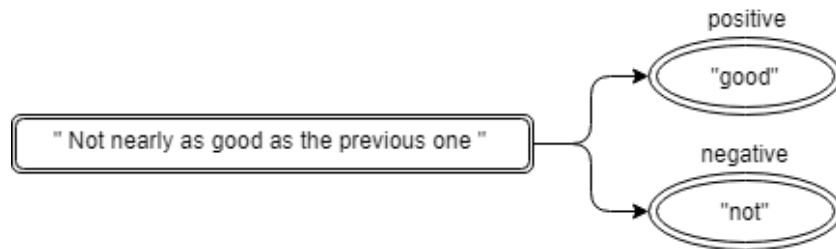


Figure 4.6: Example of Lexicon based approach

The above tweet contains some number of words but only two words show positive and negative polarities. i.e., "not" and "good".

Then we compare these polarity words with words in corpuses [8] which is already built. If a word is present in any of the two corpuses, it indicates that the word has a polarity which may be positive or negative. By doing this, the polarity for each word is found out by the use of a counter. Thus, we will be able to find the polarity of each word in the sentence. This is repeated for the entire sentences in

document.

4.4 Rule Based Approach

After calculating the polarity for each word, we apply certain grammatical rules for improving accuracy of the sentences as well as the entire document. This is done because of the fact that a word alone will not certainly make a sense but certain words and phrases together make the real sense/meaning [43][37]. By doing this, the efficiency of the classification is improved and the final result will be more accurate than only by applying the Lexicon Based Approach.

For example, consider the tweet: "The second half is not good" (Figure 4.7).

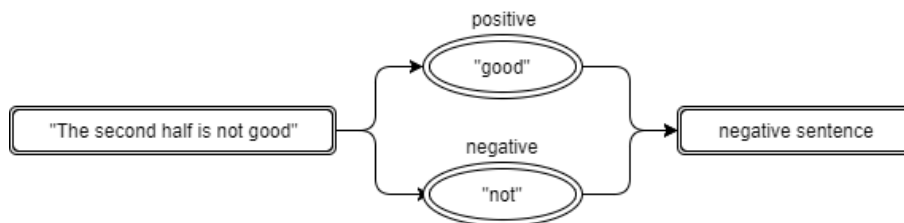


Figure 4.7: Example of Rule based approach

Here the word "not" in front of "good" makes the entire sentence negative. Thus, one rule is made in such a way that when "not" comes in front of an adverb, the sentence

will become negative. Like this, certain rules are made and applied to each sentence and their polarity is found out.

4.5 Calculation of the Polarity

Since the sentences are taken one by one, the final polarity of each sentence is calculated by totaling the polarity of words in each sentence. After that the polarity of the entire document is also calculated using the number of positive and negative sentences. Using this a summary is generated [41]. The performance of the system depends on training data sets and also content (i.e. tweets) in these data sets. Thus, the two approaches, referred in Section 4.3 and Section 4.4, together are very simple and effective to analyze the sentiment from text. Since we have used a built-in domain containing tweets, there is no need of analyzing subjective and objective tweets separately. This shows how the context or domain information affects sentiment analysis. So, based on the results, Lexicon Based Approach along with Rule Based Approach performs better than the already existing approaches for sentiment analysis on twitter data.

4.5.1 Sentiment Score (Polarity) of a Query

This section presents how the sentiment score of a query is calculated (Figure 4.8). In fact, all the steps described before as well as the whole procedure is summarized in this figure.

Majority **likes** this, but I **like** do **not** this.

- Phase 1 (negations):

$$\text{pos_score} = 0 - \text{negation weight} = -2$$

- Phase 2 (individual words):

$$\text{word "likes": pos_score} = -2 + 1 = -1$$

$$\text{word "not": neg_score} = 0 + 1 = 1$$

$$\text{word "like": pos_score} = -1 + 1 = 0$$

- Phase 3 (opposite conjunctions):

$$\text{sentiment_count} = 3$$

$$\text{total_score} = \text{pos_score} - \text{neg_score} - 1/2 * \text{sentiment_count}$$

$$= 0 - 1 - 3/2$$

$$= -5/2$$

Sentiment: **Negative**

Figure 4.8: Calculation of the polarity

4.6 Combinational Approach - Vader

Vader is a Python package which belongs to a type of sentiment analysis based on lexicons of sentiment-related words. In this approach, each of the words in the lexicon is rated as to whether it is positive or negative, and in many cases, how positive or negative is. This is the small differentiation from the above process. Also, it takes into account all the grammatical rules used in Rule Based Approach as well as some other aspects which are presented in the next section. An excerpt from Vader's lexicon is shown in Table 4.3, where more positive words have higher positive ratings and more negative words have lower negative ratings.

Word	Sentiment rating
tragedy	-3.4
rejoiced	2.0
insane	-1.7
disaster	-3.1
great	3.1

Table 4.3: Words' sentiment ratings

The lexicon needs to have good coverage of the words in each text of interest, otherwise it won't be very accurate. On the other hand, when there is a good fit between the

lexicon and the text, this approach is accurate, and additionally quickly returns results even on large amounts of text.

As is observed and as described in the previous process, when Vader analyzes a piece of text it checks to see if any of the words in the text are present in the lexicon.

For example, the sentence “The food is good and the atmosphere is nice” has two words in the lexicon (good and nice) with ratings of 1.9 and 1.8 respectively.

The way metrics are conducted is shown in Figure 4.9. Vader produces four sentiment metrics from these word ratings, which can be seen below (Table 4.4). The first three, positive, neutral and negative, represent the proportion of the text that falls into those categories. As also can be seen, our example sentence was rated as 45% positive, 55% neutral and 0% negative. The final metric, the compound score, is the sum of all of the lexicon ratings (1.9 and 1.8 in this case) which have been standardized to range between -1 and 1. In this case, our example sentence has a rating of 0.69, which is pretty strongly positive.

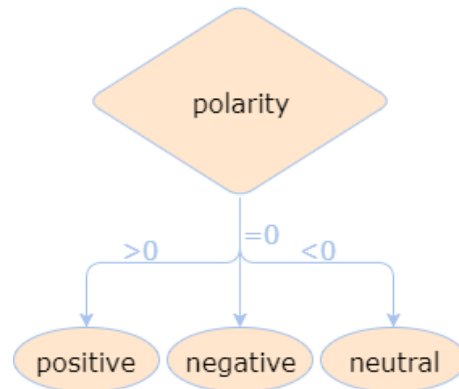


Figure 4.9: Extraction of the metrics

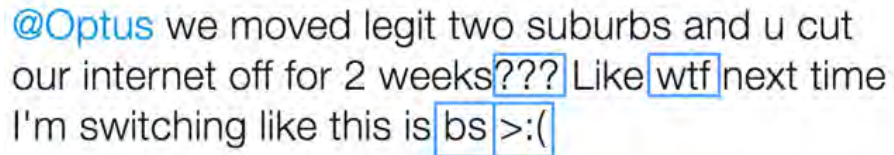
Sentiment metric	Value
Positive	0.45
Neutral	0.55
Negative	0.00
Compound	0.69

Table 4.4: Words' sentiment metrics

4.6.1 Vader: a Great Tool for Sentiment Analysis

The fact that lexicons are expensive and time-consuming to produce means they are not updated all that often. This means that they lack a lot of current slang that may be used to express how a person is feeling. Take the below tweet (Figure 4.10) to Optus' customer support account, for example. As can be seen, all of the elements of this text indicate that the writer is unhappy (in the blue boxes) but they are actually informal written - multiple punctuation marks, acronyms and an emoticon. If someone did not take

this information into account, this tweet would actually look neutral to a sentiment analysis algorithm.



@Optus we moved legit two suburbs and u cut our internet off for 2 weeks ??? Like wtf next time I'm switching like this is bs >:(

Figure 4.10: Optus tweet

Vader handles this by including these sorts of terms in its lexicon.

4.6.2 Vader's Implementation

This subsection is dealing with different tweets and the results exported using Vader.

- We start with a base sentence:

```
print_sentiment_scores("I just got a call from my boss - does he realize it's Saturday?")
```

vader's result: ("I just got a call from my boss - does he realize it's Saturday?"). 'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0
 Vader rates this sentence as neutral.
- We add an emoticon:

```
print_sentiment_scores("I just got a call from my
```

boss - does he realize it's Saturday? :(")

vader's result:("I just got a call from my boss - does he realize it's Saturday? :("). 'neg': 0.209, 'neu': 0.791, 'pos': 0.0, 'compound': -0.4404

Now Vader is rating it as negative, picking up the sad face as useful sentiment-related information.

- Let's now add the acronym 'smh' (shaking my head):
print_sentiment_scores("I just got a call from my boss - does he realize it's Saturday? smh :(")
vader's result:("I just got a call from my boss - does he realize it's Saturday? smh :("). 'neg': 0.321, 'neu': 0.679, 'pos': 0.0, 'compound': -0.6369
Vader also picks this up and rates the sentence as even more intensely negative.

What about the word context?

Vader does not just do simple matching between the words in the text and in its lexicon. It also considers certain things about the way the words are written as well as their context.

- Let's explore this with another sentence:
print_sentiment_scores("The food is good.")
vader's result:("The food is good."). 'neg': 0.0, 'neu': 0.508, 'pos': 0.492, 'compound': 0.4404

One of the things that Vader recognizes is capitalization, which increases the intensity of both positive and negative words. Below is shown that capitalizing ‘good’ increases the positive intensity of the whole sentence.

- Capitalization:
`print_sentiment_scores("The food is GOOD.")`
vader's result: ("The food is GOOD."). 'neg': 0.0, 'neu': 0.452, 'pos': 0.548, 'compound': 0.5622

Another factor that increases the intensity of sentence sentiment is exclamation marks, with up to 3 exclamation marks adding additional positive or negative intensity:

- Add exclamation marks:
`print_sentiment_scores("The food is GOOD!")`
vader's result: ("The food is GOOD!") 'neg': 0.0, 'neu': 0.433, 'pos': 0.567, 'compound': 0.6027

Vader also takes into account what happens when modifying words are present in front of a sentiment term. For example, “**extremely bad**” would increase the negative intensity of a sentence, but “**kinda bad**” would decrease it.

- Let’s see what happens if we add ‘really’ in front of


```

'good':
print_sentiment_scores("The food is really
GOOD!")
vader's result:("The food is really GOOD!") 'neg':
0.0, 'neu': 0.487, 'pos': 0.513, 'compound': 0.6391

```

Finally, Vader also handles changes in a sentence's sentiment intensity when it contains 'but'. Essentially, the rule is that the sentiments expressed both before and after the 'but' are taken into consideration, but the sentiment afterwards is weighted more heavily than that before.

- Let's see how this looks:

```

print_sentiment_scores("The food is really
GOOD! But the service is dreadful.")
vader's result:("The food is really GOOD! But the
service is dreadful.") 'neg': 0.192, 'neu': 0.529, 'pos':
0.279, 'compound': 0.3222

```

Our score has dropped from 0.64 to 0.32, as Vader has taken that 'dreadful' far more into account than the 'really GOOD!'.

So, it is clear that using Vader not only the Lexicon Based Approach but also the Rule Based Approach are illustrated. What makes Vader special for sentiment analysis, is that it takes also into consideration the emoticons, the variety of punctuation marks and the informal way of peo-

ple's writing that take place in tweets. This is the main reason we prefer to continue with it.

4.7 Sentiment Score of a Movie Data Set

For the purpose of this diploma, a data set related to users' reactions and comments was downloaded as was described in Chapter 3. Subsequently, before the sentiment score of each one had been calculated, an effort was made to rate the tweets manually and classify them into positives, negatives and neutrals. Its purpose was to confirm the correctness of what was described earlier. The results of the tweets that we rated manually and the results obtained with the use of Vader were very close to similar. Here is a subset of them (Figure 4.11).

	text	sentiment
0	The brand new issue of @prostheticsmag will be...	0
1	Uneven & rushed #TheDarkTower #MovieReview...	0
2	@StephenKing Watching #TheDarkTower Sir!	0
3	I'm watching The Dark Tower #telfie @TheDarkTo...	0
4	#TheDarkTower: C+ Not bad. Tolerable if you fo...	-0.5574
5	Whoever made #TheDarkTower movie forgot the fa...	0
6	Procrastinating in order to put off watching #...	0.5574
7	The Four Of Us Are Dying #TheDarkTower https:/...	0
8	First rule of watching #TheDarkTower - forget ...	-0.2263
9	The man in black from the dark tower series in...	0

Figure 4.11: Sentiment score of a subset

Also, the metrics of the specific tweets are presented (Fig. 4.12), which were mentioned formerly.

	text	sentiment
0	The brand new issue of @prostheticsmag will be...	neutral
1	Uneven & rushed #TheDarkTower #MovieReview...	neutral
2	@StephenKing Watching #TheDarkTower Sir!	neutral
3	I'm watching The Dark Tower #telfie @TheDarkTo...	neutral
4	#TheDarkTower: C+ Not bad. Tolerable if you fo...	negative
5	Whoever made #TheDarkTower movie forgot the fa...	neutral
6	Procrastinating in order to put off watching #...	positive
7	The Four Of Us Are Dying #TheDarkTower https:/...	neutral
8	First rule of watching #TheDarkTower - forget ...	negative
9	The man in black from the dark tower series in...	neutral

Figure 4.12: Metrics of a subset

However, having already tested the classification methods of the tweets, which are described in further sections, we do not continue this diploma and specifically the implementation, with the data set that we have just described. Instead, we use another data set consisted of 21 thousands labeled tweets, all related to movies. This one, guarantees better results because of the big importance of a large number of data in classification methods that we apply later.

Chapter 5

Classification and Evaluation

This chapter is about how the classification model is constructed and its connection with sentiment analysis. Commonly, sentiment analysis is treated as a classification task as it classifies the orientation of a text into either positive or negative. Supervised machine learning is one of the widely used approaches towards sentiment classification in addition to Lexicon Based and Rule Based methods. One of the main reasons that we select this approach is that the machine learning methods are based on training an algorithm, mostly classification on a set of selected features for a specific mission and then test on another set whether it is able to detect the right features and give the right classification. Thus, we downloaded the training labeled data set (Section 4.7).

In addition, it has been proved that Lexicon and Rule

Based techniques, do not perform as well in sentiment classification as they do in topic categorization that we showed in the previous chapter (Chapter 4) due to the nature of an opinionated text which requires more understanding of the text while the occurrence of some keywords could be the key for an accurate classification. More specific, machine learning classifiers such as Naive Bayes, Maximum Entropy and Support Vector Machine (SVM) are used for sentiment classification to achieve higher accuracy. This is the goal of this chapter and diploma, in general. Several steps such as pre-processing, data transformation, classification process and performance evaluation take place and are analyzed in detail in order to take the desired results.

In fact, we suggest a computational frame for sentiment analysis that consists of three key stages. Firstly, most relevant features will be extracted by employing extensive data transformation which had been cleaned before. Second, the Multinomial Naive-Bayes classifier will be applied on each of the feature matrices constructed in the first step and the accuracies resulting from the prediction will be computed, and third the classifier's performance will be evaluated using different metrics as well as to be compared with other classifiers.

5.1 Data

The first step for our sentiment analysis in order to continue to the implementation of the classification model is to obtain a data set related to movies as we mentioned in (Section 4.7). We selected a large enough labeled data set because the bigger it is, the better results we take. After that, necessary was to split this set into two categories, a training data set and a test data set. These two sets are very important so that we can build our classifier and train it to become as good as possible in prediction and results without failures. Below we can see a subset of both training and test (Figure 5.1) and (Figure 5.2).

	Sentiment	Text
17296	positive	Haven't tweeted nearly all day Posted my webs...
17610	positive	@hsumilo too much playing wii la ... dont bluff
833	positive	. . . a sour little movie at its core ; an exp...
69	positive	a taut , intelligent psychological drama .
6351	negative	when the fire burns out , we've only come face...

Figure 5.1: Training data set

	Sentiment	Text
19156	positive	@Snakehit new job? congrats
8033	negative	the issue of faith is not explored very deeply
16104	negative	It seems that Twitter lost some updates yester...
7270	negative	one of those pictures whose promising , if rat...
21327	neutral	#ihateitwhen i think of a cool twit,by da tym ...

Figure 5.2: Test data set

5.2 Pre-processing

The next step of our process is a kind of pre-processing of the data. In fact, pre-processing the data is the process of cleaning and preparing the text for classification. Online texts contain usually a lot of noise and uninformative parts. In addition, on words level, many words in the text do not have an impact on the general orientation of it. Keeping those words makes the dimensionality of the problem high and hence the classification more difficult since each word in the text is treated as one dimension. Here is the hypothesis of having the data properly pre-processed: to reduce the noise in the text should help to improve the performance of the classifier and speed up the classification process, thus aiding in real time sentiment analysis.

The whole process involves several steps: text cleaning, white space removal, expanding abbreviation, stop words removal, negation handling and finally feature selection.

5.2.1 Cleaning The Data

Data cleaning and word/text normalization are the main aspects of pre-processing. Especially the tweets have to be cleaned because the majority of them are unformatted written. We apply these methods of "cleaning" and word/text normalization through several techniques and phases in order to reach the appropriate form.

- Removal of URLs
Removing the URLs is important as the words they contain usually are unnecessary for the analysis and secondly there is a chance the mentioned sites to contain words that affect the polarity of the text.
- Removal of Retweets
Retweeting is the process of copying another user's tweet and posting to another account. This usually happens if a user likes another user's tweet. Retweets are commonly abbreviated with RT. For example, consider the following tweet: "RT @LEGO_DarkTower: I

have 2 versions of Lego.”

- Lowercase conversion
All letters are converted to lowercase.

- Stop words removal
A list of of words and symbols is constructed with useless information. This list contains words that do not affect the sentiment (like 'the' and 'end'), several punctuation marks as well as many symbols.

- Filtering
Usually people use repeated letters in words like happyyyy to show their intensity of expression. But, these word are not present in the our corpuses hence the extra letters in the word must be eliminated. This elimination follows the rule that a letter cannot repeat more than three times hence can eliminate such letter.

- Tokenization
The method of tokenizing words is applied that we have already described in (Section 4.1). This method also includes the removal of punctuation marks as well as the conversion to lowercase. Also, we made use of both unigrams and bigrams.

5.2.2 Word and Text Normalization

This method gives emphasis on the meaning of each word and the entire text as well as the sentiment that is observed from it. This is achieved by the followings, which we have implemented.

- Lemmatization

Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to expand abbreviation and remove inflectional endings only and to return the base or dictionary form of a word (Figure 5.3), which is known as the lemma.

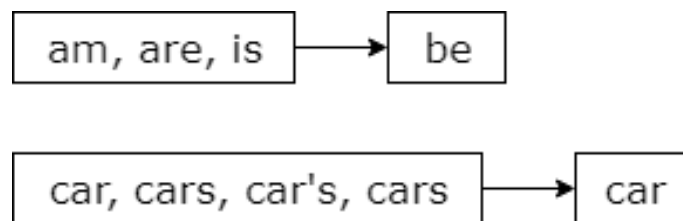


Figure 5.3: Lemmatization's form

The result of this mapping of text will be something like this (Figure 5.4):

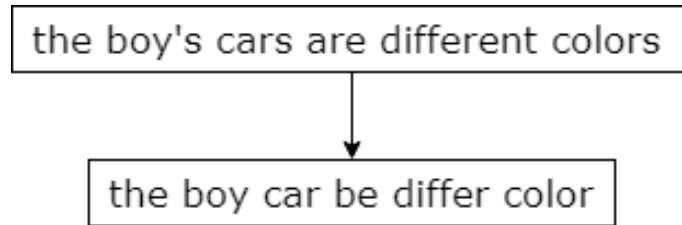


Figure 5.4: Example of lemmatization

- Part Of Speech Tagging - POS Tagging

The last step of data cleaning and data normalization to its correct form is to take care and give consideration on the position of its word in the sentence as well the meaning behind every proposal. Each word must be assigned to its correct part-of-speech, such as noun, verb, adjective, or adverb, based on its function in a sentence. Since the goal is to analyze the sentiments and classify them into three classes namely positive, negative and neutral, we consider the adjective and noun combinations. Nouns are entities and adjectives are the qualifiers, giving more weightage to their combinations would bring out the positive, the negative or even the polarity of the sentences. There are adjectives that convey positive polarity. This would give the feature values of positive movie reviews a distinction as compared to the feature values of the adjectives conveying a negative sentiment. Thus the classifier would be trained to classify positive and negative

opinions based on the features having more weights.

In this diploma this is illustrated using the Combinational approach, Lexicon Based (Section 4.3) and Rule Based approach (Section 4.4), which has been presented in Section 4.6.

So, regarding the implementation, a function called "CountVectorizer" has been built that includes all the above techniques.

5.2.3 Data Transformation

After the data cleaning, the data transformation takes place so as to convert words to features/vectors. There are several ways to assess the importance of each feature by attaching a certain weight in the text. The most popular ones are: Feature Frequency (FF), Term Frequency Inverse Document Frequency (TF-IDF), and Feature Presence (FP). FF is the number of occurrences in the document. TF-IDF is given by

$$TF - IDF = FF * \log(N/DF) [38]$$

where N indicates the number of documents, and DF is

the number of documents that contains this feature [29]. FP takes the value 0 or 1 based on the feature absent or presence in the document.

In text classification, a text document may partially match many categories. We need to find the best matching category for the text document. The term (word) frequency/inverse document frequency (TF-IDF) approach is commonly used to weigh each word in the text document according to how unique it is. In other words, the *TF-IDF* approach captures the relevancy among words, text documents and particular categories.

VSM assumes that a text document d is represented by a set of words (t_1, t_2, \dots, t_n) where in each t_i is a word that appears in the text document d_i , and n denotes the total number of various words used to identify the meaning of the text document. Word t_i has a corresponding weight w calculated as a combination of the statistics $TF(w_i, d_i)$ and $IDF(w_i)$. Therefore, d can be represented as a specific n -dimensional vector d_i as

$$d_i = (w_1, w_2, \dots, w_n)$$

Weight is the measure that indicates the statistical importance of corresponding words. The weight w_i as of word t_i as can be determined by the value of $TF(w_i, d_i) * IDF(w_i)$.

The TF value is proportional to the frequency of the word in the document and the IDF value is inversely proportional to its frequency in the document corpus.

The function encodes the intuitions that:

1. The more often a word occurs in a document, the more it is representative of the content of the text.
2. The more text the word occurs in, the less discriminating it is.

Per word vector d_i commonly contains a lot of vector elements. In order to reduce the vector dimension, we shall select elements by calculating the value of $TF*IDF$ for each element in the vector representation. The words selected as vector elements are called feature words by us. In documents, the higher-frequency words are more important for representing the content than lower-frequency words. However, some high-frequency words such as "the", "for", "at" having low content discriminating power are listed at the stop-list.

As is observed, in this diploma we use the $TF-IDF$ technique and we create the $TF-IDF$ frequency term matrix (freq_term_matrix) (Figure 5.5) which then is transformed to $TF-IDF$ weight matrix (tf_idf_weight matrix) (Figure 5.6):

(0, 25629)	2
(0, 25802)	1
(0, 25975)	2
(0, 26241)	1
(0, 27985)	1
:	:
(4318, 26055)	2
(4318, 27297)	1
(4318, 28433)	1

Figure 5.5: Sample of term's frequency matrix

(1, 23844)	0.147853815984
(1, 20644)	0.193610906629
(1, 19162)	0.171132741371
(1, 18358)	0.154244426309
(1, 18213)	0.0762211067415
:	:
(4319, 27991)	0.232374491104
(4319, 27318)	0.409148789697
(4319, 26071)	0.22967969779

Figure 5.6: Sample of weight matrix

5.3 The Classification Process

Our aim in this work was to examine whether it suffices to treat sentiment classification simply as a special case of topic-based categorization (with the three “topics” being positive sentiment, negative sentiment and neutral senti-

ment). This is done by using the training data set that we have already created (Section 5.1). This then trains the classifier we choose to be able to predict the three sentiments [24]. So, after the transformation of the data to vectors we continue by applying the classifiers. We experimented with the below machine learning algorithms:

- The Multinomial Naive-Bayes classifier
- The Decision-Tree classifier
- The Random-Forest classifier
- The K-Neighbors classifier
- The Support Vector Machine classifier

The philosophies behind these five algorithms are quite different, but each has been shown to be effective. But, out of them, the Multinomial Naive-Bayes gave the better results and higher scores. This is the classifier that we chose for the integration of our analysis and we discuss about it in this section. The other classifiers with their results are analyzed in further section.

5.3.1 The Multinomial Naive-Bayes Classifier

We introduce the Multinomial Naive-Bayes classifier, so called because it is a Bayesian classifier that makes a simplifying (naive) assumption about how the features interact. Naive Bayes is a probabilistic classifier [27], meaning that for a document d , out of all classes $c \in C$ the classifier returns the class \hat{c} which has the maximum posterior:

$$\hat{c} = \underset{c \in C}{\operatorname{arg\,max}} P(c|d)$$

So, next we derive the Naive Bayes (NB) classifier by first observing that by Bayes' rule,

$$P(c|d) = \frac{P(c)P(d|c)}{P(d)},$$

where $P(d)$ plays no role in selecting c . To estimate the term $P(d|c)$, Naive Bayes decomposes it by assuming the f_i 's are conditionally independent given d 's class:

$$P_{NB} := \frac{P(c)(\prod_{i=1}^m P(f_i|c))^{n_i(d)}}{P(d)}$$

Our training method consists of relative-frequency estimation of $P(c)$ and $P(f_i|c)$.

Despite its simplicity and the fact that its conditional independence assumption clearly does not hold in real-world situations, Naive Bayes Based text categorization still tends to perform surprisingly well as we will see in the next section.

5.4 Performance Evaluation

In the previous section we described and illustrated our classifier. After that, this classifier was trained with the training data set and then we used the test data set, in fact our tweets, so as to predict their sentiments. So, in this section we present and analyze results obtained from the experiments to evaluate the effectiveness of the proposed method by using various evaluation metrics namely accuracy, precision, recall and f-score [17]. Those metrics are computed based on the values of true positive (tp), false positive (fp), true negative (tn) and false negative (fn) assigned classes.

- **Accuracy:** It measures the percentage of the items that the system detected correctly made divided by the total number of predictions made. Accuracy is defined as:

$$Accuracy = \frac{tp+tn}{tp+fp+tn+fn}$$

- **Precision:** It measures the percentage of the items that the system detected (i.e., the system labeled as positive) that are in fact positive. Precision is defined as:

$$Precision(p) = \frac{tp}{tp+fp}$$

- **Recall:** It measures the percentage of items actually present in the input that were correctly identified by the system. Recall is defined as:

$$Recall(r) = \frac{tp}{tp+fn}$$

- **F-score:** It is a weighted method and a combination of precision and recall into a single metric. F-score is defined as:

$$F\text{-score} = \frac{2(p)(r)}{(p)+(r)}$$

The results of our metrics are the followings(Table 5.1):

Accuracy-score	0.722517935663 (72%)
Precision-score	0.708851826558 (70%)
Recall-score	0.782715186932 (78%)
F-score	0.734796218266 (73%)

Table 5.1: Classifier's metrics

From the table we can observe that the scores for this classifier are good enough reaching 72% of the accuracy score. Although accuracy might seem a natural metric, in general it considers to be not so good. That is, because accuracy does not work well enough when the classes are unbalanced (as indeed they are with spam, which is a large majority of email, or with tweets). Thus, being more interested in the other two metrics, precision score and recall score, are really good with 70% and 78% respectively.

5.5 Different Classifiers

In this section we describe several classifiers with mathematical formulas that we mentioned in (Section 5.3) and we present their performance (accuracy score, precision score, recall score, f-score). Then a comparison takes place between these classifiers. We used the scikit-learn library in order to apply them.

- **The Decision-Trees classifier:** Decision Tree classifier uses a decision tree. In these trees [22], leaves represent class labels and branches represent features which lead to those class labels. In other words, each node corresponds to one of the input variables and each leaf represents a value of the target variable given the values of the input variables represented by the path from the root to the leaf.

In fact, a decision tree is a simple representation for classifying examples. They can be described as the combination of mathematical and computational techniques to aid the description, categorization of a given data set.

Data comes in records of the form:

$$(x, Y) = (x_1, x_2, \dots, x_k, Y)$$

The vector x is composed of the features x_1, x_2, x_3 etc. while the dependent variable Y is the target variable that we are trying to understand and classify.

- **The Random Forest classifier:** Regarding the Random Forest algorithm, given a training set $X = x_1, \dots, x_n$ with responses $Y = y_1, \dots, y_n$ bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples:

For $b = 1, \dots, B$ [9]:

1. Sample, with replacement, n training examples from X, Y ; called X_b, Y_b .
2. Train a classification or regression tree f_b on X_b, Y_b .

After training, predictions for samples x' can be made by taking the average of the predictions from all the individual regression trees on x' :

$$\hat{f} = \frac{1}{B} \sum_{b=1}^B f_b(x')$$

This process leads to better model performance because it decreases the variance of the model, without increasing the bias. This means that while the predictions of a single tree are highly sensitive to noise in its training set, the average of many trees is not, as long as the trees are not correlated. Simply training many trees on a single training set would give strongly correlated trees.

The number of samples/trees, B , is a free parameter. Typically, a few hundred to several thousand trees are used, depending on the size and nature of the training set.

- **The K-Nearest Neighbors classifier:** In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive

integer, typically small). If $k = 1$, then the object is simply assigned to the class of that single nearest neighbor [26].

The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples. In the classification phase, k is a user-defined constant, and an unlabeled vector (a point) is classified by assigning the label which is most frequent among the k training samples nearest to that query point. A commonly used distance metric for continuous variables is Euclidean distance. For discrete variables, such as for text classification, another metric can be used, such as the Hamming distance.

For example, the most intuitive nearest neighbour type classifier is the one nearest neighbour classifier that assigns a point x to the class of its closest neighbour in the feature space,

$$C_n^{1nn}(x) = Y_1$$

- **The Linear Support Vector Machines classifier:** Support Vector Machines (SVMs) are useful for data classification. It finds a separating hyperplane with the maximal margin between two classes of data [16]. Given a set of instance-label pairs $x_i, y_i, x_i \in R^n$,

$y_i \in \{1, -1\}, i = 1, \dots, l$ SVM solves the following unconstrained optimization problem:

$$\min_{w, b} \frac{1}{2} w^T w + C \sum_{i=1}^l \xi(w, b; x_i, y_i)$$

where $\xi(w, b; x_i, y_i)$ is a loss function, and $C \geq 0$ is a penalty parameter on the training error. Two common loss functions are:

$$\max(1 - y_i(w^T \phi(x_i) + b), 0)$$

and

$$(\max(1 - y_i(w^T \phi(x_i) + b), 0))^2$$

where ϕ is a function that mapped training data into higher dimensional space. The former is called L1-loss SVM, and the latter is L2-loss SVM. When participating in the challenge, we choose the L2-loss function. Post-challenge experiments show that the two loss functions result in similar performances.

For any testing instance x , the decision function (predictor) is [44]:

$$f(x) = \text{sgn}(w^T \phi(x) + b)$$

Practically, a kernel function $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ may be used to train the SVM. A linear SVM has

$\phi(x) = x$ so the kernel function is $K(x_i, x_j) = x_i^T x_j$. Another popular kernel is the radial basis function (RBF):

$$K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2), \text{ where } \gamma > 0.$$

We use linear SVM for classification. For each value of C or (C, γ), we conduct validation on the training set, and choose the parameters leading to the highest accuracy.

Below (Table 5.2) we can see the metrics of all the classifiers mentioned as well as the comparison of each metric of these classifiers :

Classifiers	Accuracy Score	Precision Score	Recall Score	F-Score
Multinomial-NB	0.722518 (72%)	0.708852 (71%)	0.782715 (78%)	0.734796 (73%)
Decision-Tree	0.471419 (47%)	0.713544 (71%)	0.359722 (36%)	0.270129 (27%)
Random-Forest	0.478130 (48%)	0.707188 (71%)	0.370889 (37%)	0.293382 (29%)
K-Nearest Neighbors	0.463319 (46%)	0.533085 (53%)	0.362899 (36%)	0.363033 (36%)
Linear-SVM	0.686184 (69%)	0.778061 (78%)	0.614656 (61%)	0.654909 (65%)

Table 5.2: Metrics of different classifiers

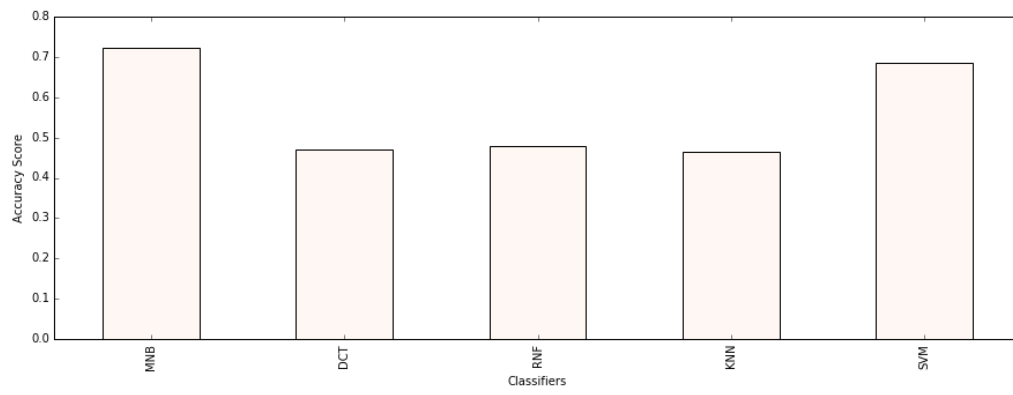


Figure 5.7: Accuracy vs Classifiers

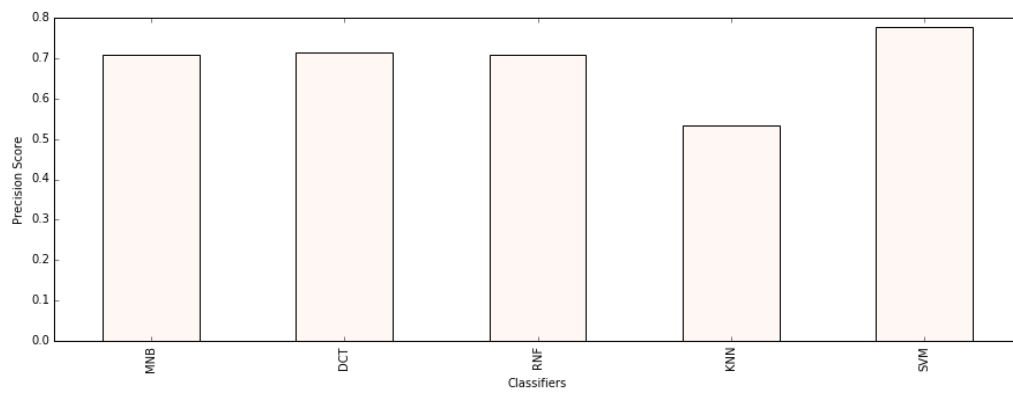


Figure 5.8: Precision vs Classifiers

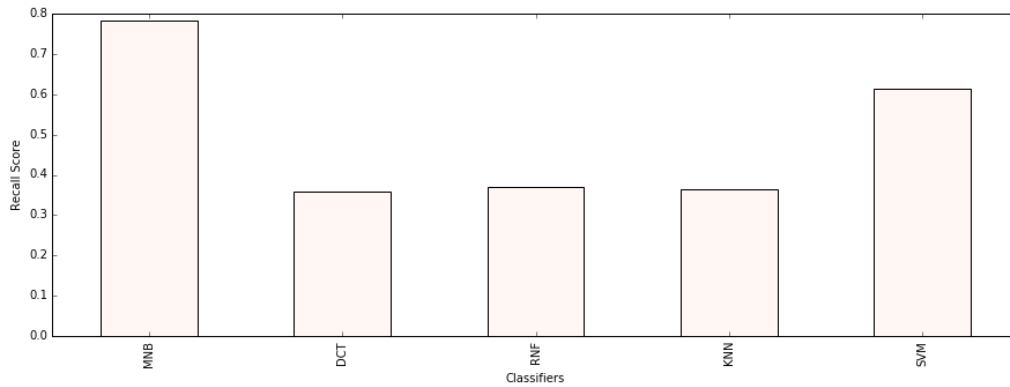


Figure 5.9: Recall vs Classifiers

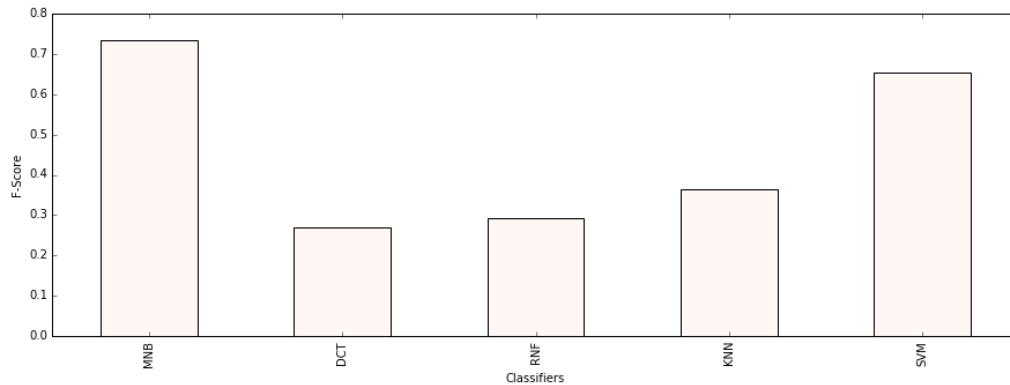


Figure 5.10: F-score vs Classifiers

As is shown from the table and figures, the precision score is the highest score of each classifier especially for the Linear Support Vector Machine classifier. The other metrics range in smaller scales. However, we can see that the Multinomial Naive-Bayes classifier reaches the best scores than the others. Thus, undoubtedly for our model this is the ideal classifier and justifiably this the one we chose.

Chapter 6

Twitter Application

This chapter illustrates the main idea of this diploma thesis. To begin with, the main goal was the building of an application through which the user could acquire knowledge about a movie. This knowledge is based on comments and reactions of the Twitter's users. So, our intention is to give the ability to the user to know not only the number of users of the Twitter that found this movie good or not but also how positive or negative was the reaction of each one on it. We followed this approach and a sample of a web interface was constructed. The application has not been completed to its final form yet. Hence, an innovative technique on how to build a functional operation is presented in the Section 7.1, in which our future work is illustrated. So, through it, each user will be able to evaluate a movie based on public opinion and watch it if its opinion fits.

As a first step, the user enters an electronic platform which will ask the movie that would like to watch. For example, the movie that the user is typing is "The Dark Tower (2017)" (Figure 6.1).

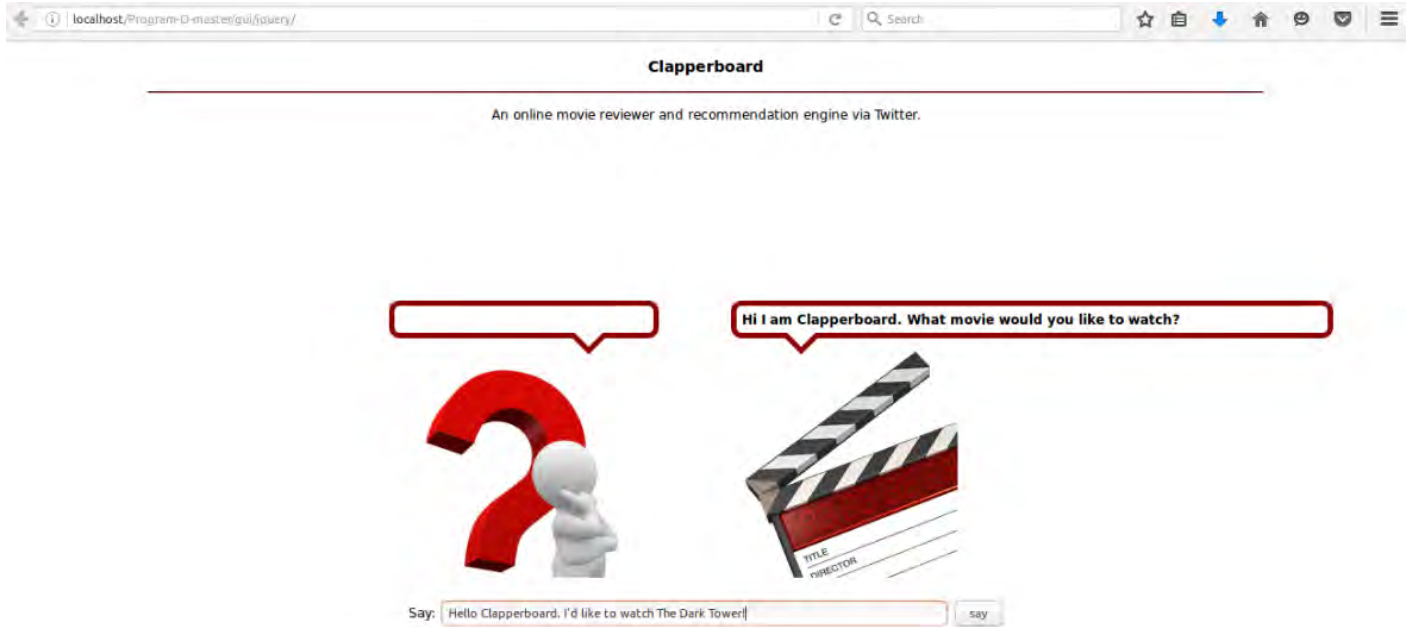


Figure 6.1: The Twitter application environment (Clapperboard)

Once the movie has been typed, the application starts collecting data from the Twitter related to movie, placing as a criterion of finding "theDarkTower". In conjunction, having obtained the data, the classification begins to take place on the basis of Multinomial-NB as described in Chapter 5.3. The result of this process will be the categorization of the tweets to positives, negatives and neutrals, as well as how much positive, negative and neutral they are.

Below (Figure 6.2) we can see a part of the categorized tweets which are presented to the user.

10	I was wondering when we might start hearing ab...	negative	2.1
11	RT @robinfurth: Hi everybody--I loved the DT f...	neutral	8
12	Super looking forward to Tom talk music about ...	positive	3.8
13	RT @WIRED: .@McConaughey & @IdrisElba answ...	neutral	9.5
14	RT @LEGO_DarkTower: Here's the first guide on ...	negative	2.5
15	RT @DarkTowerMovie: "Did you tell the kid that...	positive	3.3
16	@Sonia_KaMai You'll not have lost much. On the...	positive	0.3
17	@Sonia_KaMai You'll prob spend more \$\$ on coff...	negative	1
18	RT @gonefullgeek: Over 4,000 pages of pure, wr...	neutral	5.4
19	RT @FutureChriss: 3 days to #TheDarkTower ...	negative	0.9
20	RT @DarkTowerMovie: Don't miss these stars in ...	negative	3.4

Figure 6.2: Sample of the categorized tweets

More specific, for example, we see that the tweet *12*: *"Super looking forward to Tom talk music about the Dark-TowerMovie <https://t.co/tb0AESUYDe>"* has been categorized as positive. We can confirm that it is actually positive simply by reading it. Also, the tweet *17* has been correctly categorized as negative. However, the tweet *19*: *"RT @FutureChriss: 3 days to TheDarkTower ... happy for those who will watching in that time, sad for us international fans. 🙏"* although it is positive has been wrongly categorized as negative due to the non-sufficient accuracy of our model.

Moreover, a CSV file with all the extracted tweets is given to the user in order to read it if it is desirable.

Finally, a table is presented to the user with all the tweets categorized in three classes, positive, negative, neutral, as well as a pie chart with the percentage quantities. For our example, "The Dark Tower" movie we extracted 200 tweets and we took the following results (Figure 6.3) and (Figure 6.4):

	sentiment	number
0	positive	24
1	negative	33
2	neutral	142

Figure 6.3: Summary of the categorized tweets

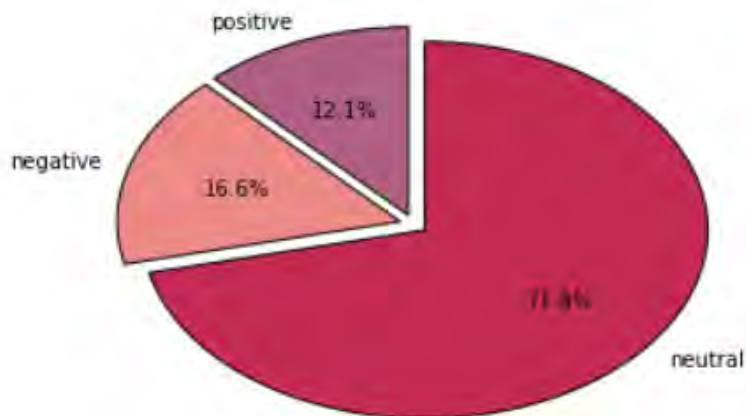


Figure 6.4: Pie-chart of the categorized tweets (%)

Chapter 7

Conclusions

Ending what we did in this diploma-thesis was to analyze and test various techniques for sentiment analysis, in order to identify the positive, negative and neutral opinions, among which are the Lexicon Based and Rule Based approaches. After all the steps we performed, we conclude that Python's tool, Vader, guarantees better results by combining the two above-mentioned techniques.

In addition, we developed our own machine learning algorithms. More specific, we built various classifiers, which we trained with a data set related to movie reviews, concluding that the Multinomial Naive-Bayes was the one that made more fits in our model. Thus, by creating our own application, we allowed the user to choose a movie of its preference. And then, after acquiring the tweets related to this movie, we categorized them in positive, negative and

neutral with the help of Multinomial Naive-Bayes.

This whole process has enabled us to give the user the ability to get information through commentary and reactions at the Twitter for the movie which would like to watch.

It should also be noted that due to its non-sufficiency of classifier, specifically 72%, the classification of tweets had some deviations, that is, it was not 100% accurate. For example, while the classifier categorized multiple tweets in the right class, we noticed that some others had been placed in the wrong class, they were given the wrong label. This is not something that diminishes the importance of the results of our research. Instead it gives us a lot of room for improvement as future work.

7.1 Future Work

This work will obviously continue to get even better results. One possible optimization is the classifier's improvement we use, as mentioned above. This can be achieved in two ways:

- either by further exploring other classifiers that will have a better fit in our data,

- or by creating a large data set - it was emphasized that the larger the set is, the more accurate the results are - using Vader (Section 4.6).

This will allow us to implement a fairly accurate sentiment analysis, namely on tweets, which have some specific features (emoticons, handwritten language). If this is achieved, something that would even produce better results is to build a data set, in order to train our classifier, which will now consists entirely of tweets and not of movies reviews.

In addition, as an optimization, we suggest the creation of an integrated application using the *MongoDB* (Database) and *Django* (Web framework for perfectionists with deadlines). By incorporating the code written in Python we target, combined with a more friendly user interface, to develop greater intimacy with the user. This can be achieved with Django, a Python web framework.

The following diagram (Figure 7.1) illustrates how each of the components of the Model-View-Template pattern interacts with each other to serve a user request.

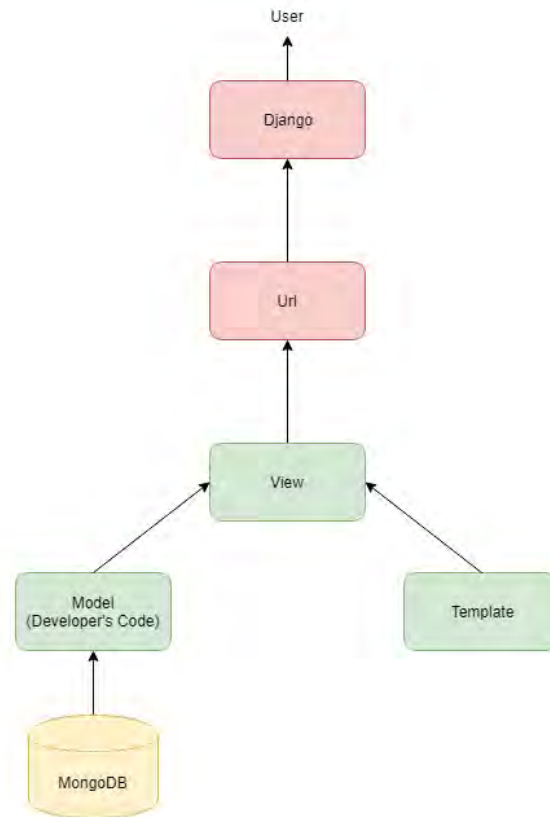


Figure 7.1: Architecture diagram of the application

The template is an HTML file mixed with Django Template Language (DTL). The developer provides the Model (Software Code that controls the interactions between the Model and View), the view and the template then just maps it to a URL and Django serves it to the user. A model is a class that represents table or collection in our Database, and where every attribute of the class is a field of the table or collection. Django makes it possible to separate Python and HTML, the Python goes in views and

HTML goes in templates. To link the two, Django relies on the render function and the Django Template language. A view function, or "view" for short, is simply a Python function that takes a web request and returns a web response. After we have a working view, we can access that view via a URL. Django has his own way for URL mapping.

Finally, the goal is that the application can then suggest to the user various movies based on the films which have been searched for so far. That is, a recommendation engine tailored to the options of each account.

Bibliography

- [1] Statista 2017. *Number of monthly active Twitter users worldwide from 1st quarter 2010 to 3rd quarter 2017 (in millions)*. URL: <https://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users>.
- [2] Mariam Adedoyin-Olowe, Mohamed Medhat Gaber, and Frederic Stahl. “A survey of data mining techniques for social media analysis”. In: *arXiv preprint arXiv:1312.4617* (2013).
- [3] Rasim M Alguliev and Ramiz M Aliguliyev. “Effective summarization method of text documents”. In: *Web Intelligence, 2005. Proceedings. The 2005 IEEE/WIC/ACM International Conference on*. IEEE. 2005, pp. 264–271.
- [4] new gen apps. *The Secret Behind Measuring Customer Emotions: Social Media Sentiment Analysis*. URL: <https://www.newgenapps.com/blog/the-secret-way-of-measuring-customer-emotions-social-media-sentiment-analysis>.
- [5] Cambridge. *Supervised and unsupervised learning*. URL: <http://beta.cambridgespark.com/courses/jpm/01-module.html>.
- [6] Cagatay Catal and Mehmet Nangir. “A sentiment classification model based on multiple classifiers”. In: *Applied Soft Computing* 50 (2017), pp. 135–141.
- [7] CNN. *NASA astronaut first to 'tweet' from space*. URL: <http://edition.cnn.com/2009/TECH/space/05/13/twitter.space>.
- [8] Ann Devitt and Khurshid Ahmad. “Sentiment polarity identification in financial news: A cohesion-based approach”. In: *Proceedings of the 45th annual meeting of the association of computational linguistics*. 2007, pp. 984–991.

- [9] Sanjeev Dhawan, Kulvinder Singh, and Ashu Jain. “Friend Recommendation System for Online Social Networks using Cohesion Based Approach”. In: *Journal of Network Communications and Emerging Technologies (JNCET) www.jncet.org* 7.9 (2017).
- [10] Xiaowen Ding, Bing Liu, and Philip S Yu. “A holistic lexicon-based approach to opinion mining”. In: *Proceedings of the 2008 international conference on web search and data mining*. ACM. 2008, pp. 231–240.
- [11] Jørgen Faret and Johan Reitan. “Twitter Sentiment Analysis-Exploring the Effects of Linguistic Negation”. MA thesis. NTNU, 2015.
- [12] Mohamed Abdel Fattah. “New term weighting schemes with combination of multiple classifiers for sentiment analysis”. In: *Neurocomputing* 167 (2015), pp. 434–442.
- [13] Usama M Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, et al. “Knowledge Discovery and Data Mining: Towards a Unifying Framework.” In: *KDD*. Vol. 96. 1996, pp. 82–88.
- [14] Rodolfo Ferro. *Sentiment analysis on Trump’s tweets using Python*. URL: <https://dev.to/rodolfoferro/sentiment-analysis-on-trumpss-tweets-using-python->.
- [15] Alejandro Guerra-Hernández. *Explorations of the BDI Multi-agent support for the Knowledge Discovery in Databases Process*. URL: https://www.researchgate.net/figure/236373188_Fig-1-Steps-in-the-KDD-process.
- [16] Emma Haddi, Xiaohui Liu, and Yong Shi. “Least Squares Support Vector Machine Classifiers”. In: *Neural Processing Letters* 9 (1999), p. 293.
- [17] Emma Haddi, Xiaohui Liu, and Yong Shi. “The role of text pre-processing in sentiment analysis”. In: *Procedia Computer Science* 17 (2013), pp. 26–32.
- [18] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques*. 3rd. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011. ISBN: 0123814790, 9780123814791.
- [19] Daniel Harris. *The Best Free Tools for Twitter Sentiment Analysis*. URL: <https://www.softwareadvice.com/resources/free-twitter-sentiment-analysis-tools>.

- [20] Vasileios Hatzivassiloglou and Kathleen R McKeown. “Predicting the semantic orientation of adjectives”. In: *Proceedings of the 35th annual meeting of the association for computational linguistics and eighth conference of the european chapter of the association for computational linguistics*. Association for Computational Linguistics. 1997, pp. 174–181.
- [21] <http://www.statisticssolutions.com>. *Time Series Analysis*. URL: <http://www.statisticssolutions.com/time-series-analysis>.
- [22] Ronny Kohavi and J Ross Quinlan. “Data mining tasks and methods: Classification: decision-tree discovery”. In: *Handbook of data mining and knowledge discovery*. Oxford University Press, Inc. 2002, pp. 267–276.
- [23] Sotiris Kotsiantis and Dimitris Kanellopoulos. “Association rules mining: A recent overview”. In: *GESTS International Transactions on Computer Science and Engineering* 32.1 (2006), pp. 71–82.
- [24] A Kowcika et al. “Sentiment analysis for social media”. In: *International journal of advanced research in computer science and software engineering* 3.7 (2013), pp. 216–221.
- [25] Haewoon Kwak et al. “What is Twitter, a social network or a news media?” In: *Proceedings of the 19th international conference on World wide web*. ACM. 2010, pp. 591–600.
- [26] Baoli Li, Shiwen Yu, and Qin Lu. “An improved k-nearest neighbor algorithm for text categorization”. In: *arXiv preprint cs/0306099* (2003).
- [27] Andrew McCallum, Kamal Nigam, et al. “A comparison of event models for naive bayes text classification”. In: *AAAI-98 workshop on learning for text categorization*. Vol. 752. 1. Citeseer. 1998, pp. 41–48.
- [28] David Murphy. *44 Percent of Twitter Accounts Have Never Tweeted*. URL: <https://www.pcmag.com/article2/0,2817,2456489,00.asp>.
- [29] Jin-Cheon Na et al. “Effectiveness of simple linguistic processing in automatic sentiment classification of product reviews”. In: *I.C. McIlwaine (Ed.)* (2004), pp. 49–54.
- [30] Nielsen. *SOCIAL MEDIA REPORT 2012: SOCIAL MEDIA COMES OF AGE*. URL: <http://www.nielsen.com/us/en/insights/news/2012/social-media-report-2012-social-media-comes-of-age.html>.
- [31] Kalyani M Raval. “Data Mining Techniques”. In: *International Journal of Advanced Research in Computer Science and Software Engineering* 2.10 (2012).

- [32] PR Rejanimol, M Sharika, and N Arun Kumar. “Sentimental Analysis of Movie Reviews Based on Twitter Data”. In: *International Journal of Pure and Applied Mathematics* 114.11 (2017).
- [33] Richard Rogers et al. “Foreword: Debanalising Twitter: The transformation of an object of study”. In: *Twitter and Society*. New York: Peter Lang Publishing, pp. ix–xxvi 89 (2014), pp. 9–27.
- [34] scribd.com. *Data Mining*. URL: <https://www.scribd.com/document/342888747/Data-Mining-is-Defined-as-the-Procedure-of-Extracting-Information-From-Huge-Sets-of-Data>.
- [35] SlickRemix. *How to get API Keys and Tokens for Twitter*. URL: <https://www.slickremix.com/docs/how-to-get-api-keys-and-tokens-for-twitter/>.
- [36] *Stemming and lemmatization*. URL: <https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html>.
- [37] P.N. Tan et al. *Introduction to Data Mining, 2nd Edition*. What’s New in Computer Science Series. Pearson Education, 2018. ISBN: 9780133128901.
- [38] Huifeng Tang, Songbo Tan, and Xueqi Cheng. “A survey on sentiment detection of reviews”. In: *Expert Systems with Applications* 36.7 (2009), pp. 10760–10773.
- [39] Adeyinka Tella. *Social Media Strategies for Dynamic Library Service Development*. IGI Global, 2014.
- [40] WhatIs.com. *social media*. URL: <http://whatis.techtarget.com/definition/social-media>.
- [41] Gbolahan K Williams and Sarabjot Singh Anand. “Predicting the Polarity Strength of Adjectives Using WordNet.” In: *ICWSM*. San Jose, CA. 2009, pp. 346–349.
- [42] www.investopedia.com. *Regression*. URL: <https://www.investopedia.com/terms/r/regression.asp>.
- [43] Chin-Sheng Yang and Hsiao-Ping Shih. “A Rule-Based Approach For Effective Sentiment Analysis.” In: *PACIS*. 2012, p. 181.
- [44] Mohammed J Zaki, Wagner Meira Jr, and Wagner Meira. *Data mining and analysis: fundamental concepts and algorithms*. Cambridge University Press, 2014.

Appendix A

Tweet Extraction and Analysis

Data analysis libraries

```
import tweepy          # To consume Twitter's API
import nltk            # Python programs work with human language data
import pandas as pd    # To handle data
import numpy as np     # For number computing
```

Plotting and visualization libraries

```
from IPython.display import display
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Collecting the tweets

We import our access keys:

```
from credentials import * # This will allow us to use the keys as
                           # variables
```

API's setup:

```
def twitter_setup():
    """
    Utility function to setup the Twitter's API
    with our access keys provided.
    """
    # Authentication and access using keys:
    auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
    auth.set_access_token(ACCESS_TOKEN, ACCESS_SECRET)

    # Return API with authentication:
    api = tweepy.API(auth)
    return api
```

Selection of a specific movie:

```
print('What is the movie you would like to watch?')
movie = input()
movie1 = movie.replace(" ", "")
```

We create a tweet list as follows:

```
tweets = extractor.user_timeline(screen_name=movie1, count=100)
print("Number of tweets extracted: {}".format(len(tweets)))
```

We print the most recent 5 tweets:

```
print("5 recent tweets:\n")
for tweet in tweets[:5]:
    print(tweet.text)
    print()
```

We create a pandas data frame as follows:

```
data = pd.DataFrame(data=[tweet.text for tweet in tweets],
                    columns=['text'])
```

We display the first 10 elements of the data frame:

```
display(data.head(10))
```

Some analysis on the extracted tweets [14]

Internal methods of a single tweet object:

```
print(dir(tweets[0]))
```

We print info from the first tweet:

```
print(tweets[0].id)
print(tweets[0].created_at)
print(tweets[0].source)
print(tweets[0].favorite_count)
print(tweets[0].retweet_count)
print(tweets[0].geo)
print(tweets[0].coordinates)
print(tweets[0].entities)
```

We add relevant data:

```
data['len'] = np.array([len(tweet.text) for tweet in tweets])
data['ID'] = np.array([tweet.id for tweet in tweets])
data['Date'] = np.array([tweet.created_at for tweet in tweets])
data['Source'] = np.array([tweet.source for tweet in tweets])
data['Likes'] = np.array([tweet.favorite_count for tweet in tweets])
data['RTs'] = np.array([tweet.retweet_count for tweet in tweets])
```

Display of the first 10 elements from the data frame::

```
display(data.head(10))
```

We extract the mean of lengths:

```
mean = np.mean(data['len'])
print("The length's average in tweets: {}".format(mean))
```

We extract the tweet with the most FAVs (likes) and the most RTs (retweets):

```
fav_max = np.max(data['Likes'])
rt_max = np.max(data['RTs'])

fav = data[data.Likes == fav_max].index[0]
rt = data[data.RTs == rt_max].index[0]

# Max FAVs:
print("The tweet with more likes is: {}".format(data['text'][fav]))
print("Number of likes: {}".format(fav_max))
print("Number of characters: {}".format(data['len'][fav]))
print("\n")
# Max RTs:
print("The tweet with more retweets is: {}".format(data['text'][rt]))
print("Number of retweets: {}".format(rt_max))
print("Number of characters: {}".format(data['len'][rt]))
```

We create time series for data:

```
tlen = pd.Series(data=data['len'].values, index=data['Date'])
tfav = pd.Series(data=data['Likes'].values, index=data['Date'])
tret = pd.Series(data=data['RTs'].values, index=data['Date'])
```

Lengths along time:

```
tlen.plot(figsize=(16,4), color='r')
plt.ylabel("Length of tweets")
plt.show()
```

Likes vs retweets visualization:

```
tfav.plot(figsize=(16,4), label="Likes", legend=True)
tret.plot(figsize=(16,4), label="Retweets", legend=True)
plt.ylabel("Likes vs Retweets")
plt.show()
```

Real-time location of the tweets:

```
tweet_coordinates = pd.read_csv('tweet_coordinates.csv',
                                encoding='utf-8')

from mpl_toolkits.basemap import Basemap

# plot the blank world map
plt.figure(figsize = (12,6))
my_map = Basemap(projection='merc', lat_0=50, lon_0=-100,
                  resolution = 'l', area_thresh = 5000.0,
                  llcrnrlon=-140, llcrnrlat=-55,
                  urcrnrlon=160, urcrnrlat=70)

# draw elements onto the world map
my_map.drawcountries()
#my_map.drawstates()
my_map.drawcoastlines(antialiased=False,
                      linewidth=0.005)

# add coordinates as red dots
longs = list(tweet_coordinates.lon)
latts = list(tweet_coordinates.lat)
x, y = my_map(longs, latts)
my_map.plot(x, y, 'ro', markersize=6, alpha=0.5)
```

```
plt.show()
```

Appendix B

Data Sentiment - Vader

We download the Vader:

```
nltk.download('vader_lexicon')
```

An example of the use of the Vader and how we can use it in order to label our unlabeled data set:

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer

sent = 'you are not very good'

sid = SentimentIntensityAnalyzer()

ss = sid.polarity_scores(sent)

print(ss)
ss['compound']
print("Compound is the mean average, so the score here is negative:
      {}".format(ss['compound']))
```

We import our training dataset:

```
train_tweets = pd.read_csv("training.csv", encoding = "cp1252")
train_tweets = pd.DataFrame(train_tweets.text, columns=['text'])
```

We transform to the suitable format (Dataframe):

```
data_sentiment = pd.DataFrame(columns=['text', 'sentiment'])
```

We calculate the polarity of each extracted tweet:

```
i = 0
for _, tweet in train_tweets.iterrows():
    i += 1
    try:
        data_sentiment.loc[i-1, 'text'] = tweet.values[0]
        ss = sid.polarity_scores(tweet.values[0])
        data_sentiment.loc[i-1, 'sentiment'] = ss['compound']

    except Exception as e:
        sentiments.loc[i-1, 'text'] = tweet.values[0]
```

We display the first 10 elements of the data frame:

```
from IPython.display import display
display(data_sentiment.head(10))
```

We calculate the sentiment metric of each extracted tweet:

```
for i in range(len(train_tweets.text)):
    if(data_sentiment.sentiment[i] > 0):
        data_sentiment.sentiment[i] = 'positive'
    elif(data_sentiment.sentiment[i] < 0):
        data_sentiment.sentiment[i] = 'negative'
    else:
```

```
data_sentiment.sentiment[i] = 'neutral'  
display(data_sentiment.head(10))  
  
data_sentiment.to_csv('data_sentiment.csv', encoding='utf-8')
```

Appendix C

Pre-processing

Reading the data:

```
df = pd.read_csv('full_training_dataset.csv', sep=',',
                 names=['Sentiment', 'Text'], dtype=str, header=0,
                 encoding='latin-1')
```

Separating the data:

```
# Splitting the data set into 2 parts: train and test
from sklearn.cross_validation import train_test_split

train, test = train_test_split(df, test_size=0.2)
train.head()
test.head()
```

Pre-processing and extraction of the feature vectors from test-data

Importing stop words (most common words are going to be removed):

```
from nltk.corpus import stopwords

stop_words = set(stopwords.words('english'))
stop_words.update((" ", ":", "\", "=", "&", ";", "%", "$",
                  "@", "%", "^", "*", "(", ")", "{", "}",
                  "[", "]", "|", "/", "\\", ">", "<", "-",
                  "!", "?", ".", "'",
                  "--", "---", "#"))
```

Pre-processing with CountVectorizer (text to lowercase, stop-words, tokenization, lemmatization, etc.)

```
from sklearn.feature_extraction.text import CountVectorizer

CountVectorizer(analyzer='word', binary=False, decode_error='strict',
                encoding='utf-8', input='content',
                lowercase=True, max_df=1.0, max_features=None, min_df=1,
                ngram_range=(1, 2), preprocessor=None,
                stop_words=stop_words,
                strip_accents=None, token_pattern='(?u)\\b\\w+\\b',
                tokenizer=None, vocabulary=dict)

count_vectorizer = CountVectorizer()
train_vec = count_vectorizer.fit_transform(train.Text)
#print("Vocabulary:", count_vectorizer.vocabulary_)
```

Calculation of each feature's occurrence in the document:

```
# Term frequency - inverse document frequency
from sklearn.feature_extraction.text import TfidfTransformer

tfidf = TfidfTransformer(norm="l2")
freq_term_matrix = count_vectorizer.transform(test.Text)
tfidf.fit(freq_term_matrix)
```

```
# Sparse format of our test dataset
print(freq_term_matrix)
```

```
# Dense format of our test dataset
print(freq_term_matrix.todense())
```

Transform of the `freq_term_matrix` to the `tf_idf_weight` matrix:

```
tf_idf_matrix = tfidf.transform(freq_term_matrix)
```

```
# Sparse format of tf_idf (weight) matrix
print(tf_idf_matrix)
```

```
# Dense format of td_idf (weight) matrix
print(tf_idf_matrix.todense())
```

Appendix D

The Classification Process

Training the Multinomial Naive-Bayes classifier with the train data set:

```
from sklearn.naive_bayes import MultinomialNB

nb = MultinomialNB()
print(nb)
classifier_nb = MultinomialNB(class_prior=[0.25, 0.50,
    0.25]).fit(train_vec, train['Sentiment'])
```

Selecting the best classifier:

```
import pickle

save_classifier = open("naivebayes.pickle", "wb")
pickle.dump(classifier_nb, save_classifier)
save_classifier.close()
```

Predicting the sentiment of each tweet from the test data set:

```
y_pred_nb = classifier_nb.predict(tf_idf_matrix)
```

Evaluating the classifier

Calculating the accuracy score of the Multinomial Naive-Bayes in order to evaluate the model:

```
from sklearn.metrics import accuracy_score

acc_score_nb = accuracy_score(test['Sentiment'], y_pred_nb,
                               normalize=True, sample_weight=None)
print(acc_score_nb)
```

Calculating other metrics (precision_score, recall_score, f-score):

```
from sklearn.metrics import precision_recall_fscore_support

metrics = precision_recall_fscore_support(test['Sentiment'],
                                          y_pred_nb, sample_weight=None, average='macro') #the scores for
                                          each class are returned
print("Precision Score, Recall Score, F-Score, Not Defined:
      \n{}".format(metrics))
```

More classification models

Training the Decision-Tree classifier with the train data set:

```

from sklearn.tree import DecisionTreeClassifier

Dtree = DecisionTreeClassifier()
print(Dtree)
classifier_Dtree = DecisionTreeClassifier().fit(train_vec,
        train['Sentiment'])

y_pred_Dtree = classifier_Dtree.predict(tf_idf_matrix)

print("\n")
acc_score_Dtree = accuracy_score(test['Sentiment'], y_pred_Dtree,
        normalize=True, sample_weight=None)
print("Accuracy Score: {}".format(acc_score_Dtree))
metrics = precision_recall_fscore_support(test['Sentiment'],
        y_pred_Dtree, sample_weight=None, average='macro') #the scores
        for each class are returned
print("Precision Score, Recall Score, F-Score, Not Defined:
        \n{}".format(metrics))

```

Training the Random-Forest classifier with the train data set:

```

from sklearn.ensemble import RandomForestClassifier

randomf = RandomForestClassifier()
print(randomf)
classifier_RF = RandomForestClassifier().fit(train_vec,
        train['Sentiment'])

y_pred_randomf = classifier_RF.predict(tf_idf_matrix)

print("\n")
acc_score_randomf = accuracy_score(test['Sentiment'],

```

```

    y_pred_randomf, normalize=True, sample_weight=None)
print("Accuracy Score: {}".format(acc_score_randomf))
metrics = precision_recall_fscore_support(test['Sentiment'],
    y_pred_randomf, sample_weight=None, average="macro") #the scores
    for each class are returned
print("Precision Score, Recall Score, F-Score, Not Defined:
    \n{}".format(metrics))

```

Training the K-Nearest-Neighbors classifier with the train data set:

```

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier()
print(knn)
classifier_NN = KNeighborsClassifier().fit(train_vec,
    train['Sentiment'])

y_pred_knn = classifier_NN.predict(tf_idf_matrix)

print("\n")
acc_score_knn = accuracy_score(test['Sentiment'], y_pred_knn,
    normalize=True, sample_weight=None)
print("Accuracy Score: {}".format(acc_score_knn))
metrics = precision_recall_fscore_support(test['Sentiment'],
    y_pred_knn, sample_weight=None, average="macro") #the scores for
    each class are returned
print("Precision Score, Recall Score, F-Score, Not Defined:
    \n{}".format(metrics))

```

Training the Support Vector Machine classifier with the train data set:

```

from sklearn import svm
from mlxtend.plotting import plot_decision_regions

classifier_SV = svm.LinearSVC().fit(train_vec, train['Sentiment'])

```

```

print(classifier_SV)
y_pred_svm = classifier_SV.predict(tf_idf_matrix)

print("\n")
acc_score_svm = accuracy_score(test['Sentiment'], y_pred_svm,
                                normalize=True, sample_weight=None)
print("Accuracy Score: {}".format(acc_score_svm))
metrics = precision_recall_fscore_support(test['Sentiment'],
                                           y_pred_svm, sample_weight=None, average="macro") #the scores for
                                           each class are returned
print("Precision Score, Recall Score, F-Score, Not Defined:
      \n{}".format(metrics))

```

Presenting the metrics of each classifier separately:

```

data = pd.read_table("thesis_data.txt", sep=',')
data = data.set_index("Classifiers")
data

```

Plots:

```

# Accuracy vs Classifiers
x_labels = ["Multinomial_NB", "Decision_Tree", "Random_Forest",
            "K_Nearest_Neighbors", "Linear_SVC"]

plt.figure(figsize=(15, 5))
ax = data["Accuracy Score"].plot(kind='bar', colormap="RdPu")
ax.set_ylabel("Accuracy Score")
ax.set_xticklabels(x_labels)
plt.show()

# Precision vs Classifiers
x_labels = ["Multinomial_NB", "Decision_Tree", "Random_Forest",
            "K_Nearest_Neighbors", "Linear_SVC"]

plt.figure(figsize=(15, 5))
ax = data["Precision Score"].plot(kind='bar', colormap="RdPu")
ax.set_ylabel("Precision Score")
ax.set_xticklabels(x_labels)

```

```
plt.show()

# Recall vs Classifiers
x_labels = ["Multinomial_NB", "Decision_Tree", "Random_Forest",
            "K_Nearest_Neighbors", "Linear_SVC"]

plt.figure(figsize=(15, 5))
ax = data["Recall Score"].plot(kind='bar', colormap="RdPu")
ax.set_ylabel("Recall Score")
ax.set_xticklabels(x_labels)
plt.show()

# F-score vs Classifiers
x_labels = ["Multinomial_NB", "Decision_Tree", "Random_Forest",
            "K_Nearest_Neighbors", "Linear_SVC"]

plt.figure(figsize=(15, 5))
ax = data["F-Score"].plot(kind='bar', colormap="RdPu")
ax.set_ylabel("F-Score")
ax.set_xticklabels(x_labels)
plt.show()
```

We create a pie chart with the percentage quantities of each class:

```
%matplotlib inline
colors = ["#af6182", "#ec8a87", "#c32c54"]

plt.pie(
    tag_number['number'],
    labels=tag_number['sentiment'],

    shadow=False,

    colors=colors,

    explode=(0.05, 0.05, 0.05),

    startangle=90,
```

```
    autopct='%1.1f%%',  
    )  
plt.show()
```

Appendix E

Twitter application

The file with the below code can be run from the terminal console and after typing a specific movie, the results about this are printed in the console.

Data analysis libraries

```
import tweepy          # To consume Twitter's API
import nltk            # Python programs work with human language data
import pandas as pd    # To handle data
import numpy as np     # For number computing
```

Plotting and visualization libraries

```
from IPython.display import display
import matplotlib.pyplot as plt
%matplotlib inline
```

We import our access keys:

```
from credentials import * # This will allow us to use the keys as
                           # variables
```

API's setup:

```
def twitter_setup():
    """
    Utility function to setup the Twitter's API
    with our access keys provided.
    """
    # Authentication and access using keys:
    auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
    auth.set_access_token(ACCESS_TOKEN, ACCESS_SECRET)

    # Return API with authentication:
    api = tweepy.API(auth)
    return api
```

We create an extractor object::

```
extractor = twitter_setup()
```

Selection of a specific movie:

```
print('What is the movie you would like to watch?')
movie = input()
movie1 = movie.replace(" ", "")
```

We create a tweet list as follows:

```
tweets = extractor.user_timeline(screen_name=movie1, count=1000)
print("Number of tweets extracted: {}".format(len(tweets)))
# For this movie there were only 200 tweets
```

We make the data frame::

```
tweets = pd.DataFrame(data=[tweet.text for tweet in tweets],
                      columns=['text'])
```

Pre-processing:

```
# Importing stop words (most common words are going to be removed)
from nltk.corpus import stopwords

stop_words = set(stopwords.words('english'))
stop_words.update((" ", ":", "\", "=", "&", ";", "%", "$",
                  "@", "%", "^", "*", "(", ")", "{", "}",
                  "[", "]", "|", "/", "\\", ">", "<", "-",
                  "!", "?", ".", "'",
                  "--", "---", "#"))

# Pre-processing with CountVectorizer (text to lowercase,
# stop-words, tokenization, etc.)
from sklearn.feature_extraction.text import CountVectorizer

# Extracting the feature vectors
from sklearn.feature_extraction.text import TfidfTransformer
count_vectorizer = CountVectorizer()
tfidf = TfidfTransformer(norm="l2")
```

We load the trained Multinomial Naive-Bayes classifier:

```
import pickle

classifier_f = open("naivebayes.pickle", "rb")
classifier = pickle.load(classifier_f)
classifier_f.close()
```

Categorization of the tweets to the three classes (positive, negative, neutral):

```

# Classify the tweets using the Multinomial Naive-Bayes
sentiments = pd.DataFrame(columns=['text', 'label', 'prob'])
i = 0
for _, tweet in tweets.iterrows():
    i += 1
    try:
        freq_term_matrix_tweet = count_vectorizer.transform(tweet)
        tfidf.fit(freq_term_matrix_tweet)
        tfidf_tweet = tfidf.transform(freq_term_matrix_tweet)
        sentiments.loc[i-1, 'text'] = tweet.values[0]
        sentiments.loc[i-1, 'label'] =
            classifier.predict(tfidf_tweet)[0]
        sentiments.loc[i-1, 'prob'] =
            round(classifier.predict_proba(tfidf_tweet)[0][1], 2)*10
    except Exception as e:
        sentiments.loc[i-1, 'text'] = tweet.values[0]

sentiments.to_csv('sentiments.csv', encoding='utf-8')
print(sentiments)

# Calculation of positives, negatives and neutrals
raw_data = {'sentiment': ['positive', 'negative', 'neutral'],
            'number': [0, 0, 0]}

tag_number = pd.DataFrame(raw_data, columns = ['sentiment',
            'number'])

for i in range(len(sentiments.label)):
    if(sentiments.label[i] == 'positive'):
        tag_number['number'][0] = tag_number['number'][0] + 1
    elif(sentiments.label[i] == 'negative'):
        tag_number['number'][1] = tag_number['number'][1] + 1
    else:
        tag_number['number'][2] = tag_number['number'][2] + 1

tag_number

```

The results we take from the movie:

```
print(movie, 'has', tag_number['number'][0], 'positive tweets,',  
      tag_number['number'][1], 'negative tweets and',  
      tag_number['number'][2], 'neutral tweets.')
```

```
#print(sentiments)
```
