A BRAND RANKING FRAMEWORK ON SOCIAL MEDIA

A Thesis
Presented to
The Academic Faculty

by

Alexandros Arvanitidis & Anna Serafi
(Registry Numbers 480 & 471)

Supervised by
Vakali Athena, Professor

Co-Supervised by
Tsoumakas Grigorios, Assistant Professor

Evaluation Committee member
Bassiliades Nick, Associate Professor

In Partial Fulfillment of the Requirements for the Degree
Postgraduate Diploma
in the
School of Computer Science

Aristotle University of Thessaloniki
February 2014

Copyright © 2014
You start dying slowly
if you do not travel,
if you do not read,
If you do not listen to the sounds of life,
If you do not appreciate yourself.

You start dying slowly
When you kill your self-esteem;
When you do not let others help you.

You start dying slowly
If you become a slave of your habits,
Walking everyday on the same paths...
If you do not change your routine,
If you do not wear different colours
Or you do not speak to those you don’t know.

You start dying slowly
If you avoid to feel passion
And their turbulent emotions;
Those which make your eyes glisten
And your heart beat fast.

You start dying slowly
If you do not change your life when you are not satisfied with your job, or with your love,
If you do not risk what is safe for the uncertain,
If you do not go after a dream,
If you do not allow yourself,
At least once in your lifetime,
To run away from sensible advice...

Pablo Neruda
Acknowledgements

We would never be able to finish our thesis without the guidance of some people throughout the process and we feel the need to thank those people, before we begin.

First, we would like to express our deepest gratitude to our advisor, Mrs Vakali Athena, for her excellent guidance, caring, patience, and providing us with an excellent atmosphere for doing research. Without her full support nothing of this would be possible.

We would also like to thank Mr Tsoumakas Grigorios, for his constant assistance throughout the semester and his willingness to patiently guide us through the parts that we had little knowledge of.

We would also like to thank the two doctoral candidates Maria Giatsoglou and Despina Hatzakou for guiding our research. They were always willing to help and give us their best suggestions, especially on technical parts of this work.

Special thanks goes to Bassiliades Nick, Associate Professor of the Department, who was willing to participate in our final evaluation committee.

We would also like to thank our families and friends. They were always supportive and encouraged us with their best wishes, even though we weren’t that consistent to our social obligations the past semester.

Sincerely,

Alexandros Arvanitidis

Anna Serafi
CONTENTS

ACKNOWLEDGEMENTS ......................................................... iii
LIST OF TABLES ................................................................. vii
LIST OF FIGURES ................................................................. viii
LIST OF SYMBOLS OR ABBREVIATIONS ................................. ix
ABSTRACT ........................................................................... ix
EXTENDED ABSTRACT .......................................................... x

1 INTRODUCTION ................................................................. 1
  1.1 Problem definition ......................................................... 1
  1.2 Motivation of this thesis .................................................. 1
  1.3 Objective of this thesis .................................................... 2
  1.4 Structure of this thesis .................................................... 2

2 THEORETICAL BACKGROUND ............................................. 3
  2.1 Machine Learning .......................................................... 3
    2.1.1 Sentiment Analysis ................................................... 6
    2.1.2 Preprocessing .......................................................... 7
  2.2 Multi Criteria Decision Analysis ....................................... 10
    2.2.1 Introduction to Multiple Criteria Decision Analysis - MCDA 10
    2.2.2 TOPSIS - Description of used MCDA method ............... 14
  2.3 Review of Literature ..................................................... 17
# TECHNICAL FRAMEWORKS AND TOOLS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Twitter Micro-blogging Service</td>
<td>22</td>
</tr>
<tr>
<td>3.1.1</td>
<td>The History of Twitter Service</td>
<td>23</td>
</tr>
<tr>
<td>3.1.2</td>
<td>Twitter Components</td>
<td>25</td>
</tr>
<tr>
<td>3.1.3</td>
<td>Data Collection via Twitter API</td>
<td>26</td>
</tr>
<tr>
<td>3.2</td>
<td>Machine Learning Tools</td>
<td>27</td>
</tr>
<tr>
<td>3.3</td>
<td>The Mongo DataBase Framework</td>
<td>29</td>
</tr>
<tr>
<td>3.4</td>
<td>Multi-threaded Programming Principles</td>
<td>31</td>
</tr>
<tr>
<td>3.5</td>
<td>The Klout Service as a Social Influence Evaluator</td>
<td>32</td>
</tr>
</tbody>
</table>

# BUILDING A BRAND RANKING FRAMEWORK

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Defining a brand page</td>
<td>35</td>
</tr>
<tr>
<td>4.2</td>
<td>Defining Features</td>
<td>36</td>
</tr>
<tr>
<td>4.3</td>
<td>Designing the Database Model</td>
<td>36</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Brand Collection</td>
<td>36</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Features Collection</td>
<td>40</td>
</tr>
<tr>
<td>4.4</td>
<td>The Architecture Behind the Framework</td>
<td>42</td>
</tr>
</tbody>
</table>

# IMPLEMENTATION

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Preliminary Procedure</td>
<td>43</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Collecting the Preliminary Required Data</td>
<td>43</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Preparing WEKA Training Set</td>
<td>46</td>
</tr>
<tr>
<td>5.2</td>
<td>Collecting and Storing the Data</td>
<td>48</td>
</tr>
<tr>
<td>5.3</td>
<td>Extracting Features from Stored Data</td>
<td>52</td>
</tr>
<tr>
<td>5.4</td>
<td>Visualizing Data on a Web Application</td>
<td>53</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Front end</td>
<td>53</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Back-end</td>
<td>55</td>
</tr>
<tr>
<td>5.5</td>
<td>Framework Limitations</td>
<td>59</td>
</tr>
</tbody>
</table>
6 DISCUSSION AND FUTURE WORK ......................... 61

Appendix A — SAMPLES OF PROGRAM’S CODE .................. 63

REFERENCES ............................................................... 64
List of Tables

1 Decision Matrix ................................................. 13
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Supervised learning model</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>The SVM representation</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>Model of the MongoDB system</td>
<td>30</td>
</tr>
<tr>
<td>4</td>
<td>Example of a MongoDB document</td>
<td>30</td>
</tr>
<tr>
<td>5</td>
<td>Representation of a document in Brand Collection</td>
<td>38</td>
</tr>
<tr>
<td>6</td>
<td>Representation of a tweet</td>
<td>39</td>
</tr>
<tr>
<td>7</td>
<td>Framework architecture</td>
<td>41</td>
</tr>
<tr>
<td>8</td>
<td>Implementation structure</td>
<td>43</td>
</tr>
<tr>
<td>9</td>
<td>Annotation of tweets for the training set</td>
<td>48</td>
</tr>
<tr>
<td>10</td>
<td>Annotation of tweets for the training set</td>
<td>48</td>
</tr>
</tbody>
</table>
# List of Symbols or Abbreviations

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application programming interface.</td>
</tr>
<tr>
<td>BSON</td>
<td>Binary JSON.</td>
</tr>
<tr>
<td>CSS</td>
<td>Cascading Style Sheets.</td>
</tr>
<tr>
<td>DOM</td>
<td>Document Object Model.</td>
</tr>
<tr>
<td>eWOM</td>
<td>electronic Word of Mouth.</td>
</tr>
<tr>
<td>GNU</td>
<td>General Public License.</td>
</tr>
<tr>
<td>HTML</td>
<td>HyperText Markup Language.</td>
</tr>
<tr>
<td>IMC</td>
<td>integrated marketing communications.</td>
</tr>
<tr>
<td>MCDA</td>
<td>Multiple Criteria Decision Analysis or Multiple Criteria Decision Making (MCDM).</td>
</tr>
<tr>
<td>MVC</td>
<td>Model-view-controller.</td>
</tr>
<tr>
<td>SNSs</td>
<td>social networking sites.</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machines.</td>
</tr>
<tr>
<td>TF/IDF</td>
<td>Term Frequency/Inverse Document Frequency.</td>
</tr>
<tr>
<td>TOPSIS</td>
<td>Technique for Order of Preference by Similarity to Ideal Solution.</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator.</td>
</tr>
<tr>
<td>WEKA</td>
<td>Waikato Environment for Knowledge Analysis.</td>
</tr>
</tbody>
</table>
ABSTRACT

A BRAND RANKING FRAMEWORK ON SOCIAL MEDIA

Alexandros Arvanitidis & Anna Serafi
(Registry Numbers 480 & 471)

Supervised by Vakali Athena, Professor
Co-Supervised by Tsoumakas Grigoris, Assistant Professor
Evaluation Committee member Bassiliades Nick, Associate Professor

In the competitive world of famous brands, strong presence in social media is of major importance for such large companies to ensure customer engagement and to advertise new products easily on the web. On that path, many systems have been built, in order to provide such functionality to famous brands. The end-users can watch their company’s web profile, see statistics or make comparisons between other companies. In this project Twitter streaming API is used to track 400 famous brands on Twitter, keeping important data about them, that show their interaction with users or what individual users think about the company. After measuring 3 categories of social impact factors (subjective, objective, metrics), a score is assigned to each brand via a multi-criteria algorithm. The brands are then ranked accordingly. The result of this work is a web application which visualises the list of the most social brands on Twitter and allows the user to be informed of important statistics. Different case studies show that the social presence of a brand is influenced by those impact factors.
Extended Abstract

In the competitive world of famous brands, strong presence in social media is of major importance for such large companies to ensure customer engagement and to advertise new products easily on the web. On that path, many systems have been built, in order to provide such functionality to famous brands.

On current systems the end-users can watch their company’s web profile, see statistics or make comparisons between other companies. However, such systems are mostly commercial and don’t provide the end-user the functionality he needs. Usually, the statistics that are shown come from a black-box analysis, which doesn’t allow the user to parameterise his query. Brands often want to rank themselves among other brands on certain fields, and see where they are better or worse, what part of their social presence they need to empower or what they already do better than their opponent. On that path, we wanted to create an open ranking framework of social web profiles, where the end-user could play with the parameters he needs and adjust them according to his needs.

For example, a user may need to compare his brand’s profile on the number of followers on Twitter and Likes on Facebook, with other companies. This is usually the most common scenario and is commonly provided on social web ranking platforms. If though, the comparison were to be made on the number of positive posts found about that user, along with the number of followers, then a more thorough approach is made, that, we proved, has totally different results than the previous query.
In this project Twitter streaming API is used to track 400 famous brands on Twitter, keeping important data about them, that show their interaction with users or what individual users think about the company. After measuring 3 categories of social impact factors (subjective, objective, metrics), a score is assigned to each brand via a multi-criteria algorithm, which receives as an input rows of numbers, representing each brand, and weights for each criterion.

The result of this work is a web application which visualises the list of the most social brands on Twitter and allows the user to be informed of important statistics. The interface also provides the user with a weight selection table, where he can query the database, on different parts. Thus, the framework, as mentioned before, allows parameterisation and shows that an analysis can vary a lot, when it’s based on so many different factors, as there exist on social media. The different weights as input, provide the end-user with functionality that never existed before. As a social web tool, we showed that the ranking procedure changes dramatically, when different input data is being given to the algorithm.

On our results section we showcased various use cases, where the importance of such a framework is shown.
Introduction

1.1 Problem definition

Social media clearly had a huge impact on the way the world wide web works. One can even say that they are now the focal point of the web. Given the way people interact with each other, celebrities being a part of it, trends being set within minutes from the actual event, things are flowing towards a more “social” web everyday.

Advertisers and brand strategists in general, have made successful efforts to promote products, approach new customers or even sell stuff through social media. To this path, nothing could be more helpful, than a social media statistics framework, where one could view certain values such as “Likes” in Facebook, “Retweets” in Twitter or “+1s” in Google+, or even more complex statistics, such as comparison between brands or elements that one could improve, in order to be competitive in the business world.

1.2 Motivation of this thesis

Various web applications have been created, in order to solve this issue, most of which are commercial applications, which require a monthly fee, therefore a clear image of what such systems do is not an easy thing to acquire. Furthermore, such systems require statistics on-demand, which means that one has to pay and clearly state the social web profiles that he wants to be followed. Another issue we discovered, was that current comparisons between brands’ social web profiles are made, using only one parameter, which is how many followers (in Twitter) or likes (in Facebook) a brand has.
Our motivation was the lack of non-commercial systems to track social media statistics for all brands and provide a ranking between them, as well as the current way brands are ranked amongst others, which only takes into account the people that follow their profile.

1.3 Objective of this thesis

The objectives of this thesis are:

1. To provide a social web presence framework, for brands to see how their social presence is rated and why.

2. To prove that using only one parameter (followers of profile) does not give the best outcome, as there are many factors that influence a brand’s popularity on social media, such as how many people talk about this brand and how many of them have a positive attitude towards that brand.

1.4 Structure of this thesis

This thesis is structured as follows:

- Chapter 2 contains the theoretical background behind this thesis. This chapter presents all the related work too.

- Chapter 3 describes the tools that were used and provides a short overview for each tool.

- Before presenting practical part of this thesis Chapter 4 describes the architecture and the database model of the framework. This chapter also defines the main components of this project.

- In Chapter 5 the implementation of the whole project is thoroughly described, how its components were created and what each one achieves. One can also read the limitations we faced and future work that could be done

- A discussion is made in Chapter 6, where the results are being processed and the outcome is described.

- In the Appendices section various components are being described more thoroughly.
In this section a historical overview of the all the related to this thesis fields of study will be presented. A part of this thesis is based on Machine Learning techniques, while another part is related with Multi Criteria Decision Analysis. After presenting a more general theoretical background, we focus on related works found in the literature. After reading this chapter the motivation behind this project will be then clear to the reader.

2.1 Machine Learning

Theories about learning, have existed on philosopher’s books since Aristotle’s time. Learning by humans comes at a rather young age, when an infant starts interacting with the environment and tries to understand how it works. By copying this human procedure, computer scientists have created tools to imitate the human behaviour on machines. Tom M. Mitchell provided a widely quoted, more formal definition [22].

**Definition 1.** “A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E”

In simpler words, Machine learning is the science of getting computers to act on their own, given some data and working with them, or improving them along the process. Over the years machine learning has been increasingly used on many different applications, such as self-driving cars, speech recognition, effective web search and generally in places where
the improvement of the system by learning is a huge step towards the improvement of the precision of its results.

A rather simple example of machine learning in practice is the 20Q game\textsuperscript{1}, based on the spoken parlor game known as twenty questions. 20Q asks the player to think of something, and the platform will try to find that by asking 20 questions. Using artificial intelligence (neural networks), the platform develops each time the game is played, by improving the guesses it makes, given the past ones. Usually after a user finishes the game, the platform asks if its prediction was right or wrong, and if it’s wrong what the correct answer was. Using this flow, machine learning algorithms try to predict the future answers, by looking up at the database of the previous answers and matching the current game with the closest entries in the database. It then makes a guess among these entries, to answer with the closest entry related to the current game.

Machine learning algorithms can be organized into a taxonomy based on the desired outcome of the algorithm or the type of input available, during the training process of the machine. Two of these algorithms are supervised and unsupervised learning, on which we’ll elaborate more for the purpose of this thesis.

- **Supervised Learning**
  Supervised learning is the machine learning task of inferring a function from labelled training data. The training data consists of a set of training examples, which is a representation of the values (usually a vector) leading to a result and the desired result itself. For example, given a vector of a being that is an animal, is a mammal, has a tail, has pointy ears and can be brown, one can guess that it’s a dog. The ideal scenario of supervised learning is the ability to foresee the results of unseen instances. The process of supervised learning can be seen on Figure 1. The teacher creates the training examples and the system learns by trying to achieve the desired response.

  The model can be improved by every interaction of the algorithm with a new instance, where the result is known. That can be done through a training process by human beings, as in the example of 20Q above, where new users can be considered as new instances and can train the algorithm by providing the result they were expecting the algorithm to guess. Supervised learning can also occur through sensors, where the machine can see a representation of a world and the results, as for example when someone sees clouds and birds flying low, a possible result might be rainy weather.

\textsuperscript{1}20 questions game on page \url{http://www.20q.net/}
CHAPTER 2. THEORETICAL BACKGROUND

An algorithm used extensively for supervised learning is SVM. SVMs are supervised learning models, used for classification and regression analysis. Given a training set, where the result belongs to one of two categories, an SVM algorithm classifies new incoming instances to one category or the other, making it a non-probabilistic binary classifier. An SVM model is a representation of the instances as points in space, divided by a clear gap that is as wide as possible. New instances are represented on the space and are classified depending on which side of the gap they are as one can see on Figure 2. The original SVM algorithm was invented by Vladimir N. Vapnik and the current standard incarnation (soft-margin) was proposed by Corinna Vortes and Vapnik in 1995.

Figure 1: Supervised learning model.

Figure 2: The SVM representation. The clear gap, divides the data into two categories (-/+).
• **Unsupervised learning**

  On the contrary unsupervised learning is the process of finding hidden structure in unlabelled data. This scenario occurs, for example when one wants to group instances into teams (clusters), where instances on the same team are more similar to each other. Since the instances given to the algorithm do not contain a result, there is no error or reward signal to evaluate the data. Unsupervised learning results mostly reflect the statistical structure of the collection, rather than an analysis. Therefore, the analyst can exploit structural details of the unlabelled data, that can improve the desired result or show patterns that weren’t visible otherwise [10].

2.1.1 **Sentiment Analysis**

Sentiment Analysis, also known as opinion mining refers to the use of natural language processing, text analysis and computational linguistics to identify and extract subjective information in source materials. In general, sentiment analysis is used, to determine the attitude of a writer, sometimes even attempting to identify certain feelings, e.g. anger, despair, happiness etc.

Over the years, sentiment analysis has been practised through many different implementations. Most of them can be categorized into 3 approaches: Lexicon-based, Machine-learning and in combination.

• **Lexicon-based**

  Lexicon-based sentiment analysis uses a static approach to analyse the given text and assign a polarity to it, given the sentiment each word represents. There are various databases of lexicons that scientists use, such as WordNet, Profile of Mood States (POMS, McNair 1971) as used on [4], or Sentiword Net, which was used on this thesis.

• **Machine Learning approach**

  Using machine-learning methods scientists have outperformed lexicon-based algorithms, as the classifier can be trained to achieve higher precision, especially when the sentiment result is restricted to positive, negative, neutral. Such algorithms have been reported to achieve a precision of 84% using SVMS [5]. The precision of the algorithm, though, highly depends on the context it is used on. It’s been reported, that given different contexts, the algorithm’s precision may vary depending on how it’s been trained to respond to every situation.
• Combination

The combination of the above approaches has been successfully applied by scientists, to exploit the advantages of both ways. On most papers, data is being processed using a lexicon as an initial pass. Then, the newly processed data are fed into a classifier, to improve the precision of the training data.

Such algorithms have been used for simple approaches as in [23], or more advanced techniques such as the objectiveness within tweets [25] or even the identification of sarcasm [15].

Due to the rise of social media, sentiment analysis has been fueled with new interest. Online reviews, ratings, comments or blog posts have been feeding data to opinion mining about certain subjects, or even products and brands. Some approaches have already considered the replacement of traditional polling, with online opinion mining using the approaches mentioned above [3]. Businesses have turned opinion mining into a kind of virtual currency of their products, as they can easily extract the opinion of the crowd on many levels. There are already many commercial systems doing exactly that, keeping a social web profile of a brand, providing a reputation score and statistics taken by opinion mining methods. On the other hand, many research teams in universities all around the world currently focus on understanding the dynamics of sentiment in e-communities, and how that reflects the company’s financial status. However, such methods contain a lot of noise, since it’s difficult to improve the precision of sentiment prediction even for humans, as text on the web can be hard to read, contain sarcasm etc. unless a more thorough approach is used, where text syntax and grammar are also part of the algorithm analysis.

2.1.2 Preprocessing

Preprocessors are systems that receive input data and provide new data as input to another program. Usually preprocessing is used to structure the data in a way, that is consistent to the desired structure that the new algorithm will use.

On machine learning, preprocessing methods are used, in order to improve precision of the classifier. Various techniques can be applied to the input data, some of which are the following:
• Stemming
Stemming is the process of reducing a word to its stem, base or root form, generally a written word form. For example, an english stemmer’s output to the word “science” would be “scienc”. One of the most popular stemmers was written by Martin Porter on July 1980, which came to be a de facto standard algorithm used for English stemming, also known as Porter Stemmer. Many interpretations of the original algorithm were created, however many of these implementation contained faults. For the source of error to be eliminated, Martin Stemmer released an official free software implementation of his algorithm\(^2\) and published it around the year 2000.

• Stop words
Any group of words can be chosen as the stop words for a given purpose. For some search machines, these are some of the most common, short function words, such as the, is, at, which, and on. In this case, stop words can cause problems when searching for phrases that include them, particularly in names such as “The Who”, “The The”, or “Take That”. Other search engines remove some of the most common words including lexical words, such as “want” from a query in order to improve performance. Hans Peter Luhn, one of the pioneers in information retrieval, is credited with coining the phrase and using the concept.

• TF/IDF
TF/IDF, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in information retrieval and text mining. The TF/IDF value increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus, which helps to control for the fact that some words are generally more common than others.

Variations of the TF/IDF weighting scheme are often used by search engines as a central tool in scoring and ranking a document’s relevance given a user query. TF/IDF can be successfully used for stop-words filtering in various subject fields including text summarisation and classification.

One of the simplest ranking functions is computed by summing the TF/IDF for each query term; many more sophisticated ranking functions are variants of this simple model.

Various ways for determining the exact values of both statistics exist. In the case

\(^2\)Snowball algorithm on page http://snowball.tartarus.org/
of the term frequency \( tf(t, d) \), the simplest choice is to use the raw frequency of a term in a document, i.e. the number of times that term \( t \) occurs in document \( d \). If we denote the raw frequency of \( t \) by \( f(t, d) \), then the simple \( tf \) scheme is \( tf(t, d) = f(t, d) \). Other possibilities include:

- Boolean “frequencies”:\( tf(t, d) = 1 \) if \( t \) occurs in \( d \) and 0 otherwise;
- logarithmically scaled frequency: \( tf(t, d) = \log(f(t, d) + 1) \)
- augmented frequency, to prevent a bias towards longer documents, e.g. raw frequency divided by the maximum raw frequency of any term in the document:

\[
tf(t, d) = 0.5 + \frac{0.5 \times f(t, d)}{\max\{f(w, d) : w \in d\}} \tag{1}
\]

The IDF is a measure of whether the term is common or rare across all documents. It is obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient.

\[
idf(t, D) = \log \frac{N}{|[d \in D : t \in d]|} \tag{2}
\]

with

- \( N \): total number of documents in the corpus
- \( |[d \in D : t \in d]| \): number of documents where the term \( t \) appears (i.e., \( tf(t, d) \neq 0 \)). If the term is not in the corpus, this will lead to a division-by-zero. It is therefore common to adjust the formula to \( 1 + |[d \in D : t \in d]| \).

Mathematically the base of the log function does not matter and constitutes a constant multiplicative factor towards the overall result.

Then TF/IDF is calculated as

\[
 tfidf(t, d, D) = tf(t, d) \times idf(t, D) \tag{3}
\]

A high weight in TF/IDF is reached by a high term frequency (in the given document) and a low document frequency of the term in the whole collection of documents; the weights hence tend to filter out common terms. Since the ratio inside the idf’s log function is always greater than or equal to 1, the value of idf (and TF/IDF) is greater than or equal to 0. As a term appears in more documents, the ratio inside the logarithm approaches 1, bringing the idf and TF/IDF closer to 0.
• **Document Length Normalization**
  This procedure is used to flatten the differences between the various sizes of the documents. If for example a word is found 50 times in a document of 3000 words, the frequency is 50, whereas if a word is found once in a document of 6 words the frequency is 1. After the normalisation occurs the new ratios are \frac{50}{3000}, \frac{1}{6}.

• **Tokenization**
  Tokenization is the process of dividing the input text into tokens:
  
  – words, which can have further morphological analysis and belong to a certain syntactic class
  – character(s) with recognisable structure, e.g. punctuation, numbers, dates

On text preprocessing the tokenization process is usually used to gather n-grams, which are continuous sequences of n-items from a given sequence of text. An n-gram of size 1 is referred to as “unigram”, size 2 is a bigram, size 3 is a trigram etc. N-grams are a preferred way to analyse text, since whole sentences or expressions can be derived, that have a certain meaning. E.g. the expression “The sun is shining” is usually used in a positive way, while its words alone don’t reflect the same feeling.

### 2.2 Multi Criteria Decision Analysis

When making decisions, people deal with multiple conflicting criteria that need to be evaluated. One common example could be car purchasing where cost, comfort, safety, and fuel consumption may be some of the main criteria someone takes into account. It is a difficult decision to make since seldom the cheapest car is the most comfortable and safe. Multiple criteria problems are met from such daily life decisions to a variety of different types of decision problems in disciplines such as mathematics, energy management, environmental planning, public services, healthcare, logistics, marketing, finance, software engineering, computer science, artificial intelligence and evolutionary computation.

#### 2.2.1 Introduction to Multiple Criteria Decision Analysis - MCDA

MCDA is an advanced field of operations research and management science which can be traced back to the 18th century and it has gained momentum as one of the fastest growing areas of research during the last decades. MCDA field of study describes various methods developed for aiding decision makers in reaching better decisions when addressing
problems involving multiple criteria goals or objectives of often conflicting nature. Many MCDA methods have been developed in the literature starting from the early 60s and continuing to the present day. Some of these methods have been studied by Triantaphyllou in 2000 and 2006 [31][34].

According to Triantaphyllou et al., MCDA methods can be categorized based on the type of data they use and the number of decision-makers involved in the decision process. Triantaphyllou also indicates the kind of analysis each method performs as another way of categorization. This is something which was previously described by another researcher, Bernard Roy (1985) [28]. Roy introduced the term “problematic” to describe the goal of each MCDA method. This term is known as the first and perhaps most important step in the study of a multi-criteria decision making problem. According to Roy there are four different kinds of analysis that can be performed in order to provide significant support to decision-makers. Roy’s “problematics” $P.\alpha$, $P.\beta$, $P.\gamma$, $P.\delta$, written in greek by definition, can be described as follows.

- **$P.\alpha$ (choice)**
  Identify the best alternative out of a set of alternatives. This method is based in making relevant comparisons.

- **$P.\beta$ (classification/sorting)**
  Classify/Sort the alternatives to predefined homogeneous classes, which are given an order of preference. In deriving the outcome, this method is based in making absolute comparisons. Each alternative is classified based on specific rules and standards which separate the categories. In this way, the classification is determined by the prescribed standards, and not by the total number of available alternatives.

- **$P.\gamma$ (ranking)**
  Rank the alternatives from the best to the worst ones. This method is based in making relevant comparisons. Thus, the selection is determined by the set of existing alternatives.

- **$P.\delta$ (description)**
  Describe the extend at which each alternative meets all the criteria simultaneously. In other words, this method identifies the major distinct features of the alternatives and perform their description based on these features.

The first three approaches (choice, ranking, classification/sorting) lead to a specific evaluation outcome [37]. For example,
• The selection of an investment project in a financial decision-making problem could be an example of P.\( \alpha \) (choice).

• Predicting a business failure when classifying firms as healthy or failed is an example of P.\( \beta \) (classification/sorting).

• An example of P.\( \gamma \) (ranking) could be the comparative evaluation and ranking of stocks according to their financial and stock market performance.

• The description of the financial characteristics of a set of firms is an example of P.\( \delta \) (description).

Although different MCDA methods follow different procedures, most of them share the following common characteristics.

• **Alternatives**
The alternatives represent the different options available to the decision maker. For instance, the alternatives might be car models in a car purchasing decision problem or the brands in the ranking problem of this thesis. In a multi-criteria decision making problem all alternatives are carefully checked, given priorities and therefore graded.

• **Multiple Decision Criteria**
Each multi-criteria decision making problem is associated with multiple decision criteria (or objectives or properties). They represent the different dimensions on which alternatives may be considered. There are cases where the number of criteria is large. In these cases, the criteria can be classified in a hierarchy where some criteria have a greater significance than others. So each main criterion may be assigned to sub-criteria, depending on the methodology used. Although some MCDA methods clearly define a hierarchical structure on the criteria of a decision problem, most of them consider a single level of criteria without hierarchies.

For the above car models alternatives, example criteria could be the cost, comfort, safety, and the fuel consumption one could take into account in order to decide the car he would purchase.

• **Different Measurement Units**
Different criteria can be associated with different measurement units. In the car example, the criteria cost and miles per litre of fuel are measured in units of money and
Table 1: Structure of a Typical Decision Matrix

<table>
<thead>
<tr>
<th>Alternative X₁</th>
<th>Criterion₁, w₁</th>
<th>Criterion₂, w₂</th>
<th>...</th>
<th>Criterionₙ, wₙ</th>
</tr>
</thead>
<tbody>
<tr>
<td>x₁₁</td>
<td>x₁₁₂</td>
<td>...</td>
<td></td>
<td>x₁ₙ</td>
</tr>
<tr>
<td>Alternative X₂</td>
<td>x₂₁</td>
<td>x₂₂</td>
<td>...</td>
<td>x₂ₙ</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Alternative Xₘ</td>
<td>xₘ₁</td>
<td>xₘ₂</td>
<td>...</td>
<td>xₘₙ</td>
</tr>
</tbody>
</table>

kilometres, respectively. So having to deal with different measurement units, multi-criteria decision making problems are inherently difficult to solve. In most cases the different criteria must be normalized.

- **Criteria Weights**

  In most of MCDA methods criteria are ascribed with weights of importance. Typically, these weights are normalized too.

The alternatives and decision criteria can be viewed as the entries of a decision matrix (see table 1). A decision matrix is a \((x_{ij})_{m \times n}\) matrix in which the element \(x_{ij}\) represents the performance of the \(i_{th}\) alternative \(X_i\) when evaluated with respect to a \(j_{th}\) criterion \(C_j\) (for \(i = 1, 2, 3, ..., m\) and \(j = 1, 2, 3, ..., n\)). It is also assumed that the decision maker has already determined the weights of relative importance of decision criteria (shown as \(w_j\), for \(j = 1, 2, 3, ..., n\) in table 1).

After presenting these basic principles of MCDA, we can now describe the multi-criteria decision making problem, which was previously referred, in the below definition (Zimmermann 1991) [36]:

**Definition 2.** Let \(X = \{x_i, \text{ for } i = 1, 2, 3, ..., m\}\) be a finite set of decision alternatives and \(C = \{c_j, \text{ for } j = 1, 2, 3, ..., n\}\) a finite set of objectives under which a decision will be drawn. Define the optimal alternative \(X^*\) with the maximum degree of preference out of all relevant objectives \(c_j\).

Having described the aim of MCDA, Roy’s “problematics” and the decision matrix, it is important to briefly introduce some of the known MCDA methods.

- **Weighted sum model (WSM)**

  [WSM on Wikipedia](http://en.wikipedia.org/wiki/Weighted_sum_mode)
CHAPTER 2. THEORETICAL BACKGROUND

- Analytic hierarchy process (AHP)\(^4\)
- PROMETHEE outranking method\(^5\)
- ELECTRE family of outranking methods\(^6\)
- Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS)\(^7\)

Since the MCDA is exceeding the scope of this master thesis the interested reader may want to consult the relevant Wikipedia page\(^8\) for more information. As part of this project TOPSIS MCDA method was selected as the most appropriate and will be presented in the following section. This choice was made after relevant research in the literature. The advantages of this method and the fact that suits the nature of our decision making problem led to this selection.

2.2.2 TOPSIS - Description of used MCDA method

The TOPSIS MCDA method, was originally developed by Hwang and Yoon in 1981 [16]. The basic idea of TOPSIS is rather straightforward since it simultaneously finds the shortest geometric distance to a best artificial alternative and the furthest from the worst one. The best alternative is the one with the best level for all attributes considered and the worst alternative, the one that has the worst attribute values. TOPSIS selects the best alternative to be the one that has the shortest geometric distance to the best alternative and is furthest from the worst alternative. TOPSIS simultaneously considers the distances to both best and worst alternative, in order to assign a ranking score to each alternative as a combination of these two distance measures.

TOPSIS is a good MCDA method compared to others, since it is based on a strategy which imitates the logic of human selection. TOPSIS has simple computational complexity (Kim et.al 1997) [19] and the only subjective inputs needed are the weights applied by the decision makers [24]. These advantages made TOPSIS an attractive method that has been applied to a major number of applications for solving real-world problems. For instance, some case studies found in the literature, and cover a variety of industrial areas, are the following:

• **Manufacturing**
  1. A performance measurement system which assists managers of advanced manufacturing systems to identify new activity investment opportunities [19].
  2. A system for the selection of the manufacturing process for a company that produces semiconductors [6].

• **Transportation**
  1. A section procedure among three European high-speed systems [17].
  2. A performance ranking procedure of highway buses of four companies [12].

• **Oil industry**
  A system for the selection of the best combat system regarding accidents with oil spill in the sea. This case study conducted for a large Brazilian oil reservoir [20].

• **Business**

• **Training systems**
  A system for the selection of the optimal training aircraft for the Air Force Academy in Taiwan [33].

• **Defense systems**
  A system for the selection of weapons as an efficiency measurement of defense systems [9].

• **Management**
  Prioritization of Strengths, Weaknesses, Opportunities and Threats strategies in a SWOT analysis in management decision making [13].

The above highlight TOPSIS wide range and flexibility justifying our selection of this method for our problem workaround.
TOPSIS Algorithm

The algorithm of TOPSIS MCDA method is composed in six steps.\(^9\)

**Input:** m alternatives and n criteria and a value of importance (weight) for each criterion.

1. Construct the decision matrix \(X(x_{ij})_{m \times n}\) consisting of the m alternatives and n criteria. Let \(x_{ij}\) be the score of alternative \(i\) with respect to criterion \(j\).

2. Construct the normalized decision matrix \(R(r_{ij})_{m \times n}\) with the following formula:

\[
    r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x_{ij}^2}}, \quad i = 1, 2, \ldots, m, \quad j = 1, 2, \ldots, n
\]

The normalization procedure allows comparisons across criteria since criteria might be in various dimensions.

3. Construct the weighted normalized decision matrix \(T(t_{ij})_{m \times n}\) by multiplying each column of the normalized decision matrix \(R\) by its associated weight as follows:

\[
    T = (t_{ij})_{m \times n} = (w_j r_{ij})_{m \times n}, \quad i = 1, 2, \ldots, m
\]

Where \(w_j\) is the normalized weight value of each criterion:

\[
    w_j = W_j / \sum_{j=1}^{n} W_j, \quad j = 1, 2, \ldots, n
\]

and \(W_j\) is the original weight given to the each criterion.

4. Identify the best artificial alternative (extreme performance on each criterion) \(A_b\).

\[
    A_b = \{\langle \min(t_{ij}|i = 1, 2, \ldots, m)\rangle | j \in J_-\}, \{\langle \max(t_{ij}|i = 1, 2, \ldots, m)\rangle | j \in J_+\}\} \equiv \{t_{bj}|j = 1, 2, \ldots, n\}
\]

Identify the worst artificial alternative (reverse extreme performance on each criterion) \(A_w\).

\[
    A_w = \{\langle \max(t_{ij}|i = 1, 2, \ldots, m)\rangle | j \in J_-\}, \{\langle \min(t_{ij}|i = 1, 2, \ldots, m)\rangle | j \in J_+\}\} \equiv \{t_{wj}|j = 1, 2, \ldots, n\}
\]

where, \(J_+ = \{j = 1, 2, \ldots, n| j\) is the set of criteria having positive impact (more is better) and \(J_- = \{j = 1, 2, \ldots, n| j\) is the set of criteria having a negative impact (less is better). For instance cost could be one negative criterion since the more expensive is the less desirable.

5. Apply the Euclidean norm distance over each criterion of the weighted normalized decision matrix $T$ to both best and worst alternative with the below formulas:

$$d_{ib} = \sqrt{\sum_{j=1}^{n} (t_{ij} - t_{bj})^2}, i = 1, 2, ..., m$$  \hspace{1cm} (9)

$$d_{iw} = \sqrt{\sum_{j=1}^{n} (t_{ij} - t_{wj})^2}, i = 1, 2, ..., m$$  \hspace{1cm} (10)

6. For each alternative, determine a ratio equal to the distance to the worst alternative divided by the sum of the distance to the worst and the distance to the best alternative,

$$s_{iw} = \frac{d_{iw}}{d_{iw} + d_{ib}}, 0 \leq s_{iw} \leq 1, i = 1, 2, ..., m$$  \hspace{1cm} (11)

**Output:** Rated ranking of all alternatives according to the ratio in Step 6. The best alternative is the one with the highest score.

Thus, TOPSIS MCDA method ranks the under consideration alternatives by minimizing the distance to the best alternative while maximizing the distance to the worst one. Meaning that, the best alternative favours the desirable criteria and disfavours the undesirable criteria. The opposite is true for the worst alternative.

### 2.3 Review of Literature

In this section, we will review similar work to our research, or some of the fields that we used, such as Twitter and social media impact in general, machine learning and sentiment analysis.

Twitter has become a great platform as seen from a marketing perspective. Millions of products compete for a slice of attention making it nearly impossible for a company to broadcast effectively. Instead of selling a message to a group of consumers, on Twitter marketers rely on their customers to talk about their product and help them reach others through electronic word of mouth (eWOM). In marketing, eWOM involves the passing of information concerning a brand, a product, or a service through electronic means. In other words eWOM refers to any statement consumers share via the Internet (web sites, social networks, instant messages, news feeds) about a product, service, brand, or company. Currently, there is some research going on in the area of social marketing. However, most of the research is heavily focused on sentiment analysis and the impact of eWOM. Social
media can be considered as a hybrid element of the promotional mix\textsuperscript{10} combining two factors. That is, social media have characteristics of traditional organization’s integrated marketing communications (IMC) strategies, where companies use social media to talk to their customers, and also play a promotion-related role as they enable the form of electronic word-of-mouth where customers can use it to communicate with one another \cite{21}. In \cite{8} Chu and Kim endorse this hybrid character of social media underlining the importance of eWOM branding via social networking sites (SNSs) like Facebook or micro-blogging sites such as Twitter in the promotional mix. Chu and Kim mention that the increase of eWOM behaviour relies on the reliability and trustworthiness of consumers’ personal network. That is, consumers are more likely to accept and share information that come from a “social friend” than marketers or unknown sources. On the other hand, companies encourage the positive eWOM of their own brands by investing considerable resources in setting up their social profile pages and engaging consumers to make friends with or become followers of the brand \cite{18}. This kind of traditional relationship marketing strategies which aims to fulfil SNS users’ satisfaction (\cite{8}) has been also studied by Hsu et al. \cite{7} three years ago.

Hsu et al, explored the consumers’ perspectives on enterprises conducting relationship marketing through micro-blogging. They conducted an online questionnaire survey of users mostly of ages ranging from 20 to 30 from Taiwan who have followed the micro-blog web page of companies (e.g. Twitter and PLURK\textsuperscript{11}). After the online questionnaire survey they applied statistical analysis methods to analyse and verify the data. According to Hsu et al. \cite{7}, consumers’ satisfaction with corporate micro-blogging has positive influences on commitment, trust, a sense of community, and behavioural intentions. Therefore, companies should enhance their social web profile continuously in order to improve consumers’ brand satisfaction. Update information more often or improve marketing quality could be an example. Another important result of Hsu’s team statistical analysis is that enterprises must design their web pages of micro-blogs according to customers’ interests. They should provide personalized services and information which meets customers’ demands. But although these strategies are vital for enhancing consumers’ satisfaction, consumers’ behavioural intention is also influenced whether they feel as part of community or not. Consequently, enterprises shall improve interactions on web pages of their micro-blogs. Hsu et al. proved that relationship marketing through micro-blogging is conducted when there is an atmosphere of one big family. Therefore, sincere interaction and conversation would be able to create the sense of friendship between consumers and enterprises.

\textsuperscript{10}Promotional mix on Wikipedia, http://en.wikipedia.org/wiki/Promotional_mix

\textsuperscript{11}PLURK official web page, http://www.plurk.com/top/
Although the above work includes Twitter users as part of a statistical research, there are not many related papers on the social presence of a brand in Twitter in the recent literature. Jansen et al [18] mention Twitter as a new entrance in eWOM and Chu and Kim in 2011 [8] believe that a future extension of their research on Twitter will enrich the theoretical background about the key factors of eWOM in social media and facilitate companies to develop tight consumer-brand relationships and implement social advertising strategies effectively. 

As companies are increasingly embracing social media sites to build momentum in their brands there is a major interest from commercial business to analyze social profiles of brand companies. Socialbakers\textsuperscript{12} is a company offering monitoring and tracking tools for analysis of social networks that are used for comparing social media statistics and metrics. Among their results they occupy with the ranking of brands profiles from Twitter micro-blog. Their ranking results are based on the number of followers a brand has. Another relevant to Socialbakers web application is Fan Page List\textsuperscript{13} which is a directory of official Facebook fan pages and Twitter accounts. Through Fan Page List users can find their favourite brands, celebrities, athletes, movies, TV shows, news and sports teams. Similarly to Socialbakers ranking strategy, Fan Page List application also counts the number of followers as an indicator to brand popularity. Apart from business analytics also research community has dealt with comparing or ranking brands through social media analysis. Hsu et al. [7] mentioned that the key of spreading information is the number of followers. Therefore they believe that enterprises should spend more efforts in increasing this number. This report investigates these points of view in order to reveal if other features like the existence of a verified profile in a micro-blog, or the authority of the user who speaks about a company could change the ranking result. Inspired by Chu and Kim [8] who mention that SNS users are more likely to follow “social influences” and therefore marketers should try to identify them, we wondered if this feature can improve the social image of a brand or not. 

The above, were mostly theoretical researches on our subject. Following, are more technical approaches, which mostly use Twitter as a base. 

Chamlertwat et al. [5] were the closest approach to our system. The researchers implemented a multi-layer system, to find tweets about predefined features and analysing the sentiment of the users on that brand, on each feature. The first step was to gather information from Twitter and classifying it whether it is opinionated or not. If the tweet was objective, it was discarded. They found that the percentage of tweets about brands that

\textsuperscript{12}Socialbakers official web page, \url{http://www.socialbakers.com/}

\textsuperscript{13}Fan Page List web page, \url{http://fanpagelist.com/}
contains opinions is the commonly mentioned 20%. Then on the Polarity Detection Module they characterize the tweet text if it is positive or not using Support Vector Machines, which are reported to have the best precision on sentiment analysis. They also perform preprocessing on each Twitter post using

1. tokenization (an array of the words contained on each tweet is created),
2. stop-word removal
3. link removal
4. term normalization (like stemming)
5. and slang handling (good to good)

The result evaluation is made by three experts in smart phone industry, and their opinions complied with the system results. They also presented their findings on several figures (Radar chart, Bar chart, Line chart). One interesting fact they produced was that the number of complimented tweets outnumbers the number of complaints in most cases. They also concluded that sentiment analysis on micro-blogs is a very useful tool for the consumer research, especially in the industries that customers spend their time on social media.

The differences to our system though, are several. We follow the same structure on the first parts of our algorithm, where we pre-process the tweets, find if they are objective or not and characterize them based on the opinion they express (positive or negative). However, our features are dynamically excluded, based on the current trending topic of the category the brand belongs. For example, the current trending topics on mobile phones may be the price, the size or the battery of the phone, but 10 years ago they would be ringtones, appearance etc. Thus, the information is changing overtime, as trends on Twitter do and it has to be followed up, if the system should be up to date. In addition, we proceed with using the features that were extracted for each brand, to rank them or compare them and find similarities on the social web profile of each brand. We assign weights on each feature, that provide us with different results and conclusions, on what is more or less important for a brand, to optimize their behavior on micro-blogging platforms.

On the sentiment analysis part, several papers have measured the precision of popular machine learning algorithms using datasets from Twitter. Movassate & Parikh [27] have tried Naive Bayes, MaxEnt variations and managed to achieve a moderate precision of 64%,
due to the variance of weights on each feature. Go, Bhayani and Huang [14] applied machine learning methods using distant supervision and after comparing their results with the Twitattr platform they got near 80% precision, using Naive Bayes, Maximum Entropy and Support Vector Machines. Finally, Pak & Paroubek [25] tried to train a classifier to find if a tweet is an objective truth, apart from positive or negative using Multinomial Naive Bayes.
Given the theory behind this framework and a literature review of all related works leading to this thesis, it is important to report all the external technical frameworks and tools used. Indeed, this project could not be done if some other projects weren’t already done. Twitter service is one on which this thesis is based to the greatest extent. This is the reason why Twitter service is covered thoroughly in this chapter. The Machine learning tools and the selected database framework we used, are other points which are going to be described in this chapter.

3.1 Twitter Micro-blogging Service

Twitter is an online social networking and micro-blogging service, that enables users to send and read “tweets”, which are texts up to 140 characters added by users on their personal page. A network is created for registered users, where they can read tweets by people they follow or post their own tweets on their time-line. Non-registered users can only read tweets. The platform can be accessed through the website interface, SMS or mobile phones. Twitter is really useful as a platform, when it comes to virality, as the small size of tweets assists the information spread around the social network. Therefore, it has become one of the most popular “trend-setting” platforms and as of March 2010, the current trending topics are visible on the Twitter homepage. The service gained worldwide fame, having nearly 1 billion registered users, 36 million unique monthly visitors and 500 million average number of tweets sent per day by September 2013 [30].
CHAPTER 3. TECHNICAL FRAMEWORKS AND TOOLS

Computer scientists commonly use Twitter as a tool \cite{14}\cite{26}\cite{3}, to prove results of their research on social media, due to its simple API and the large chunks of data one can download and use for analysis. Researchers in this area were helped since the launch of the streaming API also known as Hosebird in 2010. Given this functionality, scientists can gather a huge amount of data to process, to implement an a posteriori analysis on events, news and trends of the data they gathered. Scientists usually gather data from a certain period using the streaming API and use their approach, trying to find supporting evidence of the desirable result.

3.1.1 The History of Twitter Service

Twitter began as an idea that Twitter co-founder Jack Dorsey had in 2006. Dorsey had originally imagined Twitter as an SMS-based communications platform. Groups of friends could keep tabs on what each other were doing based on their status updates. Like texting, but not.

During a brainstorming session at the podcasting company Odeo, Jack Dorsey proposed this SMS based platform to Odeo’s co-founder Evan Williams. Evan, and his co-founder Biz Stone by extension, gave Jack the go-ahead to spend more time on the project and develop it further. One of the biggest steps to raise the platform popularity was the South by Southwest Interactive (SXSWi) conference. During the event, Twitter usage increased from 20,000 tweets per day to 60,000. Tweets were being streamed on 2 screens at the conference hallways, and led to highly positive reaction. Twitter’s usage has spikes during prominent events. For example, a record was set during the 2010 FIFA World Cup when fans wrote 2,940 tweets per second in the thirty-second period after Japan scored against Cameroon on June 14. The record was broken again when 3,085 tweets per second were posted after the Los Angeles Lakers’ victory in the 2010 NBA Finals on June 17, and then again at the close of Japan’s victory over Denmark in the World Cup when users published 3,283 tweets per second. The record was set again during the 2011 FIFA Women’s World Cup Final between Japan and the United States, when 7,196 tweets per second were published. When American singer Michael Jackson died on June 25, 2009, Twitter servers crashed after users were updating their status to include the words “Michael Jackson” at a rate of 100,000 tweets per hour. The current record as of January 1, 2013, was set by all citizens of the Japan Standard Time Zone as the new year began, reaching a record of 33,388 tweets per second (and hence beating the previous record of 25,088, also set by Japan after a television screening of the movie “Castle In The Sky”).
Twitter acquired application developer Atebits on April 11, 2010. Atebits had developed the Apple Design Award-winning Twitter client Tweetie for the Mac and iPhone. The application, now called “Twitter” and distributed free of charge, is the official Twitter client for the iPhone, iPad and Mac.

From September through October 2010, the company began rolling out “New Twitter”, an entirely revamped edition of twitter.com. Changes included the ability to see pictures and videos without leaving Twitter itself by clicking on individual tweets which contain links to images and clips from a variety of supported websites including YouTube and Flickr, and a complete overhaul of the interface, which shifted links such as “@mentions” and “Retweets” above the Twitter stream, while “Messages” and “Log Out” became accessible via a black bar at the very top of twitter.com. As of November 1, 2010, the company confirmed that the “New Twitter experience” had been rolled out to all users.

On April 5, 2011, Twitter tested a new homepage and phased out the “Old Twitter”. However, a glitch came about after the page was launched, so the previous “retro” homepage was still in use until the issues were resolved; the new homepage was reintroduced on April 20. On December 8, 2011, Twitter overhauled its website once more to feature the “Fly” design, which the service says is easier for new users to follow and promotes advertising. In addition to the Home tab, the Connect and Discover tabs were introduced along with a redesigned profile and timeline of Tweets. The site’s layout has been compared to that of Facebook. On February 21, 2012, it was announced that Twitter and Yandex agreed to a partnership. Yandex, a Russian search engine, finds value within the partnership due to Twitter’s real time news feeds. Twitter director of business development explained that it is important to have Twitter content where Twitter users go.

On March 21, 2012, Twitter celebrated its sixth birthday while also announcing that it has 140 million users and sees 340 million tweets per day. The number of users is up 40% from their September 2011 number, which was said to have been at 100 million at the time. In April 2012, Twitter announced that it was opening an office in Detroit, with the aim of working with automotive brands and advertising agencies. Twitter also expanded its office in Dublin.

On June 5, 2012, a modified logo was unveiled through the company blog, removing the text to showcase the slightly redesigned bird as the sole symbol of Twitter.

On October 5, 2012, Twitter acquired a video clip company called Vine that launched in January 2013. Twitter released Vine as a standalone application that allows users to create and share six-second looping video clips on January 24, 2013. Vine videos shared
on Twitter are visible directly in users’ Twitter feeds. Due to an influx of inappropriate content, it is now rated 17+ in Apple’s app store.

On December 18, 2012, Twitter announced it had surpassed 200 million monthly active users. Twitter hit 100 million monthly active users in September 2011.

On April 18, 2013, Twitter launched a music application called Twitter Music for the iPhone.


As of September 2013, the company’s data showed that 200 million users send over 400 million tweets daily, with nearly 60% of tweets sent from mobile devices.

### 3.1.2 Twitter Components

Important features of Twitter are the following:

- **User-based**

  - **Followers** Followers of a user are people who are subscribed to the user’s updates.

  - **Friends/Following** Friends/Following of a user are people that the user is subscribed to.

  - **Verified users** Verification refers to the accounts that are created by the celebrities themselves, or by official representatives of the account. This feature has been enabled in 2008 and it’s intended to help users follow the official page of the user.

  - **Listed users** A list is a curated group of Twitter users. Twitter users can create their own lists or subscribe to lists created by others. Viewing a list time-line will show you a stream of Tweets from only the users on that list.

- **Content-based**

  - **Tweets** A tweet is a text message up to 140 characters. Tweets are publicly visible by default, while this option can be edited to be visible only to one’s followers.
CHAPTER 3. TECHNICAL FRAMEWORKS AND TOOLS

– **Retweets** A user may retweet another user’s tweets on his own time-line. This means that the tweet of the original author is shared on the user’s time-line, and can be viewed by his followers.

– **Favorites** A user can add a tweet on his favorited tweets, and can view all his favorites altogether. Under each tweet, the number of times this tweet has been favorited is shown.

– **Mentions** When a user refers to another user, using the @ tag before his name is known as mention.

– **Hashtags** Words or phrases prefixed with a “#” tag. Hashtags are commonly used by users to describe the content of their tweet.

– **Trending topics** A word, phrase or topic that is tagged at a greater rate than other tags is said to be a trending topic. Trending topics become popular either through a concerted effort by users, or because of an event that prompts people to talk about one specific topic.

### 3.1.3 Data Collection via Twitter API

Twitter provides developers with a vast variety of requests, where one can retrieve almost all the information that users provide publicly. The Twitter API consist of three parts: two REST APIs and a Streaming API. The two distinct APIs (REST API and Search API) are different, as the Search API was originally an independent company (Summize, Inc), that provided Twitter search capabilities for Twitter data. The integration of these two APIs never occurred, thus leading to three different APIs. The Streaming API is distinct from the REST APIs, as it allows a long-lived connection to provide near-live data. The usage of the API is free of charge, it only requires an active Twitter account and is limited to 20,000 requests per hour.

- **REST APIs**
  The REST APIs allow developers to access core Twitter data. This includes update timelines, status data, and user information. The Search API methods give developers methods to interact with Twitter Search and trends data. The concern for developers given this separation is the effects on rate limiting (about 1 request/minute) and output format. Furthermore, the Search API, may not be exact at most times, which means that results returned are the most relevant to the query, but not all data that refer to the search query at the current time. The current version of the REST API is 1.1.
• **Streaming API**
  On the contrary, the Streaming API provides near real-time high-volume access to Tweets in sampled and filter form. The Streaming API allows clients to receive Tweets in near real-time. Various resources allow filtered, sampled or full access to some or all Tweets. Every Twitter account has access to the Streaming API and any developer can build applications today. Hosebird also powers the recently announced User Streams feature that streams all events related to a given user to drive desktop Twitter clients. A recent add to the streaming API is also the site stream, which is the multi-user version of user streams. Site streams are intended for servers which must connect to Twitter on behalf of many users. Twitter can also serve specific requests on their API, using the search API, which allows queries against the indices of recent or popular Tweets and behaves similarly to, but not exactly like the Search feature available in Twitter mobile or web clients.

• **Twitter4J library**
  Twitter4J [2] is an unofficial Java library for the Twitter API. Twitter4J provides a series of methods for handling procedures such as post or search for Tweets, get user’s home timeline which returns a list with the latest tweets, call the time consuming Twitter APIs asynchronously and others. The current version of Twitter4J is 3.0.5 and is 100% compatible with Twitter API v.1.1.

### 3.2 Machine Learning Tools

• **The WEKA Environment for Knowledge Analysis**
  WEKA is a suite of machine learning algorithms written in Java and developed at the university of Waikato, New Zealand.

  WEKA is free software available under the GNU. This helped a lot the scientific community, as programmers all over the world submitted algorithms on the system and enriched its database. WEKA now consists of many data analysis algorithms and visualisation techniques, and it can also be used for predictive modelling. Its mainly used for educational purposes and research.

  WEKA’s mostly used component is the Explorer, where one can import a dataset and review some of the data techniques that can be used.

  – On the preprocessing tab, the user can alter the input attributes, edit, remove them or apply preprocessing algorithms to improve the attributes. On the right
side of the preprocessing tab there are some data about the attributes, as the frequency of appearance, average values etc.

– On the Classify/Clustering tabs, the user can apply supervised/unsupervised learning algorithms on the data, by selecting the algorithm and the parameters that he wants to change, and view the results of the algorithm.

– The associate tab contains algorithms for data mining and especially association rules as the Apriori approach

– The select attributes tab allows the user to apply some transformation on the input data, to alter them the way he wants, as in the preprocessing tab. Here, though the user can test the results and select the one he likes most.

– The visualise tab allows the user to explore the data by visual representations.

One can also run WEKA using the command-line. The WEKA interface is actually a graphical interface that runs commands, by placing the parameters that the user selects and returning the results into the tabs. It can also be run via java code, as an included library. The methods one can use are still the same and there’s no difference with the actual interface.

**The SentiWordNet database**

SentiWordNet\(^1\) is a database of words, each one assigned a positive, negative, and a neutral score, depending on their sentiment. SentiWordNet is based on WordNet and is distributed under the Attribution-ShareAlike 3.0 Unported (CC BY-SA 3.0) license. It has been used in many research reviews, many of them also implementing sentiment analysis on Twitter as in [5].

**The CoreNLP tool**

Stanford CoreNLP\(^2\) provides a set of natural language analysis tools which can take raw English language text input and give the base forms of words, their parts of speech, whether they are names of companies, people, etc., normalize dates, times, and numeric quantities, and mark up the structure of sentences in terms of phrases and word dependencies, and indicate which noun phrases refer to the same entities. Stanford CoreNLP is an integrated framework, which make it very easy to apply a bunch of language analysis tools to a piece of text. Starting from plain text, you can run all the tools on it with just two lines of code. Its analyses provide the foundational building blocks for higher-level and domain-specific text understanding applications.

---

\(^1\)SentiWordNet on page [http://sentiwordnet.isti.cnr.it/](http://sentiwordnet.isti.cnr.it/)

3.3 The Mongo DataBase Framework

MongoDB is an open-source NoSQL-Database. NoSQL is a new trend in database development and is characterized as non-relational system. The traditional relational database management systems created by Microsoft, Oracle, and IBM are great for managing complex data designs that require consistency among data. However, these systems begin to show great weakness when asked to work with more than a few million rows. NoSQL databases are faster and more scalable than relational databases. The interested reader may want to read more about NoSQL databases in the recommended literature.

MongoDB was developed by the software company 10gen (now MongoDB Inc). The company began developing it in 2007 until they released MongoDB as open source product in 2009. MongoDB is a document-based database.

- A **database** is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.

- A **collection** is a grouping of MongoDB documents, instead of tables in relational databases. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection have a similar or related purpose.

- A **document** is a record in a MongoDB collection, instead of rows/records in relational databases. It is the basic unit of data in MongoDB. Documents are analogous to JSON objects but exist in the database in a more type-rich format known as BSON. BSON is a binary-encoded serialization of JSON-like documents. Like JSON, BSON supports the embedding of documents and arrays within other documents and arrays. The documents are represented as a BSON structure with three type of valid values, a primitive value, an array of documents or again a list of key-value-pairs (document).

Figure 3 shows the model described while figure 4 presents the embedded fields of an example document. Note that the entire structure of the document is defined here. This is what allows MongoDB to be “schema-free”.

This embedded data model reduces I/O activity on database system something that increases its performance. MongoDB actually supports a wide array of query operators

---


and constructs, allowing one to replicate a large number of traditional SQL queries as MongoDB queries. The queries on the collections are also BSON structured, that is are expressed as a list of key-value pairs. In addition, the database supports ascending, descending, unique and geospatial indexes. Indexes support faster queries and can include keys from embedded documents and arrays. MongoDB also provides automatic sharding\footnote{Introduction to MongoDB sharding, \url{http://docs.mongodb.org/manual/core/sharding-introduction/}} based on which data can be distributed across a cluster of machines. In such an environment, the clients connect to a special master node which analyses the query and redirects it to the appropriate node or nodes. Every logical node can consist of multiple physical servers which act as a replica set so as to avoid data losses. The replica sets eventually provide consistent reads with low-latency. This cluster structure can be used in conjunction to\footnote{MapReduce on Wikipedia, \url{http://en.wikipedia.org/wiki/MapReduce}} for a really good performance and scalability.

However, MongoDB has some limitations\footnote{MongoDB limitations, \url{http://docs.mongodb.org/manual/reference/limits/}}. One drawback of MongoDB is the fact that
it uses much more storage space for the same data opposed to relational databases. Here, every document can have different keys and the whole document has to be stored, not only the values. That’s why it is recommended to use short key names. In addition, the maximum BSON document size is 16 megabytes [1].

For more information on MongoDB, there is a wide range of publications and tutorials in the literature⁸.

### 3.4 Multi-threaded Programming Principles

Multi-threading is the ability of a software to perform multiple tasks simultaneously, by splitting the processes to more than one, leaving the handling to the CPUs or the operating system. The CPUs have hardware support to efficiently execute multiple threads. While multiprocessing systems have the intention of splitting their tasks to many cores, the main aim of such systems is to increase utilization of a single core by using thread-level as well as instruction-level parallelism.

The multi-threading paradigm has become more popular in the late 1990s, as the multi-core processors were developed at the time by Intel, AMD and others. Multi-core processors can have 2 cores (dual core), 4 cores (quad core), 6 cores, 8 cores, 10 cores or more. The improvement in performance gained by the use of a multi-core processor depends very much on the software architecture, rather than the hardware structure of the processor. In particular, possible speed improvements of the program can rely on how many of its parts can be run in parallel; this effect is described by Amdahl’s law⁹. In the best case, problems may experience speedup factors near the number of cores. Most application, however, are not accelerated that much, given the complexity of large software components, and the effort needed by the programmer to perform multi-tasking that efficiently.

Some advantages include:

- If a thread gets a lot of cache misses, the other thread(s) can continue, taking advantage of the unused computing resources, which thus can lead to faster overall execution, as these resources would have been idle if only a single thread was executed.

---


• If a thread cannot use all the computing resources of the CPU (because instructions depend on each other’s result), running another thread can avoid leaving these idle. If several threads work on the same set of data, they can actually share their cache, leading to better cache usage or synchronization on its values.

Some criticisms of multi threading include:

• Multiple threads can interfere with each other when sharing hardware resources such as caches or translation lookaside buffers (TLBs).

• Execution times of a single thread are not improved but can be degraded, even when only one thread is executing. This is due to slower frequencies and/or additional pipeline stages that are necessary to accommodate thread-switching hardware.

• Hardware support for multithreading is more visible to software, thus requiring more changes to both application programs and operating systems than multiprocessing.

• Multi-threading is known to have a non-deterministic behavior, thus leading to unpredictable causes of system failures, that is the software may run different each time, as the priority of each thread may change due to the operating system’s current priorities. Therefore, the program flow is not always the same and may lead to unexpected results.

3.5 The Klout Service as a Social Influence Evaluator

• Overview
Klout is a web service that measures the social influence of a user according to metrics of seven social media. Klout uses data from Twitter, Facebook, Google+, LinkedIn, Foursquare, Wikipedia, and Instagram in order to assign a unique “Klout Score” to its users, who are then ranked based on this score. Klout scores range from 1 to 100, with higher scores corresponding to a higher ranking of the breadth and strength of one’s online social influence. The Klout Score is not the average of user’s influence across all his networks, it is the accumulation. That is, the more networks someone is registered in, the more he increases the ability to share his expertise, and that helps his Klout Score.

In determining the user score, social influence is measured with three nominally specific measures, which Klout calls “true reach”, “amplification” and “network impact”. True reach is based on the size of a user’s engaged audience who actively
engage in the user’s messages. Amplification score relates to the likelihood that
one’s messages will generate actions. Network impact reflects the computed influ-
ence value of a person’s engaged audience. These measure are based on data from
Twitter such as following count, follower count, retweets, list memberships, how
many spam/dead accounts are following you, how influential the people who retweet
you are and unique mentions. These data are enriched by data from the other social
networks’ followings and interactions.

Klout launched in 2008 and three years after it integrates with Facebook, Foursquare
and LinkedIn, while users are able to connect their Instagram account for a year now.
Other accounts such as YouTube, Flickr, Blogger, Tumblr, Last.fm, WordPress, and
Bing can also be linked by users, however they do not weigh into the Klout Score
as of March 2013. In September 2012 Klout begins its cooperation with Microsoft
whereby Bing would have access to Klout influence technology, and Klout would
have access to Bing search data for its scoring algorithm.

Klout today calculates scores for about 500M profiles and manages a huge volume
of data, 200TB of data and 15B of daily social signals. Although having won the
confidence of the user and companies, Klout receives severe criticism. Several ob-
jections have been raised regarding both the process by which scores are generated.
While it is claimed that advanced machine learning techniques are used, leveraging
network theory, Sean Golliher analysed Klout scores of Twitter users and found that
the simple logarithm of the number of followers was sufficient to explain 95% of the
variance [29].

• Klout API

The Klout API allows developers to get the Klout Score of specific users. To be
noted that Klout receives about 50B API calls monthly. All calls to the Klout API
are defined as a set of simple HTTP request messages and require a unique kloutId.
This kloutId can at first be translated by the screenName or numeric Id that the user
has in the social network. Currently, there are three networks available, Twitter, Klout
and Google+. The response of the request with the kloutId is the score object. For
example, below is presented the procedure of getting Steve Martin’s Klout Score:

Translate a {network}/{networkId} into a kloutId.

http://api.klout.com/v2/identity.json/twitter?screenName=SteveMartinToGo

The response.

\[10\] Klout API Documentation, http://klout.com/s/developers/docs
Utilization of the returned kloutId (id) to look up user score object.

http://api.klout.com/v2/user.json/45598950992084523/score

The response.

{  
  "score": 79.97533416748047
}

Steve Martin is an influential guy according to his really high Klout Score.
Having presented all the theoretical and technical background needed now we are ready to present the core of this framework. As mentioned, having been prompted by the fact that companies are increasingly embracing social media sites to build momentum in their brands, we built a social based brand ranking framework which evaluates the social presence of brands on Twitter, in a comparative manner. Twitter Streaming and Search API are the source of tracking famous brands on Twitter.

4.1 Defining a brand page

The idea behind this project is definitely the brands. A brand page is the profile of a brand or company or product on a social networking website such as Facebook and Twitter. The brand page is not the same as a simple user profile neither a group in that it is created to represent the brands of organizations for promotional reasons. Companies usually hire social media experts to design and administer their brand pages on social networks. One aim of the company behind a brand page is to direct posts to the subscribing users, gain the interest of them and therefore promote a brand. As social media are constantly gaining people’s interest, a brand page is one of the most direct ways of advertising. In addition, a brand page can be used as a metric of trust or interest in the associated brand.

Therefore, a more formal definition of brand page that fits the framework of this master thesis is the following. In addition, figures 5 and 6 show the representation of a brand in the database model.
Definition 3. Let a brand page (or brand) \( b \) be a set of attributes featuring a brand on Twitter [32]. Some of these attributes are related to the profile a brand has on Twitter enriched by other characteristics. Apart from the profile, a brand is related to content sharing through tweets and to monitoring of specific metrics, called features.

4.2 Defining Features

After defining a brand, the next step is to define all the features of a brand. We define as features the monitoring of specific metrics which provide vital information about the social impact of brands on Twitter. For each brand \( b \) there have been collected data related to these features. On the one hand, the social expressions of the subscribed users to a brand and on the other the social presence of the brand itself. In actual fact, there is one feature that is a result of an external framework, therefore the features are metrics related to the Twitter service. We then can categorize all features into three categories. The subjective features, the objective features and the metrics. The subjective features are sentiment based features related to the opinion users have for given brands. The objective features are deterministic indicators. That is, real numbers measured for each brand. Last, metrics are features from our analysis or external analysis services, which are currently represented only by Klout.

4.3 Designing the Database Model

The model of the database is designed based on the document based MongoDB database which uses a “nested” oriented structure and can contain multi-value fields. MongoDB was firstly selected for its scalability and its efficiency in speed compared to the traditional relational databases. In addition, traditional databases show great weakness when asked to work with more than a few million rows. The model consists of two databases with the collections Brands and Features, respectively.

4.3.1 Brand Collection

This collection gathers all the characterizations of the brands. Indeed, it consists of a number of documents, each and every one of them refers to a brand. Figures 5 and 6 show the representation of a document in Brand Collection. A brand (or document of the collection Brands) is composed by many attributes as defined in section 4.1. Therefore, a brand consists of the following:
• **Id** is the unique serial number of a record.

• **Name** is the name of the brand $b$ as defined on Twitter by the user.

• **Category** is the set the brand $b$ belongs to. In this case study there are four categories. Indeed, a brand $b$ could be part of one of the set of brands under the category Fashion, Food/Beverages, Technology or Auto.

• **User id** (Given by Twitter) is the integer representation of the unique identifier for this User. A brand is consider a user on Twitter.

• **Klout id** (Given by Klout) is the id of the brand on the external Klout service. Based on this id Klout service provide the “Klout Score” for the brand.

• **Image url** is the HTTP(s)-based URL pointing to the background image the user has uploaded for his brand profile.

• **Profile** is an array of profiles. That is an array of screen names. A screen name is the user’s identification name on Twitter. Screen names are unique but subject to change. A brand may have more that one profiles under different screen names. The array holds the most of these profiles. It does not contains non English profiles.

• **Url** is a URL provided by the user showing the official web page of the brand. Url consists of the original expanded url and its shortened representation.

• **Tweets** is an array of tweets. These tweets refer to the posts other users have done about a specific brand. A tweet has its own content characteristics.

As it can be seen on figure 5 each tweet has its own content. Figure 6 shows the components a tweet, for a brand consists of.

Therefore, a tweet content consists of the following:

• **Id** is the unique serial number of a tweet.

• **Text** is the actual UTF-8 text of the tweet.

• **Created at** is UTC time when this tweet was created.

• **Hash-tags entity** is an array of all the hash-tags which have been parsed out of the text of the tweet.
CHAPTER 4. BUILDING A BRAND RANKING FRAMEWORK

Figure 5: Representation of a document in Brand Collection.
Figure 6: Representation of the content of a tweet in a document in Brand Collection.
• **User mentions entity** is an array of all user mentions parsed out of the text of the tweet.

• **Url entity** is an array of all the urls parsed out of the text of the tweet.

• **Retweet count** is the number of times this tweet has been retweeted by Twitter users.

• **Favorite count** is the number of how many times this tweet has been “favorited” by Twitter users.

• **Geolocation** is the geographic location of this tweet as reported by the user. That is the longitude and latitude of the tweet’s location.

• **Sentiment** is the result of the Sentiment Analysis of this tweet. That is an indicator of whether is positive, negative or neutral the sentiment the expressed by this tweet.

• **User** is the user who posted this tweet. A user has embedded attributes.
  
  – **User id** is the id of the user who posted this tweet.
  
  – **Followers count** is the number of followers of this user.
  
  – **Friends count** is the number of friends of this user.
  
  – **Location** is the location defined by the user, not necessarily an actual location.

The JSON representation of an example brand is presented in Appendix A.1.

### 4.3.2 Features Collection

This collection gathers all the features measured for the brands. Indeed, it consists of a number of documents, each and every one of them refering to a brand. In fact a brand here is the exact same document only differing in some attributes from a brand in Brands database. Substantially, Brand collection is the one which collects all the necessary data used for calculating the features stored in Features database. The common attributes are the name, the category, the user id, the kloud id, the image url, the profile an the url.
Figure 7: The architecture of the social based brand ranking framework.
4.4 The Architecture Behind the Framework

The architecture behind the social based brand ranking framework is presented in Figure 7. The diagram visualizes how the application works internally. There are two architecture layers, the presentation level and the application level which are implemented to run in two different web servers.

- **Application level**
  The application level is responsible for the collection, manipulation and storage of the data. Here, a Java intermediary, on the web server 2, communicates with the Twitter Streaming and Search API using the HTTP protocol in order to track the needed information about the desired brands. These data, after being manipulated, are stored in Mongo database (See section 4.3). The Java intermediary executes another request on an external application Klout. The result of this request is also part of the feature extraction and will be explained in section 5.2.

- **Presentation level**
  The user in the client side interacts with the web application that run in the web browser of the user. The client side (or front-end) is an interface between the user and the server-side (or “back-end”\(^1\)). The user’s interactions can be translated to requests to the web server 1. The client side communicates with the web server 1 using the HTTP protocol. The web server 1 forwards each request to the PHP interpreter, which in turn processes the request. The interpreter handles the communication the Mongo database where the features are stored. Once the request is processed, the results are sent back to the client and are then visualized for the user.

The practical part of this master thesis is the implementation of the social based brand ranking framework. This platform is used to measure features which will show the social presence of brands on Twitter. The rated brands after classification according to their score, are visualised on a web application. As shown on figure 8 the implementation can be split into four main steps. On the one hand, we have the preliminary procedure, where we collected all the necessary keywords for our searching case study and on the other hand, we prepared the training set for WEKA. The next step is related to the searching procedure based on the keywords collected on the previous step and therefore the storage of the return data. Followed by the the calculation of the feature, the visualisation step shows the ranked brands on a web application.

5.1 Preliminary Procedure

5.1.1 Collecting the Preliminary Required Data

The implementation of the social based brand ranking framework requires sort of preliminary work. Indeed, in order to collect data and therefore rank the social impact of brands,
we need a list of them. Following a semi-structured procedure, we filled the list of brands from many catalogues. The desired brands had to satisfy two certain conditions, have a public Twitter profile and belong to one of the predefined categories. We decided to have four categories of brands in order to perform various experiments. The four categories a brand should belong to are the following:

- **Fashion** category consists of brands related to clothing industry, footwear, accessories, make-up or home design.

- **Auto** category is a list of brands regarding different means of transport such as cars or motorcycles.

- **Technology** under this category there are brands of companies or applications related industries such as electronics, software services, hardware, computer services and Internet.

- **Food & Beverages** category is more a product related category since refers to brands in food and drink industries.

The list of brands came from a list of catalogues and commercial web sites found on the Internet. We mostly looked at Twitter and databases such as Wikipedia and Open ICEcat. A simple parser was built based on jsoup Java HTML Parser. Jsooup is a Java library which provides functions for extracting data using DOM traversal or CSS selectors. The parser looks through all DOM elements and returns the desired values of the selected elements. In this step we look for brand names, their profile names on Twitter and if possible their profile images. Subsequently, these data are stored to the Brand collection in MongoDB. Below the Java code of the parser and an illustrative example is presented.

```java
public static void parser(String categoryName) {
    try {
        doc = Jsoup.connect("http://somesite.com").get();
        list = doc.select("div");
        for (int i = 0; i < list.size(); i++) {
            // selects a brand name
            brand = list.get(i).select("p.title");
            // selects a url with Twitter profile
            profile = list.get(i).select("a");
            // selects an image
            photo = list.get(i).select("img");
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

if (profile == null || profile.text().toString().trim().equals("")) {
    br = brand.first().text();
    prof = "";
    pho = photo.first().absUrl("src");
} else {
    br = brand.first().text();
    pho = photo.first().absUrl("src");
    String followUrl = profile.first().attr("href");
    prof = "@" + followUrl.substring(followUrl.lastIndexOf("=") + 1);
    //stores in MongoDB the brand
    BasicDBObject in = new BasicDBObject("name", br).
        append("imgurl", pho).
        append("profile", prof).
        append("category", categoryName);
    db.insert(in, Mainb);
}

Listing 5.1: Example of jsoup Java HTML Parser.

The data collected had to be processed by hand. For instance a search of "nike" on Twitter will return more than ten profiles. The same occurred with our parser. It returned the same brand with many profile names. Since we are interested in analysing the brand’s social presence we made a compromise. The more profile names a brand has the more social we consider it. Therefore, our database has more than one profile name under each brand. In the Nike example, there are twenty one profile names.
To be noted that collecting the list of all the brands was a quite time-consuming procedure and has a main drawback. Inserting a new brand in the database is not dynamic, meaning that requires human intervention. This limitation left for future work since what was important here was the analysis of the data in some case studies and not building a real application for commercial usage.

5.1.2 Preparing WEKA Training Set

To create the training set in WEKA a database of tweets, that we gathered using the streaming API, was used. The text input of each tweet was given as input to the class that was created. We then assigned each instance to one of the following values: positive/negative/neutral. To optimise the procedure a GUI was created, with the text of the tweet as input and a selection of the values mentioned above. A training set of approx. 1000 tweets was created. The tweets that were used, were the most representative ones from the set, for example if a tweet contained both negative and positive feelings we didn’t assign a sentiment, to avoid noisy data on the classifier’s results.

The steps of the procedure are the following:

1. Creation of a text file, each line containing the text of the tweet to be characterised.

2. First pass of the text, by removing urls, references, punctuation, hash-tags(only the symbols, not the words), and all other non-alphanumeric characters. Question and exclamation marks, where converted to words, for the algorithm to be able to understand when these symbols exists in a sentence.

3. Insertion of the text file as input to the GUI, and assignment of sentiment to every tweet, that was representative enough, as can be seen on Figure 9 and 10

After the initial creation of the training set, the weka structure of a training set was used as in here.
@relation sentiment

@attribute tweet string
@attribute sentiwordnetpositive numeric
@attribute sentiwordnetnegative numeric
@attribute sentiwordnetneutral numeric
@attribute sentiwordnettotal numeric
@attribute sentiment {positive, negative, neutral}

@data

"i saw a jeep today and fell in love"
↓, 0.2727272727272727, 0.0, 0.2727272727272727, 0.7068120459893943, positive

"i bet the moment danny from the script appeared on the voice must have crashed"
↓, 0.1875, 0.0, 0.125, 0.06590406631439162, neutral

"get quality get performance save money experience ultrabook available now"
↓, 0.5384615384615384, 0.0, 0.15384615384615385, 0.7504481414758203, positive

"everyone needs in their lives now go go go"
↓, 0.2857142857142857, 0.0, 0.0, 0.023067840056166525, neutral

"why is todd hoeing sounds amazing"
↓, 0.125, 0.0, 0.125, 0.27272727272727276, positive

"the new maschine looks incredibly tasty"
↓, 0.3333333333333333, 0.1666666666666666, 0.0, 0.2783020458778565, positive

"payleven adds customer loyalty schemes to its mobile payments"
↓, 0.2222222222222222, 0.0, 0.0, 0.1111111111111111, 0.09947546415228026, neutral

"i want one they got em in black too one says love i thought about u"
↓, 0.1764705882352941, 0.11764705882352941, -0.02852683637009878, positive

"going hard cause i need to not cause i want too"
↓, 0.09909090909090909, 0.5454545454545454, 0.0, -0.7814933203348622, neutral

"rhinestone encrusted sandals from design shoelove zappos"
↓, 0.25, 0.0, 0.0, 0.13780314320552772, neutral

Listing 5.3: Example of WEKA structure of a training set.

Then, the StringToWord filter by weka was used, to implement standard preprocessing techniques on the training set, before using it for the classifier. The following methods were used:

- Stop words removal
- TF/IDF transform
- Document(tweet) length was normalised
- Word count was used, if a word was found more than once within a tweet
- Porter stemming
- Simple word tokenizer, where each word is a different attribute. No N-grams were used.


5.2 Collecting and Storing the Data

This is the main Java project that was used, in order to collect the tweets and insert them into the database. It consists of different classes, some of which will be explained further on this section.

- **Main class**
  This is the main class, where the main function is used. The collection of the tweets was made using Twitter’s Streaming API, which was explained in previous sections. To use Twitter’s streaming API in Java Twitter4J library was used (see section 3.1.3), to interpret Twitter API methods to java code.

  The Streaming API allows us to follow up to 400 keywords and 400 profiles. We preferred the input parameters to be the names of the companies. That, of course, raised some issues, as e.g. Apple can both refer to the company and the fruit. Therefore, the tweets that are gathered, and refer to that brand may also be tweets about someone uploading something about the fruits. This was a problem we had to face, along with the limit of up to 400 profiles to follow. On the future work section, these problems are described more thoroughly.

  To optimise the procedure, and maximise server up-time, a multi-threaded approach was used, to process the streaming API the best way possible. On our approach, we had to handle every single tweet that came to us, before we stored it in our database. That happened, because by adding the approx. 400 brand names as parameters, there was no way to know which keyword each tweet contained, to add it on the corresponding brand. Thus, a function was called, to split all words of the tweet, and find which word referred to one of our brands, and add it on its document at the database.
A hash set was used, as search queries need $O(1)$ complexity to search within the collection.

The multi-threaded approach that is followed, keeps a main, shared queue that all threads share. The queue supports concurrency, for the threads to insert/remove tweets simultaneously. Each tweet, that comes from the Streaming API is inserted into this queue by the main thread. 4 similar threads from the class “StoringThread” receive tweets from the queue, process them by searching to what brand they refer and store them into the database. Before receiving a tweet from the queue, all threads check the memory that the program is allocating. If the free memory is less than 500MB the queue is emptied, to avoid any heap allocation errors. That way, a small chunk of data is lost, but server up-time is optimised. Along with the database storing, the characterisation of the tweets occurs by the classifier.

- **Frequent Words Feature**

This class was used, in order to get the most popular hash-tags on each category, in terms of appearance. It contains a main class, which is to be used separately from the whole project. The idea is to find the trending twitter topics(hash-tags) and store them into the database every week. Every Sunday, the trending topics of the past week are generated and are stored into the database, for the analysis of the next week. Thus, we see how many tweets for each brand, are relevant to one of the trending topics and are positive. This is a good measure, given that if more users tweet about a brand and a trending topic, it’s more likely for other users to view that tweet. If this tweet is positive, then it’s positive advertisement for a brand, with a high possibility for people to see.

The class gathers all tweets from the database for the last week, grouped by category. It then keeps a HashMap for each category, that has the hash-tag string as a key and the times it appears as a value. Each collection is then inserted into a TreeMap, to be sorted by value. As the output of the algorithm, a list is shown -for each category- that prints the most seen hash-tags. This process was created, for a human eye to see the words and manually select the proper ones. We tried to skip issues, such as removal of stop-words, selection of nouns only using core-NLP, removal of words less than two characters, but the words that were produced weren’t always acceptable, e.g. some words were another trending word’s plural. Thus, we had to insert the trending topics manually, to avoid such issues.
CHAPTER 5. IMPLEMENTATION

• Klout API
As its name states, this class was used to communicate with the Klout API and receive data with HTTP requests. Klout API has been explained on 3.5. In general, it allows 2000 requests/day and is used to assign a score to each brand from an external service. There were 2 procedures that were followed, to insert the Klout score into the database.

1. In the first procedure, we collected all the Klout IDs of our brands, and inserted them into the document of each brand. This was a simple request to the Klout API and was made, as to collect data about a profile from Klout, one has to query using an id.

2. The second procedure, was gathering the scores of each brand, which was used on the Feature Extraction project and will be explained further on 5.3.

• Machine Learning Annotation
The MLAnnotator class was the GUI that was used for the characterisation of the training set, as explained on 5.1.2. It receives a text file as an input, that contains the brand name and the tweet text per line. The algorithm then shows the user a panel, on which the brand name and the text appears and 4 buttons, from which he has to select. The buttons are the algorithm’s result, thus Positive, Negative, Neutral and the skip option, if the user wants to avoid characterising the current tweet, as he may consider it noisy.

• MongoDB connector
MongoDB class, is a manually created framework for the application, which uses the Java mongo driver. A lot of functions were created to make our lifes easier, as insert, remove and update functionalities.

Furthermore, some more complex procedures were created in this class, to improve the readability of the code in other classes. For example, the insertion of tweets within a brand document, was a complex procedure as MongoDB allows a document size to be up to 16MB. This raised an issue, as after a point some brands couldn’t be updated with more tweets. This issue is explained in more details, on 5.5 section, where all limitations are pointed out. The final query that was used had a queue structure, as the oldest tweets were removed, for the collection to fit newer tweets.

Other methods, contained standard procedures such as:

– Insert a document into a collection
– Remove a document from a collection
– Update a document into a collection
– Update a document’s field into a collection
– Find all documents in a collection
– Find a single document in a collection (findOne)
– Find only one field of all documents in a collection
– Find all brand names in a collection
– Find all brand profile names in a collection
– Find all brand profile names and names in a collection
– Find all brand IDs in a collection

• **SentiWordNet with core-NLP connector**
  SentiWordNet 3.0 was used, as explained in previous sections, in order to assign a positive, a negative, a neutral, and a total sentiment score to some text. In this class, the coreNLP code is also contained, to provide the natural language analysis functions, which are needed.

SentiWordNet’s lexicon contains the sentiment score of the words, but the part of speech of the word is needed. Therefore, coreNLP had to be used, before requesting results from SentiWordNet’s data. The initialisation code of both tools was given. All we had to do, is connect both frameworks to an integrated solution, that worked as a separate class. Thus, to assign a score to an input word, first the word was given to the CoreNLP framework, to recognise its part of speech. Then, the result of coreNLP, was given to SentiWordNet with the part of speech letters:

1. **n**: noun
2. **r**: adverb
3. **a**: adjective
4. **n**: noun

The result of SentiWordNet’s score is returned from the function `getWordSentiment`.

• **WEKA**
  WEKA class is responsible for all functions regarding the Machine Learning part of the process. SentiWordNet’s functions are also used here, in order to create an instance as explained on 5.1.2.
As an input to the algorithm, the tweet text is given and the result is one of the following values: positive, negative, neutral. As described in previous sections, the StringToWord filter is used, to preprocess the text and split it into separate words for the algorithm to handle. The classifier that is used is the linear Support Vector Machines (LibLINEAR\(^5\)). The steps that are being followed are:

1. Create the classifier model from the training set
2. A first preprocessing filter occurs, to normalize the input text to simple techniques, such as lowercase, punctuation symbols renaming etc.
3. Text is given as input to the SentiWordNet class, to receive the score of each word. A counter is used for the positive, negative and neutral words that were found, as well as the total SentiWordNet score. The result of the SentiWordNet scores is given as: \( \frac{\text{positivecount}}{\text{number of words}} \), \( \frac{\text{negativecount}}{\text{number of words}} \), \( \frac{\text{neutralcount}}{\text{number of words}} \) and \( \text{totalsentiwordnetscore} \).
4. Attributes are given to the classifier, in the following order: Tweet text, SentiWordNet positive score, SentiWordNet negative score, SentiWordNet neutral score and SentiWordNet total score.

### 5.3 Extracting Features from Stored Data

The feature Extraction project, is a Java project that consist of a few classes to query the database entries and extract the features needed for the analysis. It then stores these feature values into a new database, as mentioned earlier. It also makes a request to the Twitter Search API, to receive up-to-date data about each brand.

All functions use data from this week’s Monday, to the current day of the week. This implementation was made, in order to show weekly reports on the website, which are updated every 6 hours. For example, Nike may have 300 retweets up to Wednesday of the current week, but on Thursday the retweets may reach 500. The website always shows data up to the current day of the week, so if you access the data on Thursday the retweets need to be 500. Therefore, the feature database is updated every 6 hours, to show up-to-date data, until the current day of the week.

(Notice: All tweets that are gathered for each function are tweets for this week, that is from Monday of the current week, until the current day. Only some more generic features, such as followers, friends etc. don’t use the current week restriction.)

At the end of each while loop, the features are sent into the database, and are stored as a new entry in the features sub-collection for each brand.

5.4 Visualizing Data on a Web Application

The system is completed by its fourth step, the visualisation of the ranked brands on a web application. This application is a simple mean of giving the user an idea of the logic behind this system. We try to give our results in a simple format through the web application so as to ease anyone who wants to understand the key features that influence brand social impact. Like any web application, this one has two main parts the front-end and the back-end as will be explained below. The structure is based on the MVC which divides the application into three interconnected components. Model, controllers and views. The central component, the model, consists of application data, business rules, logic, and functions. A view can be any output representation of information, such as a chart or a diagram. The third part, the controller, accepts input and converts it to commands for the model or view. Although originally developed for desktop computing, MVC has been widely adopted as an architecture for World Wide Web applications in all major programming languages. Model and controllers are implemented as part of the back-end, while views are part of the front-end.

5.4.1 Front end

This is the front-end of the web application which runs on the browser of the user. It is essentially the part of the tool the user sees at any time. The front-end visualizes the result of the social based brand ranking framework. It has been designed to provide the essential information to the user and himself to be able to interact with it easily. The core of the application relies on the continues communication between the front and the back-end via asynchronous Ajax requests. Ajax technology allows the web application to send data to, and retrieve data from, the back-end on the web server asynchronously. This means that something happens to the page after it is loaded. Traditionally, when a page is loaded, the content remains the same until the user leaves the page. With AJAX, JavaScript grabs new content from the server and makes changes to the current page. This all happens within the lifetime of the page, no refresh or redirection is needed. All Ajax requests are implemented with the JavaScript library jQuery. The application makes use of the JavaScript library jQuery for most of the actions. Indeed, the selection of DOM elements, animations effects, handling client events, and the Ajax requests, all are based on JQuery. The web application was developed also by using the web technologies HTML, PHP and CSS, while all the

---

graphs were created by Highcharts\(^8\), a third open source charting library written in pure HTML5/JavaScript.

The user by visiting the web application sees a more informative homepage, which urges him to try the application. When the user enters the main page, the front-end sends the Ajax request to the back-end, while displaying a “loading” icon, notifying the user to wait a while for the results. The request, as shown below in listing code below, calls the php controller \texttt{topsis.php} which is responsible for ranking the brands. The functionality of the php controller is explained extensively in section \textit{5.4.2}.

Then, the front-end calls the function \textbf{DisplayBrands(data)} which replaces the loading icon with the actual results. Otherwise where an error may has been occurred in the server side and the result is not the expected one, the front-end informs the user with an error message.

The results of the Ajax call are visualized as a list with first the brand with the highest rate. Each line shows the serial number of the position a brand has, the profile image and the name of the brand. The number at the right is the rate a brand gets after the implementation of the TOPSIS ranking algorithm. This rate is also visualized as a horizontal bar showing the rate as percentage of one.

In addition, the list is in descending order. If the user wishes to see the results in ascending order, he only has to press the right “Arrow Up” button found above the results. The order change does not require any request to the server. It is implemented with Javascript code.

After the page has been loaded the ranking list refers to all the brands stored in the database, regardless of category. If the user wants to filter the brands based on the category, he can. In other words, the user has the option to select to see brands from one category. In this case the user just clicks one of the menu buttons on top of the page. Each button refers to one of the categories of our case study. Technology, Fashion, Auto, Food/Beverages. Again, the result is implemented in front-end via JavaScript not requiring any request to the server.

It can be observed from the images that there are ten hash-tags. These are the ten most frequently appeared hash-tags in the tweets of each of the four categories. The ten hash-tags are initially stored to an array after the page has been loaded. This is the results of another Ajax request to the php controller \texttt{trendingTopics.php} (see section \textit{5.4.2}). Listing code below is the JavaScript code of this Ajax request and its result.

\footnote{8Highcharts library, \url{http://www.highcharts.com/}}
When needed the ten hash-tags per category are visualized as links above the ranking list of brands. Each one of these links refer to a result page on Twitter with search option the pressed hash-tag. This shows the user, what is going on on Twitter regarding the selected hash-tag. The user has been navigated to this page after clicking hash-tag #apple on Technology category.

As mentioned before and will be further explained in section 5.4.2, TOPSIS algorithm in the php controller is waiting for the weight values for each features. By default the first Ajax request executed after page has been loaded sends the weights all with value 0.1. TOPSIS algorithm requires the weights to be normalized. Therefore, the weights are first normalized as shown in listing below and then sent via the Ajax request.

Last, the user can see some statistics for each brand by clicking to a brand in the list. A window appears containing the following information:

- The name and the profile image of the brand on the header as link to its page on Twitter.

- On the left banner there are all the values of the features from the last record date.

- On the right banner there are two charts. The first one shows the trend of the followers from the first date of records. The seconds charts shows simultaneously the rate and the position a brand gets each day. These statistical values shows a lot for the brand and will be explained more in the Results section.

All the data needed for the display of the statistics are results of an Ajax request executed upon mouse release. Javascript sends to the back-end brandAjax.php controller the name of the brand and the latter retrieves all the records related to that brand. The code below shows this Ajax request. Upon success, JavaScript calls the library Highcharts in order to visualize the results as a line chart for the followers, rates and positions the brand has on each record date.

5.4.2 Back-end

The user’s interactions can be translated to requests to the web server in the Presentation level as explained in section 4.4. The back-end of the web application resides on this web server and based on the MVC model these parts of code are controller which returns a result to the front-end of the web application (viewers) after necessary calculations. The
back-end handles the communication with Mongo database where the features are stored. All the algorithms are implemented using the server-side scripting language, php⁹.

I. Ranking Brands Using TOPSIS MCDA method

As explained in the front end section 5.4.1 the main request from the user regards the evaluation of all the considered brands in a comparative manner. This problem can be seen as a multi-criteria decision problem. To the best of our knowledge this is the first approach in the literature which performs analysis on twitter data from the MCDA theoretical perspective.

As explained in section 2.2, MCDA equips decision makers with various methods so as to assist them to reach better decisions when they deal with problems involving multiple criteria of conflicting nature. The brand ranking problem in this case can be de-constructed in the basic components of a multi-criteria decision problem. Therefore, the 392 brands correspond to 392 different alternatives.

Among the numerous methods provided by the MCDA, which solve this kind of problems, the one more appropriate for the presented multi-criteria decision problem was TOPSIS technique. Recalling that TOPSIS selects the best alternative to be the one that has the shortest geometric distance to the ideal alternative and is furthest from the negative ideal alternative. The ideal alternative is the one which consists of the best level for all attributes and the negative ideal alternative is the one that has the worst attribute values. TOPSIS MCDA method was selected according to our own and previous observations found in the literature [19] [24]. TOPSIS method is based on a strategy which simulates human logic regarding the selection procedure and its computational complexity is rather simple. In addition, the necessary weights are applied by the decision makers, who in this case is the application user (See section 5.4). As far as the type of criteria values is concerned, the brand ranking problem uses features with numeric and monotonically decreasing or increasing values, the type of which is appropriate according to Yoon and Hwang for TOPSIS ranking method [35].

TOPSIS algorithm is implemented in server-side having as inputs m alternatives, n criteria and a weight for each criterion. Interest of simplicity and understanding of the algorithm, from now on alternatives will be referred as brands and the criteria as features. These values are derived from two sources. On the one hand, the weights are sent by the front end. On the other hand MongoDB is responsible for the brands and features.

After connection to the MongoDB database is established, two requests are executed to bring the data of the collections of MongoDB.

The **Features collection** in MongoDB database gathers for each of the 392 brands all the features in chronological order. The request returns the most recent record of the “features” matrix.

The six steps of TOPSIS algorithm, as shown on section 2.2.2, are implemented by three functions.

- **Normalization function**
  The decision matrix \(X(x_{ij})_{m \times n}\) is automatically constructed as these values are stored in the database appropriately. Then, the first step is to normalize the decision matrix \(X(x_{ij})_{m \times n}\). The normalization function is responsible for the construction of the normalized decision matrix \(R(r_{ij})_{m \times n}\). The normalization procedure is necessary for the proper comparisons across the features since they are in various dimensions.

  The first step of the function is to calculate the sum of all the initial squared \(x_{ij}\) features. Therefore, using formula (4), the algorithm assigns to each normalized feature of each brand \(r_{ij}\) the initial feature \(x_{ij}\) divided by the square root of the sum of all initial squared \(x_{ij}\) features. To be noted that the function assigns 0 to a normalized feature if the denominator of the formula is equal to zero. This case is extremely rare since the square root could be zero only if all the features have zero values.

  The normalization function simultaneously calculates the weighted normalized decision matrix \(T(t_{ij})_{m \times n}\). According to formula (5) \(T(t_{ij})_{m \times n}\) is computed by multiplying each column of the normalized decision matrix \(R(r_{ij})_{m \times n}\) by its associated weight. As mentioned before, the weights are sent by the web application. The web application is responsible for the normalization of the weights (see section 5.4). The normalized weights are calculated according to formula (6) base on the original weight given to the each feature by the user.

  The normalization function returns the weighted normalized decision matrix \(T(t_{ij})_{m \times n}\) with each feature multiplied by its associated normalized weight and all the features normalized in one common dimension.

- **FindMinMax function**
  The next step of the TOPSIS algorithm is to identify the best and the worst artificial brand \(A_b\) and \(A_w\). Recalling that the best (worst) alternative would be the one having the best (worst) values on each feature. The findMinMax function iterates through
all columns of the weighted normalized decision matrix $T(t_{ij})_{m \times n}$ and performs two checks. Each feature belongs to one of the $J_+$ and $J_-$ sets depending on the impact that has on the brands. If the feature has “max” value (positive impact) it is a part of the $J_+$ set. Otherwise, if the feature has “min” value (negative impact) it belongs to the $J_-$ set. In this case study, $J_+$ set consists of all features apart from the “negative” feature which has negative impact and belongs to the $J_-$ set. Therefore,

1. If the current column (feature) has positive (negative) impact to the brand, the function adds the maximum (minimum) value of this column to the $A_b$ array.

2. Otherwise, if the current column (feature) has positive (negative) impact to the brand, the function adds the minimum (maximum) value of this column to the $A_b$ array.

In other words, the $A_b$ array consists of the minimum value of the “negative” feature and the maximum values of all the other features. The $A_w$ array consists of the maximum value of the “negative” feature and the minimum values of all the other features. The findMinMax function returns a matrix with the ideal and worst brands, $A_b$ and $A_w$ respectively.

- **Distance function**

According to TOPSIS algorithm the next step is to apply the Euclidean distance over each feature of the weighted normalized decision matrix $T(t_{ij})_{m \times n}$ to both ideal and worst brand $A_b$ and $A_w$, with the formulas (9) and (10). The distance function iterates through all rows (brands) and calculates the $L2-distance$ between the current feature and the corresponding best or worst value. For instance, for the brand Instagram the result would be the number $d_{ib}$ for the Euclidean distance between Instagram and $A_b$ brand and the number $d_{iw}$ for the Euclidean distance between Instagram and $A_w$ brand.

After the Euclidean distances are calculated, the function then calculates the ratio which will be the final score for each brand. This is the last step of TOPSIS algorithm. In accordance to formula (11), the ratio is equal to the fraction of $d_{ib}$, divided by the sum of $d_{ib}$ and $d_{iw}$. In other words, this score represents the distance to the worst $A_w$ brand divided by the sum of the distance to the worst $A_w$ brand and the distance to the ideal $A_b$ brand.

The distance function returns the brands with their corresponding ratio. The final step in this php code is the ranking procedure. All the brands are sorted in descending order based on their score. The best brand is the one with the highest score.
This php controller returns all the brands to the web application. Each brand apart from the feature array explained here, has a lot of other information. Therefore, the controller returns for each brand the following information needed for visualization purposes.

5.5 Framework Limitations

This section summarises the technical limitations encountered during the implementation of the application. Through this process the difficulties or wrong estimates made are recognized. The collective address of these wrong estimates can lead to a better implementation.

- **MongoDB Limitations**
  
  Our experience on MongoDB improved a lot during this thesis, as we haven’t used it a lot before. One of the issues we faced with the database, 1 month after the creation of the website was the MongoDB document size. One of the documents exceeded the maximum document size(16MB), which prevented us from inserting new data within the document. In particular, the document was one of the brands in the tweet collection, which had more than 50,000 tweets and couldn’t fit any more. We managed to overcome this issue, by creating a queue-like structure for the tweets, as we didn’t need older tweets, since the analysis was weekly. The limit of tweets withheld within each document is now 40,000 tweets. If a document exceeds this size, old tweets are replaced with new ones as described in previous sections.

  If we were to rebuild the database model, we would create a separate collection of tweets, which would have a single document for each tweet, and a field for the id of the brand it belongs to. This would solve any document size issues and allow us to build a better relational model.

- **Twitter API limitations**
  
  Twitter’s API comes with several limitations, which prevented us from having the best results possible. At the beginning of the implementation, a lot of changes had to be made, in order to comply with the API limits. One of the major issues we had, was that we gathered more than 1000 brands and started using the REST API. We detected the issue really fast, as to make a single pass for all brands through the REST API, we needed 1000 minutes at minimum(1 request per minute). That of course needed to be changed, so we started using the Streaming API, to avoid such long intervals.
CHAPTER 5. IMPLEMENTATION

The Streaming API, though, may be near real-time and produce a huge amount of data, but still up to 400 brands are allowed per account. Thus, we had to get rid of about 600 brands to use it. The selection of the brands to remove from the database, should not be made randomly, so we had to order all brands in the database by followers, and then keep the first 400.

Still after this procedure, we figured that the Streaming API, allows up to 400 single keywords, which means that one can’t create a query such as "Nike OR @Nike OR @NikeRunning", because it allowed only one of these words to be used. Therefore, we had to limit the query to the brand names only, which raised another issue, that is the disambiguation among brand names. For example, if the Streaming API returns a tweet from the keyword "Apple", it may refer to the brand or the fruit.

To provide a solution, our thoughts to address the issue in the future are to analyse each tweet in terms of grammar and syntax, and decide if the user who tweets, refers to the brand or the other possible meanings of the word. Some thoughts were to do this, using machine learning models, where a classifier could be trained to separate the use of a brand within a sentence, among the use of another meaning of the word in other sentences. Another possible and viable solution is also the purchase of Twitter data, which can be made through various platforms we encountered during our research.

- **Human intervention limitations**

  As explained the first step to our implementation was to define the keywords based on which we could collect data. In our case study these keywords were related to brands on Twitter. Recalling that this requires a list of brands in order to collect data and therefore rank the social impact of these brands. Following a semi-structured procedure, we filled a list of approximately 400 brands. This procedure makes the application not scalable. At first, this decision was taken to our knowledge since we cared more about the research and less for the application itself.

  Therefore, a step forward could be the dynamic insertion of brands to the system. This procedure has several solutions. The most desirable one could be building a framework were the brands could subscribe at. That is the brand logs in the application and not the application searching for the brand. Another solution could be the hybrid version of both. This allows the application to be more scalable and flexible since there might be categories of interest and the application would not be only brand based.
In this chapter, we briefly describe the general outcome of this thesis. We wanted to prove that a ranking feature, which is defined by many factors, demands a deep analysis. On current similar systems to ours, the analysis is rather simple or sometimes non-existent, as most of them don’t declare how they produce their outcome. In addition, most of the current online reputation frameworks are commercial and don’t provide the simple end-user with the information he needs.

In our framework, a parameterized view of the current picture in social media presence can be created, giving brands a powerful tool to play with and analyse the results near real-time. A lot of improvements could be done to improve our system. We have already set some long-term goals, some of which are the following:

- To begin with, we want to include all social media applications on our system and provide additional features, coming from different sources. Facebook, LinkedIn, Google+ may provide valuable information to brands, apart from Twitter. We just wanted to implement a proof of concept using Twitter, and then expand to other social media platforms.

- Our current systems handles real-time streaming data from Twitter, which are being processed by a single server and inserted in a single database system. This approach was made as an initial effort, to utilise the infrastructure we had. However, the optimal system should use a distributed system approach, such as Hadoop\(^1\) or other similar systems, which would store all the streaming data coming from the API, also having the ability to scale for further improvements.

\(^1\)Hadoop official web site, http://hadoop.apache.org/
As a follow-up to the distributed infrastructure mentioned above, a future work could include more brands, if possible, all brands that have a Twitter profile. This would provide information that never existed before, and with the right structure it could allow the end-user to have a huge ranking framework between brands on social media, also giving the ability to edit the parameters of the analysis live.

Another feature would be to surpass the API limitations, which are mentioned on section 5.5. To our knowledge, there are commercial services that provide social media information and could be used as an external service to ours. This way, we can focus on the ranking algorithm or expand the features by the information that would be given to us.
APPENDIX A

SAMPLES OF PROGRAM’S CODE

The below BSON examples are documents of the three database collections.

A.1 Brand document

```json
{  
  "_id" : ObjectId("528f8474e4b00bfa99b2f375"),
  "category" : "Fashion",
  "imgurl" : "http://pbs.twimg.com/profile_images/3602756167/f625cb3973925d6bc29735671f8ec9f_normal.png",
  "klout_id" : "957218",
  "name" : "VSPINK",
  "profile" : [
    "@VSPINK"
  ],
  "tweets" : [
    {
      "ID" : NumberLong(425754607026192384),
      "created_at" : ISODate("2014-01-21T22:19:43.000Z"),
      "text" : "A girl honestly can’t have "too much" VSPINK clothing #socomfy",
      "retweet_count" : 0,
      "favorite_count" : 0,
      "geolocation" : null,
      "sentiment" : "neutral",
      "user" : {
        "user_id" : NumberLong(90268787),
        "followers_count" : 154,
        "friends_count" : 211,
        "location" : ""
      },
      "hashtags" : [
        "socomfy"
      ],
      "urls" : null,
      "mentions" : null
    }
  ],
  "url" : [
    {
      "shortened" : "http://t.co/I5rhqa8wG8"
    }
  ],
  "user_id" : 43158428
}
```

Listing A.1: The BSON example of a document in Brands collection
REFERENCES


[25] Pak, A. and Paroubek, P., “Twitter as a corpus for sentiment analysis and opinion mining.,” 2.1.1, 2.3
[26] Pak, A. and Paroubek, P., “Twitter as a corpus for sentiment analysis and opinion mining,” in Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC’10), (Valletta, Malta), European Language Resources Association (ELRA), May 2010. 3.1


[29] Sean, G., “How i reverse engineered klout score to an r2 = 0.94 (how to calculate klout score).” http://www.seangolliher.com/2011/uncategorized/how-i-reversed-engineered-klout-score-to-an-r2-094/, 2011. 3.5


[34] Wang, X. and Triantaphyllou, E., “Ranking irregularities when evaluating alternatives by using some multi-criteria decision analysis methods,” in Handbook of Industrial and Systems Engineering, Industrial Innovation Series, ch. 27, pp. 27–1 to 27–12, CRC Press, 2006. 2.2.1

