Topic Detection within Public Social Networks

Hugo Hermógenes Lopes da Costa Rosa

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Prof. João Paulo Baptista de Carvalho
Prof. Fernando Manuel Marques Batista

Examination Committee

Chairperson: Prof. Nuno Cavaco Gomes Horta.
Supervisor: Prof. João Paulo Baptista de Carvalho
Member of the Committee: Prof. Bruno Emanuel da Graça Martins

April 2014

Acknowledgements

Foremost, I would like to express my gratitude to Prof. João Paulo Carvalho for his guidance and direction. His clear mindedness and objective approach kept me on task, while always giving me room to try out new ideas.

I would also like to thank Prof. Fernando Batista for his continued contribution and excitement while discussing all aspects of this thesis.

This thesis was supported by national funds through FCT Fundacão para a Ciência e a Tecnologia, under project PTDC/IVC-ESCT/4919/2012 and project PEst-OE/EEI/LA0021/2013.

I dedicate this to Diana Rocha, for her love and support. She knows when to push me to do better and when to let me push myself. This became ever so clear during this thesis.

Finally, I would like to thank my parents and family for their support and trust in the education they provided me with. They always respect my decision making capabilities even when if they may disagree from the decision itself.

Resumo

A Deteção de Tópicos é uma disciplina cujo objetivo é determinar o tema de um dado documento, a partir de um conjunto de tópicos predefinidos. Através de várias técnicas de pre-processamento, o texto de um documento é transformado em algo que um classificador automático consiga ler e interpretar.

Nesta dissertação, aborda-se o tema da Deteção de Tópicos no Twitter através de uma nova técnica chamada "Twitter Topic Fuzzy Fingerprints". Inspirada na analogia das impressões digitais em cenas de crime, constrói-se uma lista de palavras-chave que compõem a impressão digital de um dado tópico. Posteriormente, avalia-se um conjunto de tweets cujo tema não esteja explicitado e calcula-se a sua semelhança com várias impressões digitais de tópicos.

Compara-se também a eficácia deste método com a de outros dois muito populares, nomeadamente Support Vector Machines (SVM) e K-Nearest Neighbors (kNN).

Os resultados apresentados revelam que o Twitter Topic Fuzzy Fingerprints obtém melhores resultados que os métodos supra mencionados. Adicionalmente, o método apresentado também é mais rápido na sua execução, o que representa um parâmetro de particular interesse quando se processa uma quantidade elevada de dados.

Palavras chave:

Deteção de Tópicos

Twitter

Fuzzy Fingerprints

Data Mining em Redes Sociais

Abstract

Topic Detection aims to determine the topic of a given document, given a predetermined large set of possible topics. Through several preprocessing techniques, it transforms its unstructured textual form into something a automated classifier can read and make decisions upon.
In this thesis, we propose to approach the subject of Tweet Topic Detection using a new technique called Twitter Topic Fuzzy Fingerprints. Inspired by the crime scene fingerprint analogy, a list of keywords that perfectly identifies a given topic is composed. Afterwards, uncategorized tweets will be matched against those fingerprints in and its similarity determined.
A comparison is made between the Twitter Topic Fuzzy Fingerprint and two popular text classification techniques, Support Vector Machines (SVM) and k-Nearest Neighbors (kNN).
The results show that Twitter Topic Fuzzy Fingerprints outperforms the other two techniques achieving better precision and recall, while still being much faster, which is an essential feature when processing large volumes of streaming data.

Keywords:
Topic Detection
Twitter
Fuzzy Fingerprints
Social Networks Data Mining

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Social networks have forever changed the way we behave. Many websites such as Facebook and Twitter, attempt to mimic real life interactions in the digital world of the Internet. Now, more than ever, anyone can go online and post their favorite song, announce a life milestone accomplishment, comment on a trending news topic or just chat.

In the 21st century, social networks have become the de facto means for individual opinion making and, consequently, one of the main shapers of an individuals perception of society and the world that surrounds him.

Twitter has become a major tool for sociological analysis. From its influence on the Arab Spring to the London riots, even news agencies have had to adapt and start using social networks as sources of information.

Twitter was even an important tool used amongst the London rioters to arrange for demonstration locations. What about the death of Michael Jackson? For two days, it was literally impossible to not come across a "RIP Michael Jackson" post.

Because of the amount of existing information in these platforms, they have become a subject of interest and analysis by many different research fields, even those outside of the information technology branch, such as Psychology and Sociology.

The scope of this particular thesis however, stays well within IT area, focusing on Text Mining and the task we shall henceforth name "Tweet Topic Detection".

According to [1], as of March 2013, Twitter has "well over 200 million active users creating over 400 million Tweets each day". It stands out from all other social networks because of its unique 140 character limit and innovative topic identification mechanism.

On the one hand, having such a short amount of information to write per tweet, the average

user will tend to either use acronyms or even misspell words to save a character or two. On the other hand, users will be more succinct when writing and will often use richer words to put across their thoughts. This presents both a challenge and a hopeful mindset towards what a computer can do to extract meaning out of tweets.

On the subject of topic detection, Twitter has created the concept of the # (hashtag), which works as a form of meta data. When anyone uses the # in front of any word, Twitter will identify it as being the topic in discussion. It even goes as far as grouping all the tweets that share the same hashtag, therefore making it easy to keep track of all posts that relate to a common theme. Let us take a look at an example:

On September 4th, 2013, the well known British comedian Ricky Gervais posted:

Dear #Australia! Brand new episode of #Derek on #ABC1 tonight at 10pm! RT if you're going to watch! pic.twitter.com/dxVf72tn80 Cheers :)

From this tweet, we know that he is promoting his new hit TV show "Derek" in Australia. Since it was perfectly hashtagged as (*#Derek*), any user could now tweet the text below and we would be certain the topic is the same.

I'm watching #Derek on #ABC1. It's hilarious.

Celebrities, institutions, politicians, television networks and frequent users understand the power of the # and use it often to guarantee maximum connection to their fans and customers. Twitter even gives users access to the current "Trending Topics", so that they can jump on that discussion, should they choose to. It is also very common to see hashtags relating to catastrophes around the world. Can one even imagine the swarm of tweets that would pour down if Twitter had been around during the 9/11 incident.

However, not everybody uses the # and many topic related tweets will go unnoticed, if not properly categorized. According to [2], out of a 1,318,323 tweet data set, only 15.9% contain hashtags and 4.5% use multiple hashtags.

In this thesis, we will use the 15.9% in order to take a closer look at the remaining 84.1%, with particular focus on English, Portuguese ans Spanish tweets, thus performing Tweet Topic Detection.

1.2 Goals

The goal of this thesis is to present a novel method in Tweet Topic Detection - Twitter Topic Fuzzy Fingerprint Algorithm.
This approach will avoid the more common and processor heavy space vector classification solutions, such as Support Vector Machine (SVM) and K-Nearest Neighbor (kNN), and focus on a score based output. In other words, the tweets will be scored from 0 to 1 in terms of similarity to a given top trend, where 0 means "not similar at all" and 1 means "totally similar".
In the process of developing this solution, the impact of several text preprocessing techniques such as stopwords, stemming and dimensionality reduction will be taken into account and its effectiveness in the Twitter language evaluated.
Finally, we compare our method with other well known and successful Topic Classification algorithms.

1.3 Contributions

The main contributions of this thesis to Topic Detection and Social Networks Data Mining, are the development of a novel method, Twitter Topic Fuzzy Fingerprints, that:

- provides good results in detecting top trends in non hashtagged tweets;
- in the context of Twitter, outperforms other well known classifiers;
- is faster than two of the best classifiers to date;

In the context of this thesis, an article named "Twitter Topic Fuzzy Fingerprints", with preliminary results on our new algorithm, was submitted and accepted to the 2014 IEEE World Congress on Computational Intelligence.
Another article, named "Detecting a tweet's topic within a large number of Portuguese Twitter trends", which studied the impact of increasing number of topic fingerprints on our method, was submitted and accepted to the 2014 Symposium on Languages, Applications ant Technologies.
In addition, a magazine article names "Efficient Twitter Topic Detection using Fuzzy Fingerprints" has recently been submitted to Applied Soft Computing.

1.4 Structure of this Document

This thesis starts out by explaining the foundations of text handling by computers. It lays down the basis for understanding the proposed method and then further details into other studies

done upon Twitter.

We then explain how Twitter provides an API for developers to work its data and processes.

In Chapter 4, we discuss several well known preprocessing techniques that were considered for our proposed algorithm in hopes that the Results chapter will prove them efficient.

In Chapter 5, the origin of the Fuzzy Fingerprint method is presented. Stemming from a previous study in a similar area, we provide the contextual differences to Twitter and then detail on the decisions made to make it more appropriate to such short documents as tweets.

Finally, the performance of the Twitter Topic Fuzzy Fingerprint algorithm is presented from results on three very distinct data sets. A detailed analysis is performed and conclusions are drawn.

Chapter 2

Related Work

In order to understand the problem with extracting and making sense of the information on social networks, we must first comprehend the basics of a concept know as Text Mining.
In this chapter, we will go through the backbone state-of-the-art techniques that support this thesis and incorporate them with the context of Twitter. We will understand how different classification algorithms work and how their success is measured. We will take a look at other projects that, much like this thesis, dive into the realm of extracting information and patterns from Twitter.

2.1 Text Mining

"Text Mining can be broadly defined as a knowledge-intensive process in which the user interacts with a document collection over time by using a suite of analysis tools" [3]. Essentially, this means that "text mining seeks to extract useful information from data sources through the identification and exploration of interesting patterns". In this thesis, our data source are the tweets and our patterns are its topics.
One of the most appealing things about this subject is that, contrary to how computers handle most data, text is an unstructured form of information. Consider the following sentence extracted from [4]:

Alice saw the rabbit with glasses

Looking at this sentence, it is hard to tell whether Alice saw a rabbit wearing glasses, or if she was might have worn glasses when she saw the rabbit. This is one of the main reasons why "the automatic understanding of text is a complex undertaking" [4].

However, if a collection of documents on the topic of "Alice in Wonderland" were to be compiled, a study of patterns and keywords would make it clear that the sentence belonged to that topic. These are the problems that Text Mining tries to provide solutions for, and are problems that resonate with this work's attempt at detecting topics on Twitter. This thesis evaluates the use of many of its pre-processing techniques and aims to compare itself with its well know categorization methods.

Although Text Mining has been around since the late 80's , it is historically dependent on a much older and large body of research in Information Retrieval and Natural Language Processing. However, it is not a replacement. It simply "seeks to find answers to questions that are difficult or impossible to answer with search engines alone" [4].

2.1.1 Information Retrieval

Information Retrieval is the process of getting the information from a collection of documents, based upon a query.

According to [4], "IR's long history began in the 1960s, when computers systems were being built to handle unstructured text. Many of these systems were large mainframes with proprietary software and interfaces to specialized document collections. In the 1980s, the PC was linked to these mainframe systems with an intermediary interface to search and retrieve data. (...) Until the mid-1990s, most IR development efforts concentrated on building comprehensive text databases and on improving performance and connectivity."

Searches begin with a need for information and is based upon explicit keywords. The user's query can sometimes be really complex and, as a result, so will the answer be. "In most cases, procedural questions are not answered in a single sentence" [4]. "The answer may be found in one or more sources".

2.1.2 Natural Language Processing

"Natural Language Processing began has a subtopic of Artificial Intelligence" [4]. "One of the original aims was to build a machine that could communicate in natural language. The two main problems that needed to be solved were understanding natural language and generating natural language".

The problem of understanding natural language is common to this thesis. According to [4], "shallow approaches were found to be successful for specific tasks", which is why "Natural Language Processing has come to mean the analysis or synthesis of natural text" as opposed to the understanding of text.

2.2 Text Categorization

One of the main goals in most studies done in this area, is to automatically classify the document, in our case, the tweet. This process, known as Text Categorization, is broadly defined by [3] as "the task to classify a given data instance into a pre-specified set of categories. (...) Given a set of categories (subjects, topics) and a collection of text documents, the process of finding the correct topic (or topics) for each document."

Text categorization uses machine learning methods to learn automatic classification rules, in which "a general inductive process builds a classifier by learning from a set of pre-classified examples" [3]. In the context of this thesis, our training set are tweets where the topic in question is perfectly identified by the hashtag. Our test set are tweets where the given # is not present.

2.2.1 Document Representation

It is very important to establish that text is in a unstructured form of data, which "common classifiers and learning algorithms cannot directly process" [3]. Thus, during the preprocessing step, our documents/tweets must be converted into a more manageable form.

The most familiar representation is the *bag-of-words* model. According to [3], "it simply uses all words in a document as features, and thus the dimension of the feature space is equal to the number of different words in all of the documents". It completely disregards syntax and word order. For instance, consider the following three sentences:

- John bought a car
- I love driving my car
- I love John

With these three texts, a dictionary is constructed as: {John, bought, a, car, I, love, driving, my}. The collection would then be a binary representation as:

$$A_{m,n} = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Figure 2.1: Binary Bag-of-Words Representation

Where $m = 3$ represents the number of documents, and $n = 8$ is the total number of unique features according to the previously mentioned dictionary.

In Figure 2.1, the words are presented in binary fashion. Which means, they only tell whether a given term exists in the document, without detailing on its importance to the collection of documents. A more common representation is the TF-IDF scheme, which combines the concept of term frequency with inverse document frequency.

2.2.2 TF-IDF

The TF-IDF scheme is a scoring method that can tell the importance of a word in a collection of documents. It can be calculated as a simple multiplication:

$$tfidf = tf \times idf \tag{2.1}$$

Term frequency (tf) can be simply the frequency (number of occurences) of the word in the document. The more common the word, the higher the term frequency will be.
On the other hand, words that occur in few documents, are probably richer in details that could better characterize the document. The inverse document frequency (idf) spans from the principle that a word that occurs in many documents is not relevant in differing each document from each other. idf can be obtained by dividing the total number of documents N by the number of documents n_i containing the term, and then taking the logarithm of that quotient, as expressed in Eq. (2.2).

$$idf = \log \frac{N}{n_i} \tag{2.2}$$

By combining the Eqs. (2.1) and (2.2), the TF-IDF of word in a document can be expressed by Eq. (2.3).

$$tfidf_i = tf \times \log \frac{N}{n_i} \tag{2.3}$$

As [5] so succinctly explains, "$tfidf$ assigns to a term, a weight in document that is:

1. highest when the term occurs many times within a small number of documents;

2. lower when the term occurs fewer times in a document, or occurs in many documents;

3. lowest when the term occurs in virtually all documents;"

There are several interesting variations of $tfidf$. In Figure 2.1 we showed an example of a the binary representation and with Eq. (2.3) we presented $tfidf$ in its most common form.

In [4], Kondachy M., suggests a normalized term frequency where "each individual word frequency is divided by the total number of content words in the document", Eq. (2.4), where w is the number of occurrences of a given term in a document with a total of W tokens.

$$tfidf_i = \frac{w}{W} \times \log \frac{N}{n_i} \tag{2.4}$$

Another common variation is presented in [6], which proposes "to use instead the logarithm of the term frequency", where $wf_{t,d}$ is the word frequency of term t in document d:

$$wf_{t,d} = \begin{cases} 1 + \log tf_{t,d} & tf_{t,d} > 0 \\ 0 & otherwise \end{cases} \tag{2.5}$$

Figure 2.2, extracted from [6], shows all the most commonly accepted variants of $tfidf$.

Term frequency		Document frequency	
n (natural)	$\mathrm{tf}_{t,d}$	n (no)	1
l (logarithm)	$1 + \log(\mathrm{tf}_{t,d})$	t (idf)	$\log \frac{N}{\mathrm{df}_t}$
a (augmented)	$0.5 + \frac{0.5 \times \mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$	p (prob idf)	$\max\{0, \log \frac{N - \mathrm{df}_t}{\mathrm{df}_t}\}$
b (boolean)	$\begin{cases} 1 & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$		
L (log ave)	$\frac{1 + \log(\mathrm{tf}_{t,d})}{1 + \log(\mathrm{ave}_{t \in d}(\mathrm{tf}_{t,d}))}$		

Figure 2.2: SMART notation for $tfidf$ variants.

2.2.3 Similarity

Now that it has been established that documents are represented like vectors in a high dimensionality space, via the bag-of-words model, how does one come to the conclusion that a training data document and a test data document are similar?

According go [3], "a similarity function takes a pair of objects and produces a real value that is a measure of the objects' proximity. To do so, the function must be able to compare the internal structure of the objects."

In Text Mining, the most common measure is the cosine similarity:

$$Sim(x_i, x_j) = (x'_i \cdot x'_j) = \sum_k x'_i k \cdot x'_j k \tag{2.6}$$

Equation (2.6) is a dot product of two vectors, where x' is a normalized vector $x' = \frac{x}{|x|}$ and values can differ from -1 to 1.

2.3 Performance Metrics

In this section we take a look at the metrics used to determine how good or poorly a classifier performs. Typically there are four key concepts: Accuracy, Precision, Recall and F-Measure. Before the formulas are presented, it is important to grasp the statistical definitions that constitute those formulas, within the scope of Twitter topic classification:

1. True Positive (TP): This means that a tweet belonging to a given topic, has been correctly identified as belonging to that topic;
2. False Positive (FP): This means that a tweet that does not belong to a given topic, has been incorrectly identified as belonging to that topic;
3. True Negative (TN): This means that a tweet that does not belong to a given topic, has been correctly identified as not belonging to that topic;
4. False Negative (FN): This means that a tweet belonging to a given topic, has been incorrectly identified as not belonging to that topic.

With this in mind, the definition of the metrics are:

$$Accuracy = \frac{\#TP + \#TN}{\#TP + \#TN + \#FP + \#FN} \tag{2.7}$$

$$Precision = \frac{\#TP}{\#TP + \#FP} \tag{2.8}$$

$$Recall = \frac{\#TP}{\#TP + \#FN} \tag{2.9}$$

$$F - Measure = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.10}$$

With accuracy, the whole of the data set is being evaluated on how often the classifier is correct in its statement. For example, an 80% accuracy rate will mean that 80% of the time, the decision of a topic belonging or not belonging is correct. This metric is typically not used because most classification methods (including ours) are good at detecting "True Negative" cases. In addition, such cases will dominate the number of occurrences when comparison to the remaining "True Positive", "False Positive" and "False Negative". As a consequence, accuracy will always be very high (above 95%) and therefore unhelpful to determine how good an algorithm really performs. Precision establishes just how correct is the method's statement of a true case. An 80% precision rate indicates that out of all the times that the method indicated a topic as being correctly

identified, 20% were actually incorrect. Precision can also be called the Positive Predictive Value.

Immediately, there is a possibility of a system being accurate but not precise.

Recall, as defined by Eq. (2.9), is a measure of how good the process is at classifying positive cases. An 80% recall rate means that in 20% of the times, the algorithm failed to identify a true positive scenario. It is also known as Sensitivity within the scope of a binary classification test.

And finally, F-Measure can be interpreted as a weighted average of the precision and recall, where the score reaches its best value at 1 and worst score at 0. Typically, it is the benchmark metric to determine the overall performance of a classifier.

2.4 Classification Methods

We will now take a look a two very widely used examples of Categorization Methods. It is important to remember that, as [4] mentions: "the results from the categorization method do not solely depend on the algorithm alone. The term supervised learning is used to describe text categorization". This means that quality of the training data has a major influence on the results.

In general, please keep in mind that most categorization algorithms follow four steps [3]:

1. Decide the categories that will be used to classify the instances;
2. Provide a training set for each of the categories;
3. Decide on the features that represent each of the instances;
4. Choose the algorithm to be used for the categorization;

The classifiers to be presented are the K-Nearest Neighbor (kNN) and the Support Vector Machine (SVM). Yang and Liu, in [7], suggest that these are the two best algorithms to date, and claim that SVM and kNN significantly outperform the other classifiers.

2.4.1 k-Nearest Neighbours

The kNN is an example-based classifier. This means it will not "build explicit declarative representations of categories but instead rely on computing the similarity between the document to be classified and the training documents" [3]. In this case, the training data is simply the "storing of the representations of the training documents together with their category labels".

In order for kNN to "decide whether a document d belongs to a category c, kNN checks whether the k training documents most similar to d belong to c. If the answer is positive for a sufficiently

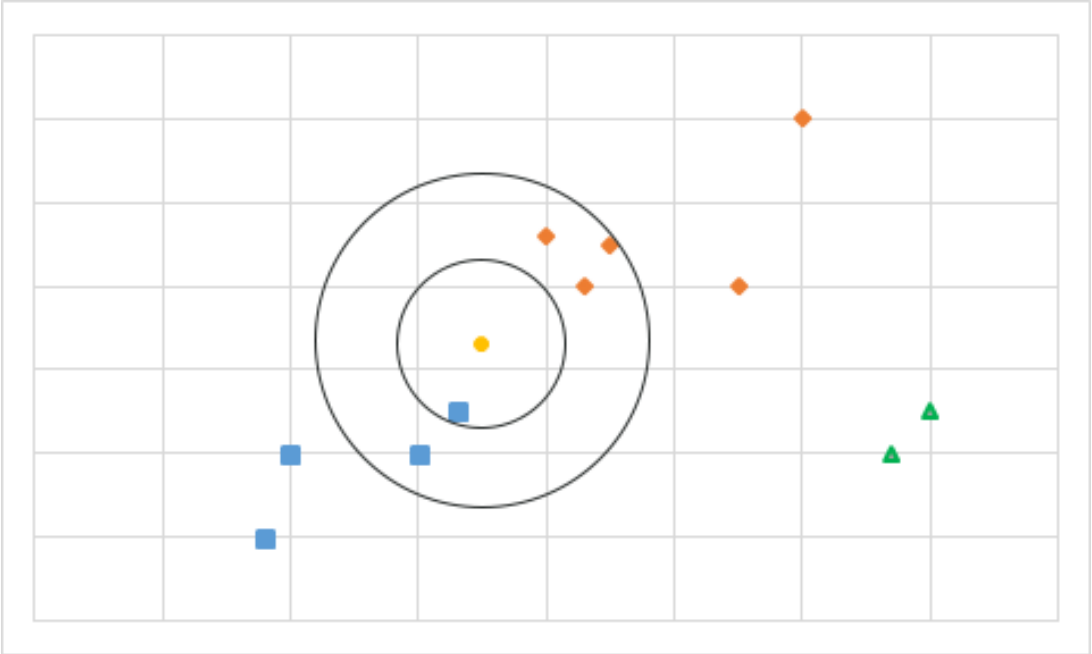

Figure 2.3: Example of the K Nearest Neighbour Algorithm

large proportion of them, a positive decision is made. The distance-weighted version of kNN is a variation that weighs the contribution of each neighbour by its similarity to the test document".

Please consider Figure 2.3, a two-dimensional example of how kNN works, where there are 3 different categories: blue, orange and green. The yellow dot represents the document to be categorized. If $k = 1$ (smaller circle), the document will look for its closest neighbor and determine that it belongs to the blue category, therefore, it will be classified as blue. However, if $k = 5$, (larger circle), it will determine that 3 of its neighbors belong to the orange category and 2 to the blue category. By a majority rule, the document will be classified as orange.

As one can deduce, an appropriate value of k is of the utmost importance. While $k = 1$ can be too simplistic, as the decision is made according only to the nearest neighbor, a high value of k can have too much noise in it and favor dominant categories. According to [4], "when the training set is large and consists of many distinct categories, then a large k is more appropriate (...) however, if the training collection is made up of documents from subtopics with some overlap, then a small k is sufficient".

This method seems to have reduced efficiency when performing on unbalanced data, i.e., where the training examples are not evenly distribute among different classes and there is usually a dominant class. According to [8] "when dealing with unbalanced data sets, this leads to either trivial classifiers that completely ignore the minority class or classifiers with many small (specific) disjuncts that tend to over fit the training example".

The kNN is considered to be one of the best performing text classifiers, whose main drawback

is "the relatively high computational cost of classification - that is, for each test document, its similarity to all of the training documents must be computed" [3]. In kNN, "the training is fast, but classification is slow. Computing all the similarities between a document that has not been categorized and a collection of documents, is slow" [4].

2.4.2 Support Vector Machine

A support vector machine (SVM) is a very fast and effective binary classifier. According to [4],"every category has a separate classifier and documents are individually matched against each category". Given the vector space model in which this method operates, geometrically speaking, [3] describes SVM as a "hyperplane in the feature space, separating the points that represent the positive instances of the category from the points that represent the negative instances. The classifying hyperplane is chosen during training as the unique hyperplane that separates the known positive instances from the known negative instances with the maximal margin".

Please consider Figure 2.4 as a two dimensional example of SVM. As one would expect, in this scenario, the hyperplanes are lines.

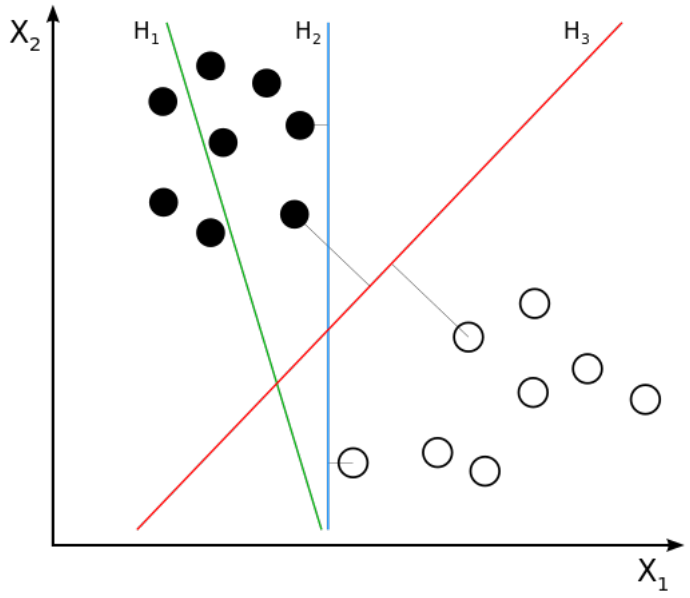

Figure 2.4: Two dimensional Support Vector Machine

As one can see in Figure 2.4, the hyperplane H_1 does not separate the positive from the negative instances. H_2 does, but it does not guarantee the maximum distance between them. Finally, H_3 offers the necessary solution.

"It is interesting to note that SVM hyperplanes are fully determined by a relatively small subset of the training instances, which are called the support vectors" [3].

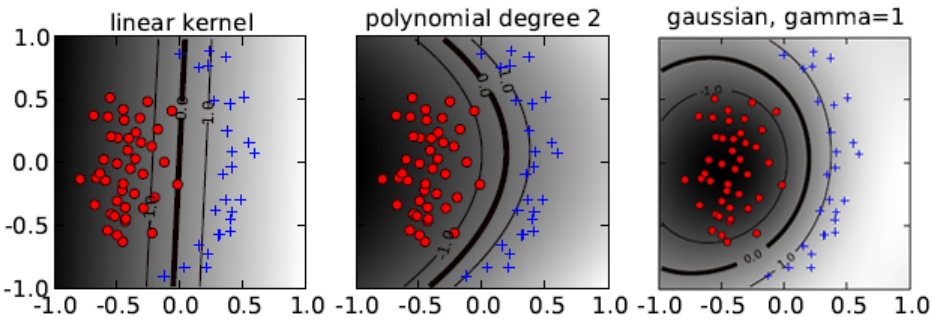

Figure 2.5: Two dimensional Support Vector Machine - Linear and Non-Linear Kernels

Further detail into this algorithm can be obtained from Ben-Hur and Weston's work in [9]: "SVM's belong to the general category of kernel methods. A kernel method is an algorithm that depends on the data only through dot-products. When this is the case, the dot product can be replaced by a kernel function which computes a dot product in some possibly high dimensional feature space.". Figure 2.4 is an example of a linear kernel. Figure 2.5 compares the behavior of different kernel functions: linear, polynomial and gaussian:

The SVM method also has another "set of parameters called hyper-parameters: The soft margin constant, C, and any parameters the kernel function may depend on (width of a Gaussian kernel or degree of a polynomial kernel).",[9].

As far as hyper-parameters are concerned, it is import to understand "the soft-margin constant, whose role is illustrated in Figure 2.6. For a large value of C a large penalty is assigned to errors/margin errors. This is seen in the left panel of Figure 2.6, where the two points closest to the hyperplane affect its orientation, resulting in a hyper-plane that comes close to several other data points. When C is decreased (right panel of the Figure 2.6), those points become margin errors; the hyperplane's orientation is changed, providing a much larger margin for the rest of the data."

To sum up, and according to [4], "SVM has at least three major differences with the previous categorization method:

1. All training documents are not used. Only documents near the classification border are used to build the SVM function;

2. An SVM can construct an irregular border to separate positive and negative training documents;

3. All features (unique words) from training documents are not necessary for classification;"

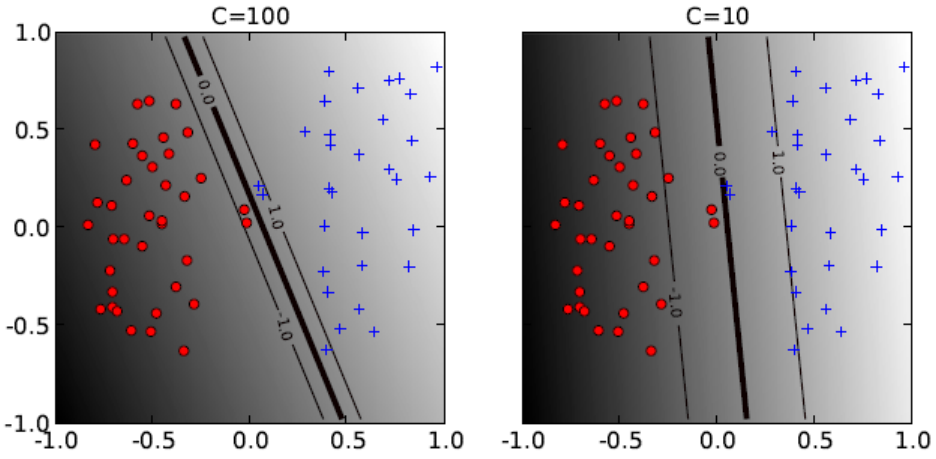

Figure 2.6: Two dimensional Support Vector Machine - The effect of the soft margin

Much like kNN, SVM has finds problems with unbalanced data sets. In [10], a point is made that since "SVM's try to minimize total error, they are inherently biased toward the majority concept. In the simplest case, a two-class space is linearly separated by an ideal separation line in the neighborhood of the majority concept. In this case, it might occur that the support vectors representing the minority concept are far away from this ideal line, and as a result, will contribute less to the final hypothesis. Moreover, if there is a lack of data representing the minority concept, there could be an imbalance of representative support vectors that can also degrade performance".

Despite that fact, "SVM methods for text categorization have attracted some attention since they are currently the most accurate classifiers" [4].

2.5 Topic Classification vs Topic Detection

During the development of this thesis, we came upon a discussion that Topic Classification and Topic Detection, although similar, may very well be two different concepts.

Topic Classification, as a concept, definitely fits into what we have presented thus far in this chapter. Given a set of predetermined categories, a topic classifier will attempt to determine whether a tweet belongs to one or more of such categories. Typically, this set of categories is short and generic, ranging from things such as politics to sports and the documents will often belong to at least one of those categories. It is very rare that a document does not fit into any topic.

In Topic Detection however, we feel there is a more objective and particular approach, where an attempt is made to determine the topic of the document, given a predetermined large set

of very distinct topics. This stems from Twitter's own nature that at any given time, millions of users are talking about millions of different subjects with millions of different hashtags. In addition, the topics are so unique amongst themselves that there is a high probability that a tweet without a hashtag may very well not belong to any of the current trends.

With this in mind, we believe that this work falls into the latter concept.

2.6 Twitter Data Mining

In this section several studies that tackle different issues with mining data from Twitter are presented.

Most ideas that have come forth, look at Twitter as a way to know what is going on in the world. For instance, in [11], a study is made in order to "retrieve in real-time the most emergent topics expressed by the community". Let us consider the examples that the authors provide in Figures 2.7 and 2.8.

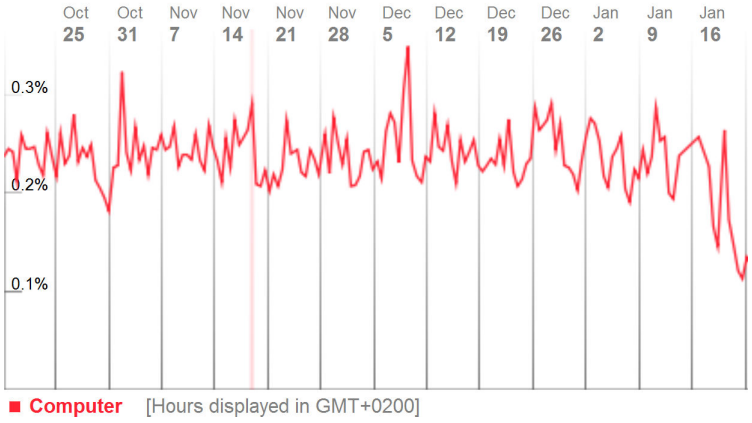

Figure 2.7: Statistical use of the term "computer" in Twitter from Oct.2009 to Jan.2010

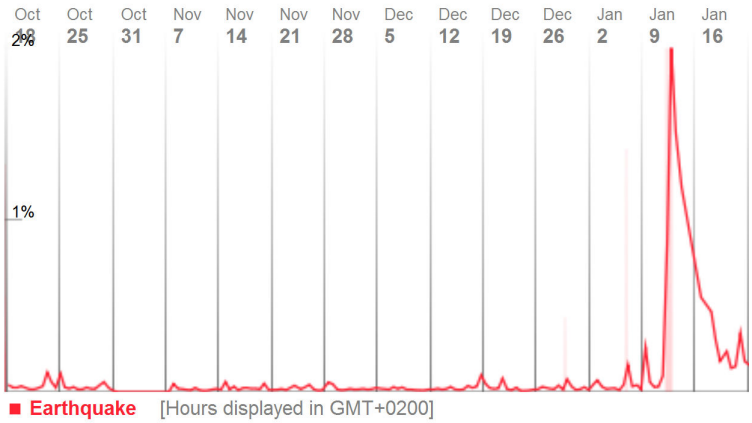

Figure 2.8: Statistical use of the term "earthquake" in Twitter from Oct.2009 to Jan.2010

As one can see, the figures show that the word "computer" is more frequent than "earthquake" through an extended period of time. However, the peek in Figure 2.8 makes the term "earthquake" part of an emergent topic, due to the catastrophe that occurred in Haiti on the 12th, January, 2010. This is a very good indicator of how Twitter is a window for important world events.

In [12], Mathioudakis and Koudas develop "a system that performs trend detection over the Twitter stream. The system identifies emerging topics (i.e. trends) on Twitter in real time and provides meaningful analytics that synthesize an accurate description of each topic". Much like [11], TwitterMonitor first identifies bursty keywords, i.e. "keywords that suddenly appear in tweets at an unusually high rate". It then groups bursty keywords into trends or, in other words, identifies a trend as a set of bursty keywords that occur frequently together in tweets. "After a trend is identified, TwitterMonitor extracts additional information from the tweets that belong to the trend, aiming to discover interesting aspects of it." [12].

The studies performed in [13] and [14] also provide interesting solutions into the problem of detecting emerging topics. However, in our work, we already assume the existence of trending topics and aim at efficient detecting tweets that are related to them, despite not being explicitly marked as so.

In [15], Weng and Lee take event detection one step further and attempt to make a distinction between relevant and irrelevant topics. They attempt to tackle the problem of tweets reporting big real life events being usually overwhelmed by high flood of trivial ones. "About 40% of all the tweets are pointless 'babbles' like 'have to get something from the minimart downstairs' " [15]. They propose EDCoW (Event Detection with Clustering of Wavelet-based Signals) which "builds signals for individual words which captures only the bursts in the words appearance.(...) It then filters away the trivial words by looking at their corresponding signal auto-correlations. EDCoW then measures the cross correlation between signals. Next, it detects the events by clustering signals together by modularity-based graph partitioning (...) To differentiate the big events from trivial ones, EDCoW also quantifies the events' significance, which depends on two factors, namely the number of words and the cross correlation among the words relating to the event."

Table 2.1 shows all the relevant topics detected by EDCow in June of 2010. ϵ represents the relevance of the topic and according to [15], an event is declared relevant when $\epsilon > 0.1$.

Unsurprisingly, most topics relate to the Football World Cup that took place in South Africa during that month.

In [16], the authors attempt to "classify Twitter Trending Topics into 18 categories such as:

Table 2.1: All the Events Detected by EDCoW in June 2010

Day	Event	ϵ value	Event Description
1-3	No event detected		
4	1. democrat, naoto	0.417	Ruling Democratic Party of Japan elected Naoto Kan as chief.
	2. ss501, suju	0.414	Korean popular bands Super Junior's and SS501's performance on mubank.
	3. music, mubank	0.401	Related to Event 2, mubank is a popular KBS entertainment show.
	4. shindong, youngsaeng	0.365	Related to Event 2, Shindong and Youngsaeng are member of the two bands.
	5. junior, eunhyuk	0.124	Related to Event 2, Eunhyuk is a member of super junior.
5	6. robben, break	0.404	No clear corresponding real-life even
6	No event detected		
7	7. kobe, kristen	0.417	Two events: Kristen Stewart won some MTV awards, and Kobe Bryant in a NBA match.
	8. #iphone4, ios4, iphone	0.416	iPhone 4 released during WWDC 2010
8	9. reformat, hamilton	0.391	No clear corresponding real-life event
	10. avocado, commence, ongoing	0.124	No clear corresponding real-life event
9	11. #failwhale, twitter	0.360	A number of users complained they could not use twitter due to over-capacity. A logo with whale is usually used to denote over-capacity.
10	12. vuvuzela, soccer	0.387	People started to talk about world cup.
11	13. #svk, #svn	0.418	#svk and #svn represent Team Slovakia and Slovenia in World Cup 2010.
12	14. #kor, greec, #gre	0.102	A match between South Korea and Greece in World Cup 2010.
13	15. whale, twitter	0.417	Similar as Event 10.
14	16. lippi, italy	0.326	Italy football team coach Marcello Lippi made some comments after a match in World Cup 2010.
15	17. drogba, ivory	0.417	Football player Drogba from Ivory Coast is given special permission to play in World Cup 2010.
	18. #prk, #bra, north	0.114	A match between North Korea and Brazil in World Cup 2010.
16	19. orchard, flood	0.357	Flood in Orchard Road.
17	20. greec, #gre, nigeria	0.122	A match between Greece and Nigeria in World Cup 2010.
18	21. #srb, podolski	0.403	A match between Germany and Serbia in World Cup 2010. Podolski is a member of Team Germany in World Cup 2010.
19-30	No event detected		

sports, politics, technology, etc". Based on previously discussion regarding Topic Classification vs Topic Detection, one can claim that this study falls into the Topic Classification.

Figure 2.9 shows the accuracy of several text-based classifiers namely the Naive Bayes Multinomial(NBM), Naive Bayes (NB) and Support Vector Machines (SVM-L) with linear kernel. "TD represents the trend definition. Model(x,y) represents classifier model used to classify topics, with x number of tweets per topic and y top frequent terms" [16].

Fig 2.10 presents the comparison of classification accuracy using different classifiers for network-based classification. Clearly, C5.0 decision tree classifier gives best classification accuracy (70.96%) followed by kNN (63.28%), SVM(54.349%) and Logistic Regression (53.457%).

In [17], the goal "is to demonstrate how to use Twitter to automatically obtain breaking news from the tweets posted by Twitter users". In 2009, when Michael Jackson died, "the first tweet was posted 20 minutes after the 911 call, which was almost an hour before the conventional news media first reported on his condition". Figure 2.11 shows the relative increase in samples of overall tweet activity regarding that subject:

In [18], Cheong M. and Lee V. "attempt to dissect the anatomy of a trending topic to find out what makes it tick". Specifically, they "select at random 4 topics which appear in the top 3 category of the trending topics list and 2 control topics (which are non-trending), gather

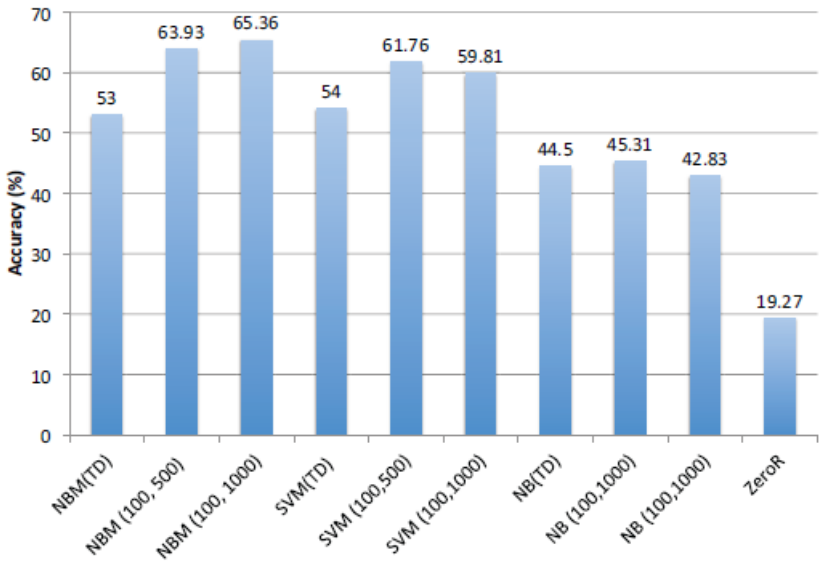

Figure 2.9: Text-based accuracy comparison over different classification techniques

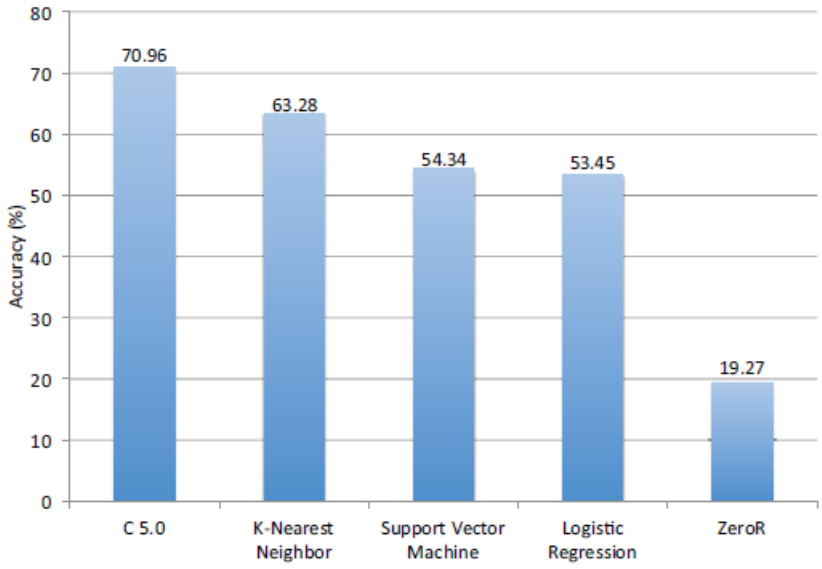

Figure 2.10: Network-based accuracy comparison over different classification techniques

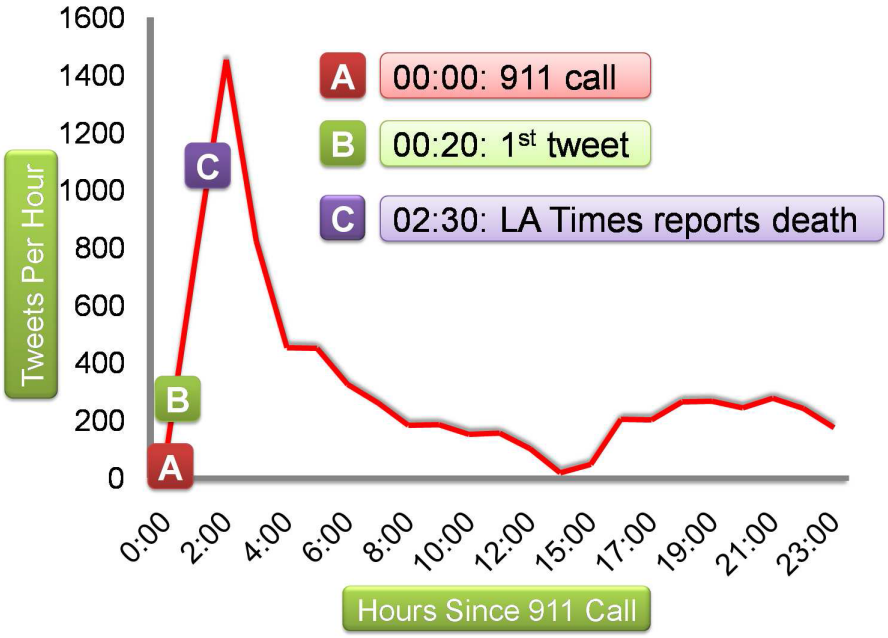

Figure 2.11: Traffic in tweets per hour, relating to Michael Jackson's death

information about the posts mentioning the topic and (...) analyse the data to investigate any patterns that occur in trending topics". The study categorizes users as being either Personal (sharing information and messaging friends), Group (fan clubs and other non-profit networks), Aggregator (news agencies), Satire and Marketing.

Regardless of the fact that the 2 control topics were "Coffee" and "revolver head", in Figure 2.12 it stands out that the "majority of Twitter users participating in chatter are users who talk about their personal life and use Twitter as a form of communication and social networking" [18].

Having established how Twitter portraits current events, this thesis will look to further engage in the theme of Tweet Topic Detection.

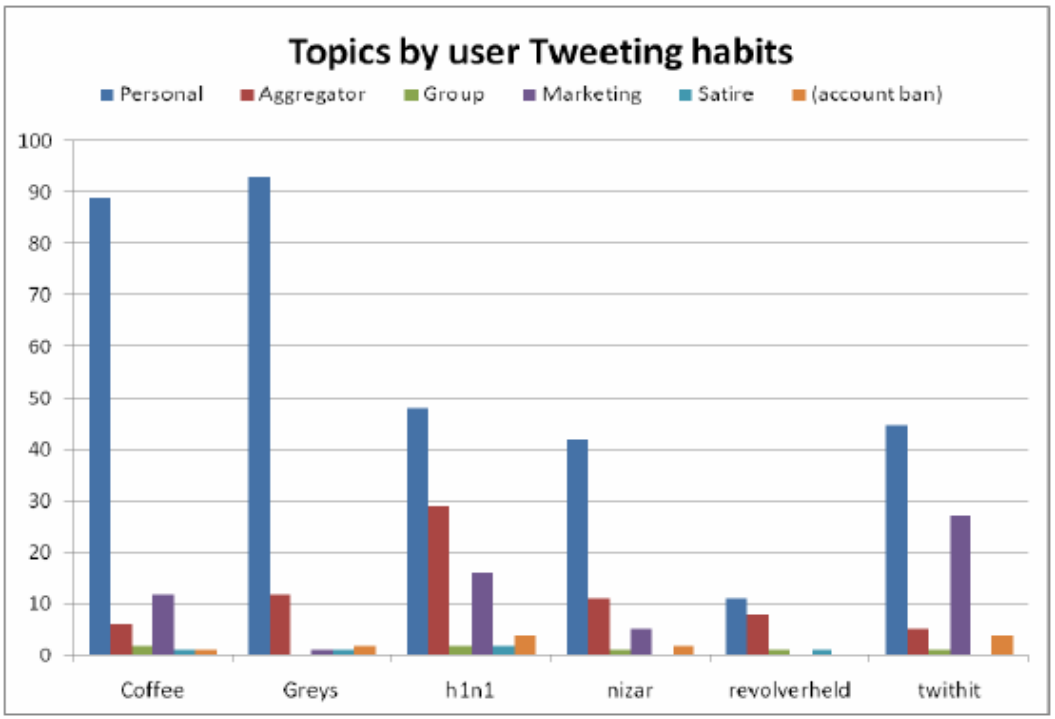

Figure 2.12: Topics by user Tweeting habits

Chapter 3

Twitter Data

Twitter provides an API for developers who wish to either explore its content or functionalities. This is divided into two separate API's: the REST API and the Streaming API.

The REST API provides simple interfaces for most Twitter functionality. With it, one can develop applications that will allow one to do pretty much anything a normal user can: post, re-tweet, message, follow, etc... This is usually a starting place for developers looking to incorporate Twitter into their apps.

The Streaming API gives developers low latency access to Twitter's global stream of tweet data. It gives access to literally million of user tweets. It is often used by developers with intentions similar to ours, who are in need of a large amount of data to work with.

In the scope of this thesis, we used one feature from each.

3.1 Public Streams

The following streams offer samples of the public data flowing through Twitter. Once applications establish a connection to a streaming endpoint, they are delivered a feed of Tweets.

It is important to note that, by opening the stream connection, one does not get all the tweets posted at that time. According to [19], with firehose access, users can get 1 % of the actual public tweets. Nonetheless, with this method, we obtained nearly two million tweets per day from the 18th of May to the 21st of May, 2013.

During the execution of this work, the language parameter became available to filter stream tweets [20]. This was promising given the intention to extract only English, Portuguese and Spanish tweets. Despite this fact, it came to our attention that re-tweeted posts often had an empty language information. Since re-tweeting is a big part of spreading a topic around, the language filter was not user and the dataset will contain tweets in languages other than those

mentioned above.

3.2 Trending Topics

The "GET trends/weekly" method returns the top trending topics for each day in a given week. It was used to get the list for the 18th of May, 2013, as it can be seen in the Appendixes. This method belongs to the REST API, version 1.0.

As of August of 2013, the method has been deprecated and replaced in version 1.1 by "GET Trends/place" where one can get the 10 top trending topics filtered by geographical location, at the time of method execution. Should a user want the worldwide top trends, the place parameter value should be "1".

With the extracted list, we used only the hashtagged topics to see how often they occurred in the two million tweets that we streamed with the Stream API, for the 18th of May, 2013. Out of the nearly two million (1950596) documents sample, approximately 11000 have the hashtags identified as being a part of the top trending topics. In spite of this fact, an overall hashtag counter revealed that 318710 tweets had at least one #, for a 16.3% value which validates Mazzia and Juett's claim in [2], that only 15.9% of tweets are hashtagged.

The low value of tweets containing the top trending topics can be explained by Twitter own view on what constitutes a trending topic. According to [21], "Twitter Trends are automatically generated by an algorithm that attempts to identify topics that are being talked about more right now than they were previously. The Trends list is designed to help people discover the most breaking news from across the world, in real-time. The Trends list captures the hottest emerging topics, not just what is most popular. Put another way, Twitter favors novelty over popularity".

Consider once again the example of *#michaeljackson* in Figure 2.11. It probably had more hits per day than a topic such as *#nowplaying* which refers to what song a user is listening at the moment. However, it was not considered to be trending until his death, when in a short period of time, more than ever before that hashtag was used. "Sometimes, popular terms do not make the Trends list because the velocity of conversation is not increasing quickly enough, relative to the baseline level of conversation happening on an average day" [21].

This definition, alongside the information that only 1% of the tweets can be streamed, explains a seemingly low presence of the top trending topics of the 18th of May.

Chapter 4

Text Preprocessing

One of the most important tasks in extracting information from text, is how the text itself is handled and represented. In this chapter, we will explain some of the most common text preprocessing techniques that were implemented in our algorithm and discuss how they will apply to our problem.

In this thesis, the tweets are not represented in a typical bag-of-words fashion, Figure 2.1. As Chapter 5 will show, we are not looking to group the tweets in clusters of categories, but instead, we aim to produce a similarity score. As a consequence, any given tweet can be represented as a vector of strings, where each string is a called a token. This process is known as Tokenization and it is broadly defined as the process of breaking a stream of text into words, phrases or symbols.

Due to Twitter's unique features, when tokenizing, very special care is taken into assuring that hashtags (e.g. *#derek*), usernames (e.g. *@rickygervais1*) and web links (e.g. *http://www.twitter.com*) are kept intact. Also, any punctuation is removed, as detailed by the code in Figure 4.1.

The Tokenization process was accomplished via Python's Regex incorporation into NLTK's Tokenizer function:

The code above ensures that any occurrence of any token starting with "www" or "http" is kept intact (web links). The same goes for tokens starting with one or more occurrences of the characters "@" and "#" (usernames and hashtags).

```
regexp_tokenize(text, pattern=
r'(www)+([.\/]+\w+)*|(http)+([.:\/]+\w+)*|\@+\w+|\w+|#+\w*')
```

Figure 4.1: Regex Expression for Tokenizing Method

Let us take another look at Ricky Gervais' tweet from Chapter 1.

Dear #Australia! Brand new episode of #Derek on #ABC1 tonight at 10pm! RT if you're going to watch! pic.twitter.com/dxVf72tn80 Cheers :)

In our algorithm, this tweet would translate to:

["Dear", "#Australia", "Brand", "new", "episode", "of", "#Derek", "on", "#ABC1", "tonight", "at", "10pm", "RT", "if", "you", "re", "going", "to", "watch", "pic.twitter.com/dxVf72tn80", "Cheers"]

Instinctively, we can spot many opportunities to make this model more meaningful. For instance, all of the words should be lower cased to ensure when comparing this tweet to a given fingerprint, the words "Cheers" and "cheers" can be identified as the same.
From this point forward, more complex techniques come into play.

4.1 Stopwords

When looking at a random text, some words will pop up very frequently. Words such as "the" or "and" offer literally no value in terms of deciphering what a text is about. For that reason, this kind of words are considered *stopwords*.
According to [5], "the several hundred most common words in English (...) are often removed from documents before any attempt to classify them". It stands to reason that this logic is valid regardless of the idiom.
For the purpose of this thesis, only English, Spanish and Portuguese tweets were taken into account. The full list of stopwords considered, can be found in Appendix A, and it is taken from [22], the Natural Language Toolkit.
With that mind, let us take another look at the list of tokens we have previously extracted from Ricky Gervais' tweet, with the lower case rule in action, and remove some stopwords.

["dear", "#australia", "brand", "new", "episode", "#derek", "#abc1", "tonight", "10pm", "rt", "re", "going", "watch", "pic.twitter.com/dxVf72tn80", "cheers"]

You can very quickly tell which words were removed:

[“of”, “on”, “at”, “if”, “you”, “to”]

As one can see, by removing the stopwords, the most import words remain and some of the noise has been removed. Not only have we removed words that provided no value to detecting the topic of this tweet, but we also reduced the number of words that classification algorithms will have to compute, which is ensures faster processing.

4.2 Short Length Words

In addition to stopwords, one can argue that some words are so short that they provide little value into disclosing a tweet's topic. Words such as the colloquial “hi” or even “bye” could, theoretically, be removed from the corpus without damaging any algorithm's chance of classifying it. As a consequence, the list of tokens that comprises a tweet will be shorter and also improve the algorithm's processing time. In the context of a bag-of-words representation, Figure 2.1, this process is known as Dimensionality Reduction.
But there are risks to this approach. Namely, some very famous acronyms such as “USA” and “UK” could be overlooked, despite being helpful in adding information to the tweet. Let us, once more, look at our representation Ricky's tweet, after the removal of stopwords:

[“dear”, “#australia”, “brand”, “new”, “episode”, “#derek”, “#abc1”, “tonight”, “10pm“, “rt”, “re”, “going”, “watch”, “pic.twitter.com/dxVf72tn80”, “cheers”]

Now, let us indulge the argument made about removing short words, and remove all words with 4 or fewer characters. The remaining tokens would be:

[“#australia”, “brand”, “episode”, “#derek”, “#abc1”, “tonight”, “going”, “watch”, “pic.twitter.com/dxVf72tn80”, “cheers”]

You can very quickly tell which words were removed:

[“dear”, “new”, “re”, “rt”, “10pm”]

The tweet remains easily classifiable while being easier to process, due to reduced number of features.
In this thesis, the length of a word is parameter taken into account while executing, and results

will be compared based of the removal of shorter than j-sized words from the corpus. During our experiments, j can take any value from 1 to 4 and conclusions will be drawn from the "Test and Results" chapter.

4.3 Stemming

Stemming is the name given to the process of reducing words to their root (stem) form. The stem form is achieved after the "prefix and suffix have been removed" [4]. For instance, the stem form of the word "running" is "run". Consider the example in [4], with the root word "prevent": "variants of this root word are achieved by adding suffixes to create *prevents*, *preventing* and *prevention*."

By incorporating this technique into the text preprocessing, it further achieves the goal of reducing the dimensionality of the document's representation, while keeping the meaning intact. Although prefixes such as "un-" give opposite significance to a word, the purpose of it its kept by the stem word. Both "tie" and "untie" relate to either the action or substantive "tie". Although available in several languages, it is the English language that stands out with Porter's Stemmer [23]. According to [24], it "became the de-facto standard for word stemming. This algorithm works by applying a set of different rules, yielding to the word stem after 5 iterations (so-called 'steps')".

The Stemming mechanism is one of the parameters used in this thesis. The impact of this mechanism on the results will be taken into account on the "Tests and Results" chapter.

4.4 Word Spell Correction

As Twitter grows, it also creates its own lingo. User often use acronym such as "LOL' (Laughing Out Loud)' and "OMG" (Oh My God), but they also mistype a lot. It can be done unconsciously or purposefully. For instance, in order to enhance a sentiment about something:

I looooooooooooove Ricky's new show #derek

As a consequence, this thesis took under consideration the use of a word corrector, during the text preprocessing phase, by Carvalho J. and Coheur L. in [25].

According to [25],the algorithm, named UWS, "was developed with the goal of automatic detection and correction of typographical and other word errors in unedited corpus data when creating word lists".

It scores words in similarity from 0 to 1, where "0.68 was considered as the limit over which false positives and false negatives are highly unlikely,i.e., where both precision and recall are close to 1" [25].
It stands out as being very precise and fast, with precision results of 95.3%, a recall of 82% and F-Measure of 88.3%. In terms of quickness, it is approximately 285% faster than the Damerau-Levenshtein approach,[26], when comparing similarity of a word with 5 characters in length, with a dictionary of 109583 words (it took 2 seconds).
Ultimately, it was not used because, as quick and efficient it may be, when streaming millions of tweets, it would delay the process just enough so that our proposed method would loose its appeal to possibly become a near real-time classifier.

4.5 Twitter Username

The owner of the tweet can provide valuable information towards discovering the theme of it. For example, it his likely that music related personalities, will tweet about music trends quite often.
Although the author of the tweet is not technically a part of its content, it is additional metadata that can help shed a light on the topic under discussion. As a consequence, in this thesis, the author of the tweet is a part of the tweet features.
Despite being clear that a user that often posts about the same subject can be helpful, a user that posts often about several topics, may be damaging to make hashtags distinguishable. This is solved using Inverse Document Frequency, IDF, as explained in Chapter 5.

Chapter 5

The Fuzzy Fingerprint Algorithm

The proposed method is derived from Nuno Homem's and João Paulo Carvalho's work in [27]. In that thesis, the researchers tackle the problem of text authorship and use the crime scene fingerprint analogy to claim that a given text has its authors writing style embedded in it. Therefore, having the fingerprint been previously defined, it should be entirely possible to identify if, a text whose author is unknown, has a known author's fingerprint on it.
In this chapter, we will detail on how the original method works and how it adapts to the Twitter reality, thus establishing that given a topic's fingerprint, a measure of similarity can be calculated between topic and tweet.

5.1 Original Method

According to [27], "the main concept behind this algorithm is that authors have a stable enough behavior that allows a set of features to be extracted, fuzzified and then compared. The most frequent words in the texts of a single author present the required stability". The "set of words to consider in the fingerprint should be large enough to allow a comprehensive sample of the author style and vocabulary".However, to be useful, the fingerprint should comply with some basic criteria:

1. Include a minimal set of features that describe the author in a compact format.

2. Allow for update operations whenever new information (texts) on the author is available.

3. Allow for a fast comparison process once a new text needs to be identified.

4. Scalability, i.e., performance should not degrade significantly when the number of texts or authors in the pool increases.

5. Flexibility, i.e., should allow new authors to be included in the process, whenever information is available.

In order to complete the above mentioned step 1, "an approximated algorithm is used for this purpose since classical exact top-k algorithms are inefficient and require the full list of distinct elements to be kept (storing 100000 words per author is inefficient if only the top-1000 are needed.) The Filtered Space-Saving algorithm is used for this purpose since it provides a fast and compact answer to the top-k problem".
The algorithm itself works like this:

1. Gather the top-k word frequencies in all known texts of each known author;

2. Once the top-k word frequencies are available, the fingerprint is constructed by applying a fuzzifying function to the word frequencies;

3. Perform the same calculations for the text being identified and then to compare this text fuzzy fingerprint with all the available author fuzzy fingerprints. The most similar fingerprint is chosen and the text is assigned to the fingerprint author;

5.1.1 The Filtered Space-Saving Algorithm

According to [27], "to allow the use of authorship identification techniques in near real time and for a large number of potential authors and documents, a key issue is to be able to extract the relevant features using an efficient algorithm with reduced memory usage. In this case, features are the most frequent words in the author's texts. The choice was to use an approximate top-k algorithm capable of generating good quality estimates using a reduced memory footprint".
The Filtered Space-Saving (FSS) algorithm uses "a bitmap counter with h cells, each containing two values, α_i and c_i, standing for the error and the number of monitored elements in cell i. An hash function that transforms the input values (words) into an uniformly distributed integer range is used to obtain $h(x)$. The hashed value $h(x)$ is then used to increment the corresponding cell on the bitmap counter. Initially all values of α_i and c_i are set to 0" [27].
There is a second a list of monitored elements A with size m. The list is initially empty. Homem N. and Carvalho J. say that "each element contains three parts; the value itself v_j, the estimate count f_j and the associated error e_j. The minimum required value to be included in the monitored list is always the minimum of the estimate counts, $\mu = min\{f_j\}$. While the list has free elements, the minimum is set to 0."
But how does the algorithm work? According to [27] "when a new word is received, its hash is calculated and the bitmap counter is checked. If there are already monitored elements with

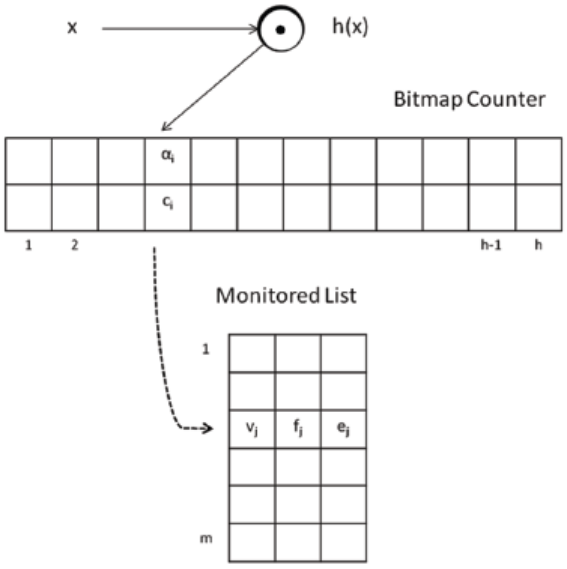

Figure 5.1: Featured Space-Saving Algorithm Diagram

that same hash ($c_i > 0$) the list is searched to see if this particular element is already there. If the element is in the list then the estimate count f_j is incremented. If the element is not in the list then it is checked to see if it should be added. A new element will be inserted into the list if $\alpha_i + 1 \geqslant \mu$. If the element is not monitored then α_i is incremented."

Finally, "if the element is included in the monitored list, then c_i is incremented and set $f_j = \alpha_i + 1$ and $e_j = \alpha_i$. If the list has exceeded its maximum allowed size, then the element with the lower f_j is selected.The selected element is removed from the list, the corresponding bitmap counter cell is updated, c_j is decreased and α_i is set with the maximum error incurred for that position in a single element" [27].

5.1.2 Membership Functions

One of the most important aspects in this method, is the fuzzifying function and "the chosen approach is to assign a membership value to each word in the set based only on the order in the list". In this case, the order of the frequency was chosen. The more frequent words will have a higher membership value.

The authors present three different attribution functions for each element i of the top-k list:

$$\mu(i) = \frac{k - i}{k} \tag{5.1}$$

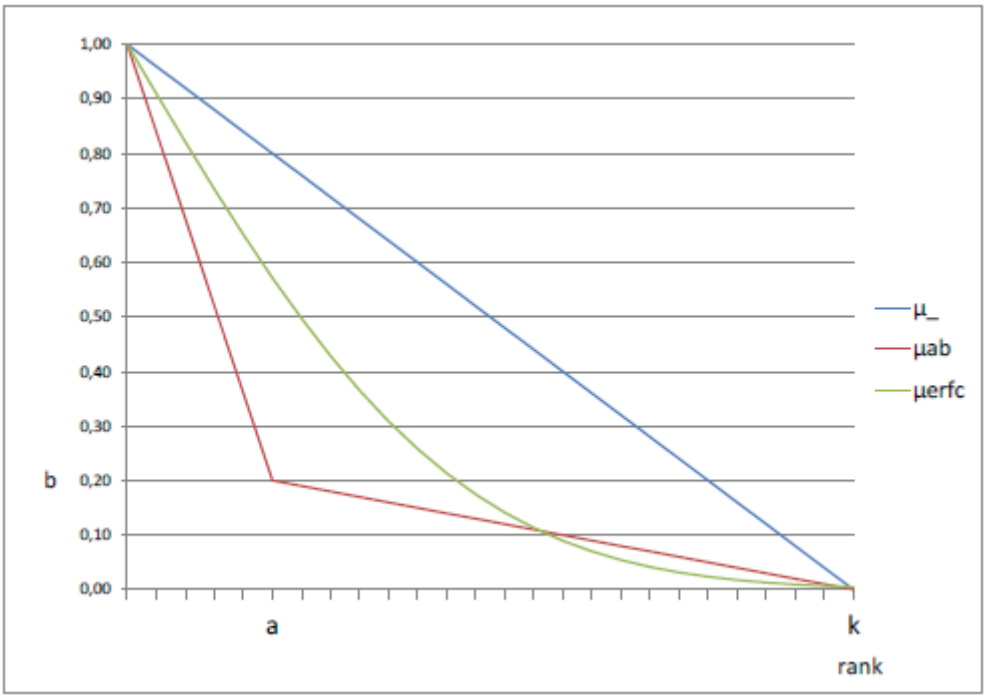

Figure 5.2: Fuzzyfing functions

$$\mu_{ab}(i) = \begin{cases} 1 - (1-b)\frac{i}{kb} & i < a \\ \frac{a(1-\frac{i-a}{k-a})}{k} & i \geq a \end{cases} \tag{5.2}$$

$$\mu_{erfc}(i) = 1 - erf(\frac{2i}{k}) \tag{5.3}$$

where *erf* is the Gauss error function. Figure 5.2 presents the used functions.

Consider an ordered k-sized vector containing the most frequent words of a given authors. This is his fingerprint. The more frequent words are weighed to be more important by attaining a higher value through either 5.1, 5.2 or 5.3. The first word (*i=0*) has a weight of 1.0 on any fingerprint, regardless of the fuzzyfing function used. The second word (*i=1*) will have a lower value, and it will be different depending on the which function was used. And so on, and so on.

5.1.3 Similarity Score

So how do the authors propose to find the author of a given unknown text D? "One starts by computing the size-k fingerprint of D, $\Phi(D)$. Then one compares the fingerprint of D with the fingerprints $\Phi(k)$ of all authors present in the fingerprint library. Authorship is attributed to the author j that has the most similar fingerprint to $\Phi(D)$. Fingerprint similarity is calculated using:"

$$sim\Phi_{D,J} = \sum \frac{\min(\mu_v(\Phi(D)), \mu_v(\Phi(j))}{k} \tag{5.4}$$

where $\mu_v\Phi(x)$ is the membership value associated with the rank of word v in fingerprint x.

5.2 Twitter Topic Fuzzy Fingerprints

Our method, while similar in intention and form, differs in a few crucial steps.

First it is important to establish the parallel between the context of author ownership and Tweet Topic Detection. Instead of author fingerprints, in this work we are looking to obtain the fingerprints of hashtagged Twitter topics (#). Once we have a topics fingerprint library, each unclassified tweet can be processed and compared to the fingerprints existing in the topic library.

Secondly, different criteria were used in selecting the top-k words for the fingerprint. While [27] uses word frequency as the main feature to create the top-k list, here we use an adaptation of an Inverse Document Frequency technique, aiming reducing the importance of frequent terms that are common across several topics, such as "follow", "RT" and "like".

Lastly, the similarity score differs from the original, based on the fact that tweets are, by design, very short texts, while the original Fuzzy Fingerprint method was devised to classify much longer texts (newspaper articles, books, etc. ranging from thousands to millions of characters). Here we propose the use of a normalized score with values between 0 and 1, where the lowest score indicates that the tweet in question is in no way similar to the topic fingerprint, and the highest value indicates that the tweet is totally similar. In addition, the Filtered Space-Saving algorithm was not included, although it remains a feature of interest for future work.

5.2.1 Building the FingerPrint

This step is mostly similar to the original. The method will go over the training set, which, in this situation, are tweets containing the Trending Topics of the day. For each, tweet it will acknowledge the existence of the # and add that word to a hash table alongside with its counter of occurrences.

Consider Table 5.1 as an example of the end result that algorithm in Figure 5.3 would produce, after going through several tweets of some previously mentioned topics:

The main difference between the original method and ours, is that due to the small size of each tweet, its words should be as unique as possible in order to make the fingerprints distinguishable amongst the various topics. Therefore, in addition to counting each word occurrence, we also

```
createDataStructure(trainingSet,topTrends)
        trendFP = Set of topicFingerprints()
        for tweet in trainingSet
                tokens = tokenize(tweet)
                kw = words in tokens and topTrends
                for k in kw
                        for t in tokens
                                if t not in topTrends
                                        trendFP{k}[t]++
```

Figure 5.3: Pseudo-Code to explain data structure used

account for of its Inverse Topic Frequency (ITF), an adaptation of the Inverse Document Frequency in Eq. (2.2), where N becomes the topic fingerprint library size (i.e., the total number of topics), and n_i becomes the number of #topics where the word is present.
Table 5.1 shows an example of a possible top-k output produced by the algorithm Figure 5.3 for a fingerprint size $k = 3$, after going through a small training set. By multiplying the occurrences of each word per topic with its ITF, we obtain the third column of Table 5.1.

Table 5.1: Fingerprint hash table before and after ITF

Key	Feature	Counter	Feature	ITF
#michaeljackson	dead	4	dead	1.90
	rip	2	rip	0.95
	sing	1	sing	0.48
#haiti	earthquake	10	earthquake	4.77
	rip	5	rip	1.43
	help	1	help	0.17
#derek	show	8	show	3.81
	help	3	australia	0.95
	australia	2	help	0.52

As expected, the term “help”, which was the only one that occurred in more than one fingerprint, got dropped to last position in the ranking of fingerprint words for the topic “#derek”.

5.2.2 Membership Functions

Now that we already have the fingerprint for a top-k with ITF, we take the same approach as the original method, and use a membership function to calculate the weight of each word, according to its position in the ranking.

In [27], results showed that μ_{ab}, as described by Eq. (5.2), provided the best results. Through extensive testing, we reached the same conclusion. In addition, it is also computationally simpler which provides slightly faster results, a feature of great importance in the scope of this thesis.

5.2.3 Tweet-Topic Similarity Score

Once the fingerprints have been established, and each membership word value given according to Eq. (5.2), every tweet in the test data set is used to calculate a similarity score to every fingerprint.

```
for tweet in testSet:
   for fp in fingerprint:
      value = similarity(tweet,fp)
```

Figure 5.4: Pseudo-Code to explain the similarity calculation

In the original method, Eq. (5.2), in order to check the authorship of a given document, a fingerprint would be built for the document (using the procedure described above), and then the document fingerprint would be compared with each fingerprint present in the library.
Within the Twitter context, such approach would not work due to the very small number of words contained in one tweet - it simply does not make sense to count the number of individual word occurrences. Therefore we developed a Tweet-Topic Similarity Score (T2S2) that tests how much a tweet fits to a given topic. The T2S2 function, Eq. (5.5), provides a normalized value ranging between 0 and 1, that takes into account the size of the (preprocessed) tweet (i.e., its number of features).

$$T2S2(\Phi, T) = \frac{\sum_{v} \mu_{\Phi}(v) : v \in (\Phi \cap T)}{\sum_{i=0}^{j} \mu_{\Phi}(w_i)} \tag{5.5}$$

In Eq. (5.5) Φ is the #topic fingerprint, T is the set of words of the (preprocessed) tweet, $\mu_{\Phi}(v)$ is the membership degree of word v in the topic fingerprint, and j is the is the number of features of the tweet.
Essentially, T2S2 divides the sum of the membership values $\mu_{\Phi}(v)$ of every word v that is common between the tweet and the #topic fingerprint, by the sum of the top j membership values in $\mu_{\Phi}(w_i)$ where $w \in (\Phi)$.
Equation 5.5 will tend to 1.0 when most to all features of the tweet belong to the top words of the fingerprint, and tend to 0.0 when none or very few features of the tweet belong to the bottom words of the fingerprint.

Let us consider Table 5.1 fingerprint for the topic "#derek". Using the function from Eq. (5.2), it's membership values for a $k = 3$ sized fingerprint, will be { *"show", 1.000; "australia", 0.1667; "help", 0.083*}.

Now consider the following example of an uncategorized tweet on the subject:

I love Ricky's show

Which, after being tokenized and preprocessed, would translate into the following vector of words { *"love", "ricky", "show"*}. The only word in the tweet that belongs to the fingerprint is "show", and it has a membership value of 1.0. The calculation made would be:

$$T2S2 = \frac{1.000}{1.000 + 0.167 + 0.083} = 0.800 \tag{5.6}$$

That tweet has an T2S2= 0.800 similarity to the topic #derek. Which, if one thinks about it, is a pretty good value. After all, this is a short tweet that talks about someone's love one of Ricky's show. But which show? The Office? Extras? Derek?

On the other hand, other short tweets such as "*I love Jerry's show*" would also score the same and not belong to #derek. This is ultimately a small example with few features and a poor fingerprint. The next chapter will shed some more light on the advantages and disadvantages of this approach.

Also in the next chapter, we will look at results with different threshold T2S2 values for what is determined to be a tweet belonging to a certain top trend. These values will range from 0.5 to 0.05.

To sum up, Eq. (5.5) will:

- tend to 1.0 when most to all features of the tweet belong to the top words of the fingerprint;
- tend to 0.0 when fewer to none features of the tweet belong to the bottom words of the fingerprint.

Chapter 6

Tests and Results

In this chapter, we will present results on the tests made to the Twitter Topic Fuzzy Fingerprints method.
Using the metrics presented in Chapter 2, we will determine how good the algorithm is and explain the features of three distinct datasets. Results will be provided with different parameter settings (scenarios) and conclusions will be drawn on which conditions are more beneficial to this approach.

6.1 Parameters

Several testing scenarios were put in place, in order to find the algorithms' optimal performance setting. For every test below, these are the parameter that were put into play:

1. k, is the size of the fingerprint. It determines how many of its the most important words are used to compare with the tweets features. Several increasing values were taken into to account, in order to determine whether a bigger k value would provide better results;

2. *stopwords*. For each scenario, the results provided were measured with and without the removal of stopwords. This aims to ascertain the true impact of the removal of stopwords;

3. *stemming*. For each scenario, the option to return words in their stem form can be either turned on or off. With this parameter, we aim to determine the impact of this preprocessing technique towards getting better results.

4. *minimum j sized words*. For each scenario, different values of j were considered as being the minimum size of the words to feature in the tweets list of terms. The purpose of this variable, is to test how the removal of small words may help keep richer tokens and get better results;

5. *threshold value*. It represents the T2S2 value from which our method will declare that a certain tweet belongs to a given trend. For the purpose of this thesis, values of 0.5, 0.15, 0.10 and 0.05 were tested.

6.2 Data Set I

The first data set used to test the Twitter Topic Fuzzy Fingerprints, was constructed from the tweets streamed from the 18th of May to 21st of May, 2013. A brief glimpse of this data has been given in Chapter 3.

6.2.1 Training Set

The Training Data Set consists of approximately 21000 tweets containing the hashtag for 21 impartially chosen topics of interest out of the top trends of the 18th of May, 2013. Approximately 4% of which are Portuguese, 35% are Spanish, 60% are English and 1% are in other languages. In addition to the content of the tweet itself, the author's username is also included into the list of features of each tweet.

Please consult Appendix A, Table A.1, to analyze the distribution of top trend hashtags across these tweets. As one can surely see, the distribution of tweets per topic is uneven. While 7959 of those tweets contain the trend *#nowplaying* (nearly 35%) , *#mtvonedirection* only appears in 145 tweets and *#5hfridayquestions* even shows up only once.

This makes for what is known as an unbalanced dataset, where the categories do not have the same number of sample documents. Additionally, one single category can even dominate the training set in such fashion, that some classifiers may incorrectly categorize most of the test set as that being a part of that one category.

6.2.2 Tests

The Test Data Set consists of 585 tweets that do not contain any of the top trending hashtags, although they may contain others. Each tweet was impartially annotated to belong to one of the 21 chosen top trends. Its distribution can be find in Appendix A, Table A.2.

Finding tweets that clearly belonged to the main list of hashtags was a challenge, given how few we came across. This speaks to the difference between Topic Classification and Topic Detection, that was introduced in Chapter 2.

Table 6.1: Results for Data Set I - no stemming and no stopword removal

		Threshold 0.5			Threshold 0.15			Threshold 0.10			Threshold 0.05		
j	k	Pre	Rec	F-Mea	Pre	Rec	F-Mea	Pre	Rec	F-Mea	Pre	Rec	F-Mea
1	10	0.908	0.270	0.416	0.662	0.754	0.705	0.645	0.768	0.701	0.591	0.947	0.728
1	15	0.955	0.217	0.353	0.654	0.773	0.708	0.645	0.816	0.720	0.587	0.950	0.726
1	20	0.979	0.164	0.280	0.653	0.790	0.715	0.633	0.830	0.718	0.604	0.948	0.738
1	25	0.968	0.105	0.189	0.647	0.799	0.715	0.630	0.881	0.735	0.575	0.952	0.717
1	30	0.962	0.086	0.158	0.661	0.800	0.724	0.628	0.928	0.749	0.551	0.957	0.699
1	50	0.900	0.031	0.060	0.688	0.788	0.735	0.609	0.943	0.740	0.525	0.967	0.680
1	100	0.833	0.017	0.034	0.700	0.714	0.707	0.626	0.923	0.746	0.490	0.988	0.655
1	250	0.769	0.017	0.034	0.673	0.795	0.729	0.570	0.945	0.711	0.407	0.993	0.578
2	10	0.913	0.270	0.417	0.735	0.754	0.744	0.706	0.771	0.737	0.619	0.948	0.749
2	15	0.954	0.213	0.349	0.717	0.776	0.745	0.700	0.819	0.755	0.611	0.952	0.744
2	20	0.962	0.176	0.297	0.713	0.792	0.750	0.682	0.835	0.751	0.640	0.952	0.765
2	25	0.958	0.119	0.211	0.699	0.799	0.745	0.680	0.883	0.769	0.595	0.952	0.732
2	30	0.955	0.110	0.198	0.708	0.804	0.753	0.672	0.933	0.781	0.584	0.959	0.726
2	50	0.920	0.040	0.076	0.721	0.793	0.756	0.637	0.950	0.762	0.547	0.969	0.699
2	100	0.842	0.028	0.053	0.733	0.737	0.735	0.653	0.928	0.767	0.507	0.991	0.671
2	250	0.826	0.033	0.063	0.691	0.823	0.751	0.585	0.952	0.725	0.411	0.995	0.581
3	10	0.906	0.267	0.412	0.736	0.745	0.741	0.741	0.769	0.755	0.687	0.947	0.796
3	15	0.958	0.238	0.381	0.742	0.778	0.760	0.748	0.816	0.780	0.682	0.952	0.795
3	20	0.954	0.213	0.349	0.743	0.802	0.772	0.756	0.928	**0.833**	0.705	0.952	0.810
3	25	0.950	0.165	0.282	0.747	0.854	0.797	0.746	0.936	0.831	0.679	0.955	0.794
3	30	0.955	0.146	0.254	0.748	0.857	0.799	0.725	0.941	0.819	0.665	0.962	0.787
3	50	0.926	0.086	0.157	0.764	0.831	0.796	0.704	0.948	0.808	0.643	0.966	0.772
3	100	0.913	0.072	0.134	0.780	0.799	0.789	0.693	0.952	0.802	0.585	0.988	0.734
3	250	0.881	0.102	0.182	0.708	0.861	0.777	0.615	0.964	0.751	0.445	0.993	0.614
4	10	0.904	0.306	0.458	0.734	0.735	0.734	0.737	0.752	0.744	0.658	0.830	0.734
4	15	0.935	0.224	0.361	0.738	0.768	0.753	0.744	0.799	0.770	0.655	0.831	0.733
4	20	0.944	0.203	0.334	0.739	0.785	0.761	0.732	0.814	0.771	0.667	0.845	0.746
4	25	0.946	0.181	0.303	0.733	0.788	0.760	0.716	0.821	0.765	0.652	0.854	0.739
4	30	0.949	0.160	0.274	0.733	0.793	0.762	0.682	0.828	0.748	0.642	0.857	0.734
4	50	0.939	0.107	0.192	0.735	0.756	0.745	0.665	0.845	0.745	0.619	0.862	0.720
4	100	0.915	0.093	0.169	0.721	0.726	0.724	0.637	0.842	0.725	0.567	0.878	0.689
4	250	0.841	0.127	0.221	0.658	0.775	0.711	0.566	0.866	0.685	0.435	0.919	0.590

6.2.2.1 Scenario A - NO Stemming and NO Stopword Removal

In this section, we consider the several scenarios that k (size of the fingerprint) and j (minimum size of the tweet features) take when considering that neither stemming nor stopword removal is put into practice. Please consider Table 6.1.

An early analysis seems to suggest that there is inverse relation between precision and recall. As the top-k value increases, so does recall, while the precision value decreases.

According to Eq. (2.8) and (2.9), this can be explained by the fact that as k increases, T2S2 values will also increase. This means that previous cases of "Negative" will now tend to be determined as "False Positive", thus decreasing precision. On the other side of this consequence, tweets that had been determined as "False Negative" will enter the "Positive" frame, therefore increasing the recall.

An additional look at the results of Table 6.1 also suggests that the threshold value of 0.5 provides a weak recall and f-measure values. Considering that the latter metric is a weighted average of recall and precision, one can use it to determine what is the best performance scenario above. By the f-measure standard, its maximum value of 0.833 is reached for threshold 0.10 when the minimum size word $j = 3$ and top-$k = 20$.

6.2.2.2 Scenario B - NO Stemming and YES Stopword Removal

In this section, we consider the several scenarios that k (size of the fingerprint) and j (minimum size of the tweet features) take when considering the stemming is not used but stopword removal is.

An early look at the results in Table 6.2 seem to keep the previously mentioned relation between precision and recall. As the first decreases, the second increases. The removal of stopwords provides slightly higher values all around, which can be shown by the highest f-measure value so far, 0.841. This is result is obtained with minimum word size $j = 1$, top-$k = 25$ and a threshold value of 0.10.

6.2.2.3 Scenario C - YES Stemming and NO Stopword Removal

In this section, we consider the several scenarios that k (size of the fingerprint) and j (minimum size of the tweet features) take when considering the stemming is used but stopword removal is not. Please conside Table 6.3.

Initial conclusions regarding the inverse relation between precision and recall continue to be the same. While the first decreases, the latter decreases. Best case scenario occurs for an f-measure value of 0.833 when $j = 3$, top-$k = 20$ and threshold= 0.10. These values are exactly the same as those in Table 6.1, which seems to suggest that stemming provides little to no impact. Nonetheless, this will be further looked into in the following section of this chapter.

Table 6.2: Results for Data Set I - no stemming but with stopword removal

		Threshold 0.5			Threshold 0.15			Threshold 0.10			Threshold 0.05		
j	k	Pre	Rec	F-Mea	Pre	Rec	F-Mea	Pre	Rec	F-Mea	Pre	Rec	F-Mea
1	10	0.919	0.272	0.420	0.737	0.747	0.742	0.740	0.769	0.754	0.713	0.947	0.814
1	15	0.951	0.236	0.378	0.736	0.769	0.753	0.747	0.819	0.782	0.712	0.952	0.814
1	20	0.959	0.203	0.335	0.741	0.799	0.769	0.767	0.928	0.840	0.731	0.950	0.826
1	25	0.962	0.172	0.292	0.753	0.854	0.800	0.760	0.940	**0.841**	0.714	0.950	0.815
1	30	0.951	0.134	0.235	0.752	0.859	0.802	0.744	0.941	0.831	0.680	0.957	0.795
1	50	0.927	0.065	0.122	0.771	0.840	0.804	0.723	0.950	0.821	0.662	0.959	0.783
1	100	0.818	0.031	0.060	0.787	0.787	0.787	0.711	0.924	0.804	0.619	0.986	0.761
1	250	0.892	0.057	0.107	0.727	0.852	0.784	0.666	0.950	0.783	0.523	0.993	0.685
2	10	0.912	0.284	0.433	0.738	0.747	0.743	0.741	0.769	0.755	0.685	0.947	0.795
2	15	0.952	0.238	0.380	0.740	0.773	0.756	0.750	0.819	0.783	0.684	0.952	0.796
2	20	0.953	0.210	0.344	0.744	0.802	0.772	0.759	0.935	0.838	0.711	0.950	0.814
2	25	0.956	0.188	0.314	0.749	0.864	0.803	0.755	0.940	0.837	0.684	0.954	0.797
2	30	0.955	0.146	0.254	0.750	0.869	0.805	0.730	0.941	0.823	0.680	0.957	0.795
2	50	0.927	0.088	0.160	0.767	0.855	0.809	0.710	0.950	0.812	0.662	0.959	0.783
2	100	0.889	0.055	0.104	0.778	0.807	0.792	0.702	0.935	0.802	0.616	0.985	0.758
2	250	0.893	0.086	0.157	0.720	0.862	0.785	0.657	0.957	0.779	0.517	0.991	0.679
3	10	0.896	0.312	0.462	0.737	0.749	0.743	0.741	0.769	0.755	0.686	0.948	0.796
3	15	0.942	0.253	0.399	0.740	0.778	0.758	0.749	0.823	0.784	0.686	0.950	0.797
3	20	0.951	0.232	0.373	0.744	0.809	0.775	0.759	0.936	0.838	0.706	0.954	0.811
3	25	0.906	0.217	0.350	0.750	0.874	0.808	0.751	0.940	0.835	0.684	0.952	0.796
3	30	0.897	0.194	0.320	0.751	0.878	0.810	0.724	0.941	0.818	0.680	0.955	0.795
3	50	0.949	0.129	0.227	0.764	0.867	0.812	0.705	0.954	0.811	0.658	0.969	0.784
3	100	0.931	0.093	0.169	0.771	0.833	0.801	0.685	0.957	0.798	0.615	0.990	0.759
3	250	0.878	0.136	0.235	0.713	0.885	0.790	0.631	0.967	0.764	0.512	0.993	0.676
4	10	0.891	0.324	0.475	0.734	0.735	0.734	0.737	0.752	0.744	0.661	0.830	0.736
4	15	0.918	0.231	0.369	0.738	0.769	0.753	0.744	0.800	0.771	0.661	0.840	0.740
4	20	0.931	0.210	0.343	0.738	0.788	0.762	0.733	0.819	0.774	0.671	0.854	0.752
4	25	0.890	0.194	0.319	0.731	0.792	0.760	0.722	0.823	0.769	0.656	0.855	0.742
4	30	0.877	0.172	0.288	0.729	0.793	0.760	0.686	0.828	0.750	0.646	0.857	0.737
4	50	0.930	0.114	0.202	0.732	0.776	0.754	0.669	0.854	0.750	0.619	0.866	0.722
4	100	0.912	0.107	0.191	0.723	0.747	0.735	0.640	0.852	0.731	0.581	0.881	0.700
4	250	0.842	0.138	0.237	0.666	0.785	0.720	0.584	0.874	0.700	0.468	0.919	0.621

Table 6.3: Results for Data Set I - with stemming but with no stopword removal

		Threshold 0.5			Threshold 0.15			Threshold 0.10			Threshold 0.05		
j	k	Pre	Rec	F-Mea	Pre	Rec	F-Mea	Pre	Rec	F-Mea	Pre	Rec	F-Mea
1	10	0.908	0.270	0.416	0.662	0.754	0.705	0.645	0.768	0.701	0.591	0.947	0.728
1	15	0.955	0.217	0.353	0.654	0.773	0.708	0.645	0.816	0.720	0.587	0.950	0.726
1	20	0.979	0.164	0.280	0.653	0.790	0.715	0.633	0.830	0.718	0.604	0.948	0.738
1	25	0.968	0.105	0.189	0.647	0.799	0.715	0.630	0.881	0.735	0.575	0.952	0.717
1	30	0.962	0.086	0.158	0.661	0.800	0.724	0.628	0.928	0.749	0.551	0.957	0.699
1	50	0.900	0.031	0.060	0.688	0.788	0.735	0.609	0.943	0.740	0.525	0.967	0.680
1	100	0.833	0.017	0.034	0.700	0.714	0.707	0.626	0.923	0.746	0.490	0.988	0.655
1	250	0.769	0.017	0.034	0.673	0.795	0.729	0.570	0.945	0.711	0.407	0.993	0.578
2	10	0.913	0.270	0.417	0.735	0.754	0.744	0.706	0.771	0.737	0.619	0.948	0.749
2	15	0.954	0.213	0.349	0.717	0.776	0.745	0.700	0.819	0.755	0.611	0.952	0.744
2	20	0.962	0.176	0.297	0.713	0.792	0.750	0.682	0.835	0.751	0.640	0.952	0.765
2	25	0.958	0.119	0.211	0.699	0.799	0.745	0.680	0.883	0.769	0.595	0.952	0.732
2	30	0.955	0.110	0.198	0.708	0.804	0.753	0.672	0.933	0.781	0.584	0.959	0.726
2	50	0.920	0.040	0.076	0.721	0.793	0.756	0.637	0.950	0.762	0.547	0.969	0.699
2	100	0.842	0.028	0.053	0.733	0.737	0.735	0.653	0.928	0.767	0.507	0.991	0.671
2	250	0.826	0.033	0.063	0.691	0.823	0.751	0.585	0.952	0.725	0.411	0.995	0.581
3	10	0.906	0.267	0.412	0.736	0.745	0.741	0.741	0.769	0.755	0.687	0.947	0.796
3	15	0.958	0.238	0.381	0.742	0.778	0.760	0.748	0.816	0.780	0.682	0.952	0.795
3	20	0.954	0.213	0.349	0.743	0.802	0.772	0.756	0.928	**0.833**	0.705	0.952	0.810
3	25	0.950	0.165	0.282	0.747	0.854	0.797	0.746	0.936	0.831	0.679	0.955	0.794
3	30	0.955	0.146	0.254	0.748	0.857	0.799	0.725	0.941	0.819	0.665	0.962	0.787
3	50	0.926	0.086	0.157	0.764	0.831	0.796	0.704	0.948	0.808	0.643	0.966	0.772
3	100	0.913	0.072	0.134	0.780	0.799	0.789	0.693	0.952	0.802	0.585	0.988	0.734
3	250	0.881	0.102	0.182	0.708	0.861	0.777	0.615	0.964	0.751	0.445	0.993	0.614
4	10	0.904	0.306	0.458	0.734	0.735	0.734	0.737	0.752	0.744	0.658	0.830	0.734
4	15	0.935	0.224	0.361	0.738	0.768	0.753	0.744	0.799	0.770	0.655	0.831	0.733
4	20	0.944	0.203	0.334	0.739	0.785	0.761	0.732	0.814	0.771	0.667	0.845	0.746
4	25	0.946	0.181	0.303	0.733	0.788	0.760	0.716	0.821	0.765	0.652	0.854	0.739
4	30	0.949	0.160	0.274	0.733	0.793	0.762	0.682	0.828	0.748	0.642	0.857	0.734
4	50	0.939	0.107	0.192	0.735	0.756	0.745	0.665	0.845	0.745	0.619	0.862	0.720
4	100	0.915	0.093	0.169	0.721	0.726	0.724	0.637	0.842	0.725	0.567	0.878	0.689
4	250	0.841	0.127	0.221	0.658	0.775	0.711	0.566	0.866	0.685	0.435	0.919	0.590

6.2.2.4 Scenario D - YES Stemming and YES Stopword Removal

In this section, we consider the several scenarios that k (size of the fingerprint) and j (minimum size of the tweet features) take when considering the both stemming and stopword removal are put into practice. Please consider Table 6.4.

As the top-k parameter increases, precision decreases while recall increases. With treshold= 0.10, the best case scenario happens when f-measure= 0.841, for $j = 1$ and $k = 25$. Overall values seem to be identical to those in Table 6.2, which keeps suspicion high regarding stemming providing no improvement.

6.2.3 Results Analysis

In order to fully comprehend the results, a "point of view" approach will be used. In other words, we will make sense of those values from different pertinent perspectives that will help measure how each of steps taken influence the results.

6.2.3.1 Best T2S2 Value

Let us take a look at the Table 6.5, with all the best performance values taken from scenarios A, B, C and D. Remember that j represents the minimum sized word kept by the algorithm and that k is the maximum size of the fingerprint.

Looking at Table 6.5, it is clear that the best performance comes when threshold= 0.10, thus demolishing the notion that 0.5 would be a fair value.

Such a low threshold means that T2S2 score values are typically small. Since our formula comes from Eq. (5.5), there is a direct relation between the number of words that match from the tweet onto the fingerprint and the T2S2 score itself. Please consider Figure 6.1 taken from the above scenario B (stopword removal but no stemming).

The vertical axis contains the number of tweets with $T2S2 >= 0.10$ which the algorithm deemed to belong to a certain trend ("True Positives" and "False Positives"), while the horizontal axis has the ratio between the number of tweet words that match the top-k list and the number of total words in the tweet. In other words, when half of the words in the tweet belong to the top-k word list of a given trend (horizontal axis= 0.5), 2500 classifications with $T2S2 >= 0.10$ have been given.

It is important to remember that, even if typically half of the words belong to a trends top-k list, it does not mean those are high valued by the membership function. This further explains why T2S2 values are so low.

Further analysis into Figure 6.1 shows that there are very few perfect match cases ($T2S2 = 1$),

Table 6.4: Results for Data Set I - with stemming and with stopword removal

		Threshold 0.5			Threshold 0.15			Threshold 0.10			Threshold 0.05		
j	k	Pre	Rec	F-Mea	Pre	Rec	F-Mea	Pre	Rec	F-Mea	Pre	Rec	F-Mea
1	10	0.919	0.272	0.420	0.737	0.747	0.742	0.740	0.769	0.754	0.713	0.947	0.814
1	15	0.951	0.236	0.378	0.736	0.769	0.753	0.747	0.819	0.782	0.712	0.952	0.814
1	20	0.959	0.203	0.335	0.741	0.799	0.769	0.767	0.928	0.840	0.731	0.950	0.826
1	25	0.962	0.172	0.292	0.753	0.854	0.800	0.760	0.940	**0.841**	0.714	0.950	0.815
1	30	0.951	0.134	0.235	0.752	0.859	0.802	0.744	0.941	0.831	0.680	0.957	0.795
1	50	0.927	0.065	0.122	0.771	0.840	0.804	0.723	0.950	0.821	0.662	0.959	0.783
1	100	0.818	0.031	0.060	0.787	0.787	0.787	0.711	0.924	0.804	0.619	0.986	0.761
1	250	0.892	0.057	0.107	0.727	0.852	0.784	0.666	0.950	0.783	0.523	0.993	0.685
2	10	0.912	0.284	0.433	0.738	0.747	0.743	0.741	0.769	0.755	0.685	0.947	0.795
2	15	0.952	0.238	0.380	0.740	0.773	0.756	0.750	0.819	0.783	0.684	0.952	0.796
2	20	0.953	0.210	0.344	0.744	0.802	0.772	0.759	0.935	0.838	0.711	0.950	0.814
2	25	0.956	0.188	0.314	0.749	0.864	0.803	0.755	0.940	0.837	0.684	0.954	0.797
2	30	0.955	0.146	0.254	0.750	0.869	0.805	0.730	0.941	0.823	0.680	0.957	0.795
2	50	0.927	0.088	0.160	0.767	0.855	0.809	0.710	0.950	0.812	0.662	0.959	0.783
2	100	0.889	0.055	0.104	0.778	0.807	0.792	0.702	0.935	0.802	0.616	0.985	0.758
2	250	0.893	0.086	0.157	0.720	0.862	0.785	0.657	0.957	0.779	0.517	0.991	0.679
3	10	0.896	0.312	0.462	0.737	0.749	0.743	0.741	0.769	0.755	0.686	0.948	0.796
3	15	0.942	0.253	0.399	0.740	0.778	0.758	0.749	0.823	0.784	0.686	0.950	0.797
3	20	0.951	0.232	0.373	0.744	0.809	0.775	0.759	0.936	0.838	0.706	0.954	0.811
3	25	0.906	0.217	0.350	0.750	0.874	0.808	0.751	0.940	0.835	0.684	0.952	0.796
3	30	0.897	0.194	0.320	0.751	0.878	0.810	0.724	0.941	0.818	0.680	0.955	0.795
3	50	0.949	0.129	0.227	0.764	0.867	0.812	0.705	0.954	0.811	0.658	0.969	0.784
3	100	0.931	0.093	0.169	0.771	0.833	0.801	0.685	0.957	0.798	0.615	0.990	0.759
3	250	0.878	0.136	0.235	0.713	0.885	0.790	0.631	0.967	0.764	0.512	0.993	0.676
4	10	0.891	0.324	0.475	0.734	0.735	0.734	0.737	0.752	0.744	0.661	0.830	0.736
4	15	0.918	0.231	0.369	0.738	0.769	0.753	0.744	0.800	0.771	0.661	0.840	0.740
4	20	0.931	0.210	0.343	0.738	0.788	0.762	0.733	0.819	0.774	0.671	0.854	0.752
4	25	0.890	0.194	0.319	0.731	0.792	0.760	0.722	0.823	0.769	0.656	0.855	0.742
4	30	0.877	0.172	0.288	0.729	0.793	0.760	0.686	0.828	0.750	0.646	0.857	0.737
4	50	0.930	0.114	0.202	0.732	0.776	0.754	0.669	0.854	0.750	0.619	0.866	0.722
4	100	0.912	0.107	0.191	0.723	0.747	0.735	0.640	0.852	0.731	0.581	0.881	0.700
4	250	0.842	0.138	0.237	0.666	0.785	0.720	0.584	0.874	0.700	0.468	0.919	0.621

Table 6.5: Summary of the Best Case Scenarios

Scenario	Threshold	j	k	Pre	Rec	F-Mea
A	0.10	3	20	0.756	0.928	0.833
B	0.10	1	25	0.760	0.940	0.841
C	0.10	3	20	0.756	0.928	0.833
D	0.10	1	25	0.760	0.940	0.841

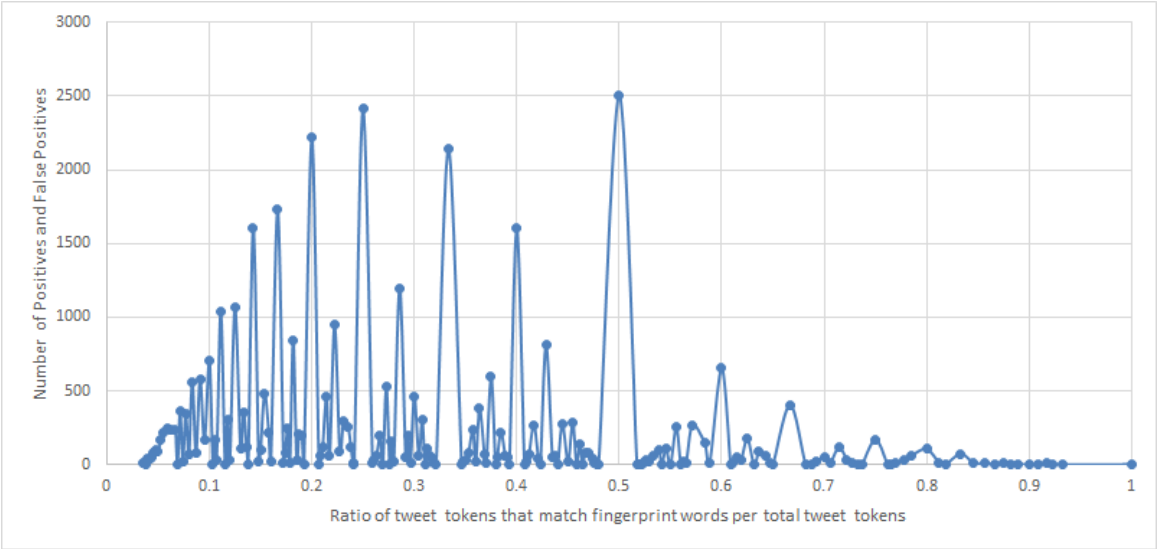

Figure 6.1: Num. of tweets with $T2S2 >= 0.10$ per % of tweet matching words with fingerprint

i.e., all the words in the tweet match the top words from the trend top-k list. In addition, very few tweets with less than 10% of matching words are considered to belong to any trend.
In conclusion, with the Twitter Topic Fuzzy Fingerprint, best performance comes when a threshold value is low, typically 0.10.

6.2.3.2 Importance of Fingerprint Size

One of the parameters under scrutiny by this thesis, is the impact of the k value in the top-k approach to building a fingerprint. As seen in chapter 5, k equals the maximum number of keywords from which to build the fingerprint with, based on the training data set.
In this project, k has a minimum value of 10 and a maximum value of 250. Looking at Figure 6.2, built from Scenario B, it is immediate apparent that for k= $[10, 15, 20, 25]$ as k increases, so does precision,recall and f-measure. It peaks at $k = 25$ where precision starts to decline while recall continues to increase, with the exception of $k = 100$ where it has a small decrease.
Precision seems to have an higher impact on f-measure since it follows its trend to decline for $k > 25$ despite an increasing recall. This suggests that the number of False Positives will increase for higher values of k.

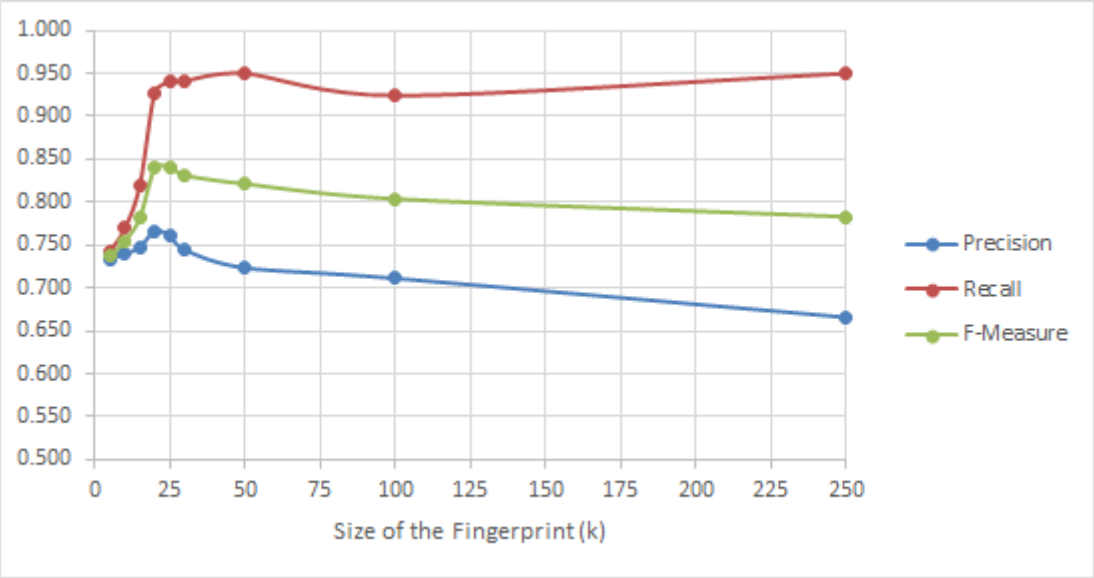

Figure 6.2: Impact of Fingerprint Size on performance

By analyzing f-measure, it can be stated that for a low value of $k = 25$, under the right parameter configuration, optimal results are provided (f-measure= 0.841). This is particularly interesting given Twitter's nature of limiting content to 140 characters, which, as a consequence, limits the number of words.

Such a low value of k implies that each topic has very important terms that will feature high on the top-k list and that become essential in helping our method perform better.

6.2.3.3 Importance of Minimum Sized Words

Let us now take a look at how the removal of short words impacts this algorithm. Please consider Scenario A, where neither stemming nor stopword removal was executed. The threshold value is 0.10 and $k = 20$.

In Figure 6.3, precision, recall and f-measure reach their maximum value when the minimum sized word kept is at least 3 characters long. This suggests that there is a fine balance to be maintained between removing no words and removing too many words based on their length.

For a more detailed look at the values in Figure 6.3, please consult the Table 6.6.

Despite Table 6.6, the Twitter Topic Fuzzy Fingerprint algorithm performs better when stopwords are removed (Scenario B). For $j = 1$ and $k = 25$, f-measure= 0.841 as opposed to Scenario A's performance ($j = 3$, $k = 20$, f-measure= 0.833). In other words, if stopwords are removed, it will not be necessary to remove short words. A possible explanation to this phenomenon is that the short words to be removed are typically stopwords, which will result in a similar effect. However, some stopwords have more than 2 characters ($j >= 3$) which means that it will not

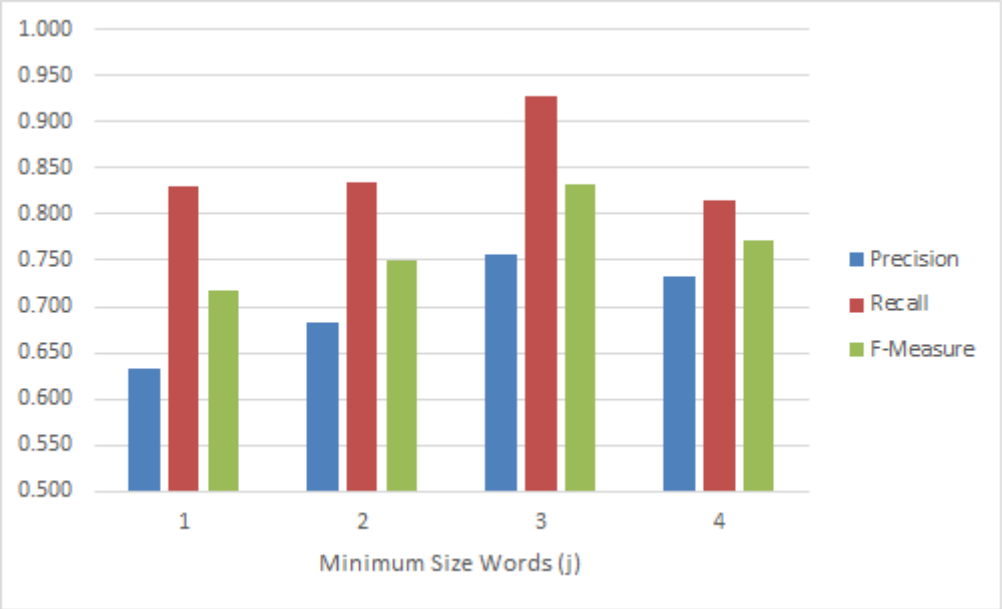

Figure 6.3: Impact on performance as the minimum size of words kept is varied

Table 6.6: Numerical analysis of the importance of removing short words

		Original Memb. Func.		
j	k	Pre	Rec	F-Mea
1	20	0.633	0.830	0.718
2	20	0.682	0.835	0.751
3	20	0.756	0.928	0.833
4	20	0.732	0.814	0.771

provide the full force of stopword removal, therefore not reaching the same f-measure value. In conclusion, best case scenario in terms of the minimum sized words to be kept is usually reached when all words have at least 3 characters, as long as stopwords are not removed.

6.2.3.4 Importance of Removal of Stopwords

In order to determine the importance the preprocessing technique of removing stopwords, we will compare all the performance metrics across scenarios A,B,C and D. Please remember the rules of each scenario:

- Scenario A: Without Stemming and Without Stopword Removal
- Scenario B: Without Stemming and With Stopword Removal
- Scenario C: With Stemming and Without Stopword Removal
- Scenario D: With Stemming and With Stopword Removal

Table 6.7 shows how our method behaves for the previously agreed upon best case scenario of threshold= 0.10 and minimum size words $j = 1$.

Table 6.7: Numerical analysis of the importance of removing stopwords and stemming

		Scenario A			Scenario B			Scenario C			Scenario D		
j	k	Pre	Rec	F-Mea	Pre	Rec	F-Mea	Pre	Rec	F-Mea	Pre	Rec	F-Mea
1	5	0.654	0.756	0.701	0.733	0.742	0.737	0.654	0.756	0.701	0.733	0.742	0.737
1	10	0.645	0.768	0.701	0.740	0.769	0.754	0.645	0.768	0.701	0.740	0.769	0.754
1	15	0.645	0.816	0.720	0.747	0.819	0.782	0.645	0.816	0.720	0.747	0.819	0.782
1	20	0.633	0.830	0.718	0.767	0.928	0.840	0.633	0.830	0.718	0.767	0.928	0.840
1	25	0.630	0.881	0.735	0.760	0.940	0.841	0.630	0.881	0.735	0.760	0.940	0.841
1	30	0.628	0.928	0.749	0.744	0.941	0.831	0.628	0.928	0.749	0.744	0.941	0.831
1	50	0.609	0.943	0.740	0.723	0.950	0.821	0.609	0.943	0.740	0.723	0.950	0.821
1	100	0.626	0.923	0.746	0.711	0.924	0.804	0.626	0.923	0.746	0.711	0.924	0.804
1	250	0.570	0.945	0.711	0.666	0.950	0.783	0.570	0.945	0.711	0.666	0.950	0.783

Figure 6.4 shows the evolution of precision, recall and f-measure per scenario for top-$k = 25$.

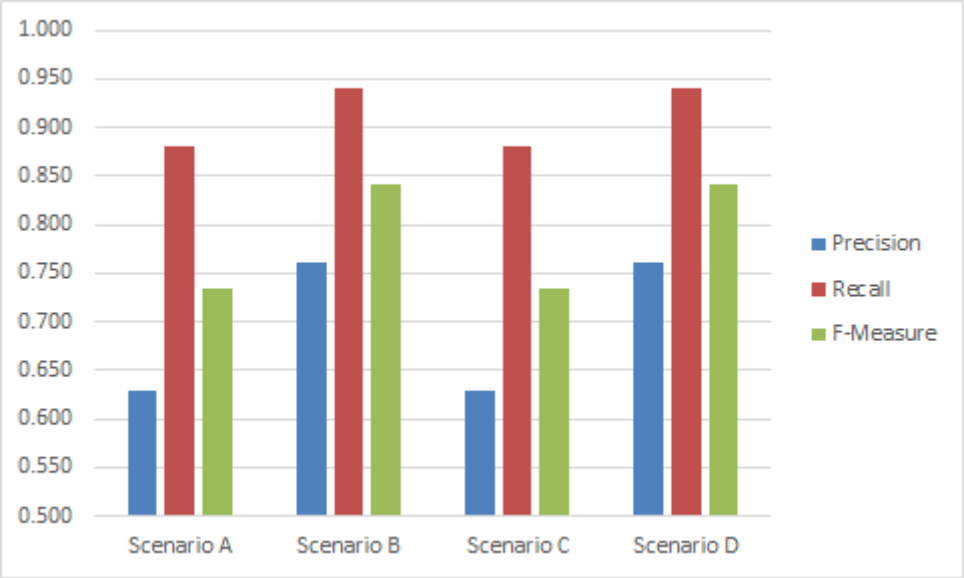

Figure 6.4: Impact of stopwords removal in performance

When analyzing both Table 6.7 and Figure 6.4, it becomes clear that Scenarios B and D, where stopwords are removed, perform better in terms of precision, recall and f-measure.

In conclusion, the removal of stopwords will provide an improved and more precise algorithm for this data set of tweets.

6.2.3.5 Importance of Stemming

Further analysis into Table 6.7 and Figure 6.4, will also reveal how stemming impacts our method: it does not.

Comparing Scenario A with Scenario C, where stopwords removal was not done, the values are exactly the same, despite the fact that stemming was activated for latter case. The same is true for the comparison between Scenarios B and D, where stopword removal was executed.

A possible explanation for this, may derive from the nature of language in Twitter itself. Since tweets are short in nature, words may often occur in their stem form or in such form that a formal stemmer cannot work. Porter's Stemmer cannot process words such as "luv","lol","omg",etc... This is a lingo very unique to the informal nature of social networking communication, while a Stemming algorithm can only truly be effective with formal and well written texts.

In conclusion, Stemming, as a preprocessing technique, does not improve nor damage our method's performance. However, since it is a time and resource consuming task, it is preferable to not stem.

6.2.3.6 Execution Efficiency

So far, we have looked at the Twitter Topic Fuzzy Fingerprint method from a performance point of view only. However, as it was mentioned throughout this document, due to the immediate nature of Twitter and amount of data it processes, time is also an important parameter.

Please consider Table 6.8, where j represents the minimum sized word to be kept by our method, and k is the number of words of the fingerprint. All values are represented in seconds and tests were performed on a laptop with Intel i7-3517U processor, 8GB RAM 1600Mhz and a SSD (solid state drive).

Table 6.8: Execution Speed

Scenario	j	k	Preprocessing Build Model	Evaluate Test Set	Total
A	3	20	2.030 s	0.507 s	2.537 s
B	1	25	49.900 s	0.643 s	50.543 s
C	3	20	24.385 s	0.615 s	25.000 s
D	1	25	81.558 s	0.506 s	82.064 s

We considered the best performance for each Scenario, as shown by Table 6.5. Table 6.8 shows that the evaluation of our 585 test tweets is quite fast, regardless of whether stopwords or stem words were used. It will always take an average of 0.6 seconds, regardless of the scenario considered.

However, the use of stopword removal and/or stemming does impact on how long it takes to

pre-process the tweets and build the data model. Stemming on its own (Scenario C) takes almost 11 times longer to perform the first step, while stopword removal (Scenario B) takes 24 times longer. In Scenario D, where both techniques are in play, it takes almost 40 times longer.
By comparing the total time taken with Scenarios B and C, it is shown that the removal of stopwords has a bigger impact on computational performance than stemming.
While Table 6.5 shows that removing stopwords provides better results (f-measure= 0.841), the fact of the matter is that without removing stopwords, the Twitter Topic Fuzzy Fingerprint is much faster and almost equally effective (f-measure= 0.833). Henceforth, the latter will be called "Most Effective Case Scenario" while the the first will be the simply be called "Best Case Scenario".

6.2.4 Most Effective Case Scenario

With all of the above sections taken under consideration, it is fair to sum up things as follows. The Twitter Topic Fuzzy Fingerprint Algorithm for Topic Detection within Twitter, achieves optimal performance and execution when:

- considering a low threshold value for acceptance of a twitter belonging to a trend (0.10)
- configuring a low value of k for the size of the list of the fingerprint ($k = 20$)
- removing short words from the corpus, only keeping words with a minimum length of 3 characters
- not removing stopwords from the corpus
- not performing stemming operations

In other words, Scenario A, for $j = 3$, $k = 20$ and threshold=0.10.

6.2.5 Comparison to Other Methods

It is fair to state that our method performs quite well, so far. But how does it fare against other methods that are known for good performance in Topic Detection?
In this section, the Twitter Topic Fuzzy Fingerprint will be put up against the two methods we presented documented in the "Related Work" chapter: k-Nearest Neighbor (kNN) and Support Vector Machine (SVM). The exact same training data sets and test data sets were used.
For this purpose of this comparison, we will first see how the configurations of the parameters influence kNN and SVM. Secondly, we will compare them to our algorithm in equal parameter values. Finally, we will compare best case scenarios.

The tests were performed by the WEKA tool [28], after both the training and test sets were preprocessed to a bag-of-words representation, Figure 2.1, with $tfidf$ weighting, Eq. (2.3).

6.2.5.1 Twitter Topic Fuzzy FingerPrint vs kNN

As detailed in Chapter 2, kNN is an algorithm heavily influenced by the spacial representation of the tweets (bag-of-words). It places the tweet under analysis, inside that N-dimensional representation of the data set and then inquires the k Nearest Neighbors as to which category/trend they belong to. Using a majority rule, it will then attribute that same category to the tweet. Firstly, let us see how it stands on its own according to the same Scenarios A, B, C and D. Please consider Table 6.9 with the performance results for all scenarios. Different values of k where considered and j represents the minimum sized word kept for each situation. Due to limitations in the use of WEKA software, the distance measure considered was the Euclidean distance as opposed to the cosine similarity presented in Eq. 2.6.

Table 6.9: kNN's performance for Scenarios A,B,C and D

		Scenario A			Scenario B			Scenario C			Scenario D		
j	k	Pre	Rec	F-Mea	Pre	Rec	F-Mea	Pre	Rec	F-Mea	Pre	Rec	F-Mea
1	1	0.479	0.135	0.211	0.470	0.123	0.195	0.479	0.135	0.211	0.470	0.123	0.195
1	3	0.466	0.321	0.380	0.454	0.329	0.382	0.466	0.321	0.380	0.454	0.329	0.382
1	5	0.467	0.280	0.350	0.484	0.309	0.377	0.467	0.280	0.350	0.484	0.309	0.377
1	10	0.332	0.229	0.271	0.384	0.232	0.289	0.332	0.229	0.271	0.384	0.232	0.289
2	1	0.509	0.137	0.216	0.470	0.125	0.197	0.509	0.137	0.216	0.470	0.125	0.197
2	3	0.453	0.324	0.378	0.447	0.323	0.375	0.453	0.324	0.378	0.447	0.323	0.375
2	5	0.422	0.285	0.340	0.455	0.316	0.373	0.422	0.285	0.340	0.455	0.316	0.373
2	10	0.369	0.227	0.281	0.380	0.239	0.293	0.369	0.227	0.281	0.380	0.239	0.293
3	1	0.486	0.137	0.214	0.463	0.125	0.197	0.486	0.137	0.214	0.463	0.125	0.197
3	3	0.448	0.319	0.373	0.453	0.319	0.374	0.448	0.319	0.373	0.453	0.319	0.374
3	5	0.441	0.299	0.356	0.443	0.314	0.368	0.441	0.299	0.356	0.443	0.314	0.368
3	10	0.360	0.247	0.293	0.366	0.237	0.288	0.360	0.247	0.293	0.366	0.237	0.288
4	1	0.634	0.142	0.232	0.613	0.128	0.212	0.634	0.142	0.232	0.613	0.128	0.212
4	3	0.466	0.341	**0.394**	0.461	0.340	0.391	0.466	0.341	**0.394**	0.461	0.340	0.391
4	5	0.470	0.324	0.384	0.458	0.323	0.379	0.470	0.324	0.384	0.458	0.323	0.379
4	10	0.463	0.247	0.322	0.464	0.239	0.315	0.463	0.247	0.322	0.464	0.239	0.315

The results are quite poor, with the best f-measure value peaking at an unimpressive 0.394, when 3 neighbors are consulted and words under 4 characters long are removed from the corpus. A possible explanation for this poor performance is the unbalanced nature of the training data set, as explained by Zhang in [8]. When dealing with unbalanced data sets, kNN completely ignores the minority classes and will often mistakenly classify a tweet to the majority category.

It is interesting to see that impact of stemming remains null. If, as it was done in the previous section, comparing Scenario A with C, one can state that the results are exactly the same. This remains true to the comparison between Scenarios B and D.

In addition, the results tend to be better with stopword removal but only for low values of j. When $j = 4$, kNN performs better for Scenarios A and C (no stopword removal) which seems to dictate that dimensionality reduction is of great importance to improve this method.

When comparing kNN with the Best Case Scenario from Twitter Topic Fuzzy Fingerprints ($j = 1$, $k = 25$, threshold= 0.10 and stopword removal), Table 6.10, it is clear that our algorithm is better:

Table 6.10: kNN's performance Best Case Scenario

Method	Pre	Rec	F-Mea
Twitter Topic Fuzzy Fp ($j = 1$,$k = 25$)	0.760	0.940	0.841
kNN ($j = 4$,$k = 3$)	0.466	0.341	0.394

But let us now compare both methods according to the same configuration, 6.11. In this case, kNN's best case configuration occurs when $j = 4$, no stopwords are removed and stemming is not performed (Scenario A).

Table 6.11: kNN vs Twitter Topic Fuzzy Fingerprint

Method	k	Pre	Rec	F-Mea
kNN	1	0.634	0.142	0.232
	3	0.466	0.341	0.394
	5	0.470	0.324	0.384
	10	0.463	0.247	0.322
Twitter Topic Fuzzy FP	10	0.737	0.752	0.744
	15	0.744	0.799	0.770
	20	0.732	0.814	0.771
	25	0.716	0.821	0.765
	30	0.682	0.828	0.748
	50	0.665	0.845	0.745
	100	0.637	0.842	0.725
	250	0.566	0.866	0.685

Once again, kNN finds itself outperformed. Whether it is precision, recall of f-measure, under the same configuration, the Twitter Topic Fuzzy Fingerprint is just better.

Before departing from this comparison, it is important to remember that, as previously explained, the meaning of k is different between the two methods. In scope of kNN, it represents the number of neighbors used to determine the category/trend that a tweet belongs to. In the

scope of the Twitter Topic Fuzzy Fingerprint, it represents the size of the fingerprint, i.e., the number of keywords the fingerprint list contains.

6.2.5.2 Twitter Topic Fuzzy FingerPrint vs SVM

Much like kNN, SVM is also a vector representation influenced algorithm (bag-of-words). It defines an hyperplane between categories and uses the borderline documents to decide which it belongs to.

Once again, with the help of WEKA, tests were conducted to compare SVM's performance against the Fuzzy Fingerprint. A number of different parameters were tested and optimized, and the best performance was achieved using a linear kernel and a small soft-margin value: $C = 0.01$. Additionally, the parameters to "normalize" and "probability estimates" where activated. This means that, as opposed to a binary output on whether a tweet belonged to a certain topic (either True(+1) of False(+1)), SVM made internal calculations that estimated how probable the tweet was that to belonged to it.

With these parameters in mind, please consult Table 6.12 for SVM's performance. It contains once again, all for Scenarios from the previous testing (A, B, C and D) and j represents the minimum size of the words kept in each scenario.

Table 6.12: SVM's performance for Scenarios A,B,C and D

	Scenario A			Scenario B			Scenario C			Scenario D		
j	Pre	Rec	F-Mea	Pre	Rec	F-Mea	Pre	Rec	F-Mea	Pre	Rec	F-Mea
1	0.786	0.700	0.741	0.858	0.652	0.741	0.786	0.700	0.741	0.858	0.652	0.741
2	0.773	0.691	0.730	0.863	0.712	**0.780**	0.773	0.691	0.730	0.863	0.712	**0.780**
3	0.807	0.683	0.740	0.872	0.674	0.760	0.807	0.683	0.740	0.872	0.674	0.760
4	0.787	0.589	0.674	0.836	0.625	0.715	0.787	0.589	0.674	0.836	0.625	0.715

Overall, the performance is far better than kNN's. For Scenarios B and D, where stopwords were removed, when $j = 2$ it scores a f-measure=0.780. However, it does not match the Twitter Topic Fuzzy Fingerprints Best Case Scenario where f-measure= 0.841 for $j = 1$, $k = 25$, threshold=0.10 and stopwords are removed, as shown by Table 6.13.

Table 6.13: SVM performance Best Case Scenario

Method	Pre	Rec	F-Mea
Twitter Topic Fuzzy FP ($j = 1$,$k = 25$)	0.760	0.940	0.841
SVM ($j = 2$)	0.863	0.712	0.780

Despite being more precise, SVM falls short in terms of recall, which explains the difference in f-measure. This SVM performance was only achieved when using a normalized $tfidf$ approach

to the document representation, Eq. (2.4). During our testing, if using the most common $tfidf$ variant, Eq. (2.3), the results would deteriorate immensely as SVM would fall into the same trap as kNN and classify all samples to the majority category/topic.

Let us now compare both methods according to the same configuration, 6.14. In this case, SVM's best case configuration occurs when $j = 2$, stopwords are removed and stemming is not performed (Scenario B). In terms of our algorithm, the best value of k is $k = 20$.

Table 6.14: SVM vs Twitter Topic Fuzzy Fingerprint

Method	Pre	Rec	F-Mea
SVM ($j = 2$)	0.863	0.712	0.780
Twitter Topic Fuzzy FP ($j = 2$,$k = 20$)	0.759	0.935	0.838

The conclusion remains the same. SVM is more precise but the Twitter Topic Fuzzy Fingerprint method excels in terms of recall which provides a better f-measure.

6.2.5.3 Overall Comparison

When comparing all 3 methods, it is evident that the Twitter Topic Fuzzy Fingerprint algorithm outperforms both kNN and SVM. The proposed method achieves the best values for precision, recall and f-measure. The SVM strategy, while being more precise achieves almost 32% lower recall. The results achieved by kNN are definitely worse than the other two methods.

The reason why SVM performs outperforms kNN is because it can be better fine tuned than the latter. As mentioned above, this performance is the result of a linear kernel function input with a soft margin $C = 0.01$. Additional testing showed that non-linear kernels displayed the same kind of results as kNN, exactly for the same reason of being unable to handle an unbalanced data set.

Table 6.15: Overall Comparison

Method	Precision	Recall	F-Measure
Twitter Topic Fuzzy FP ($j = 1$,$k = 25$)	0.760	0.940	0.841
kNN ($j = 4$,$k = 3$)	0.466	0.341	0.394
SVM ($j = 2$)	0.863	0.712	0.780

6.2.5.4 Execution Efficiency Comparison

In this section, we will inspect how the Twitter Topic Fuzzy Fingerprint fares against SVM and kNN from a time consumed perspective. In this occasion, we will consider the Most Effective Case Scenario from our algorithm, where f-measure= 0.833 when $j = 3$, $k = 20$ and threshold=0.10.

In terms of time taken to build the model and to perform classification, one can not directly compare fuzzy fingerprints with the other two methods because different tool-kits are being used.

In our algorithm, the preprocessing stage, implemented in Python, is a part of reading the tweets and creating the model as the execution progresses. In kNN and SVM, the preprocessing stage consist not only of removing stopwords, but also calculating the TF-IDF and creating an output that is fit for WEKA, often known as the Attribute-Relation File Format (ARFF). For the these two methods, the model itself is only built within the WEKA execution.

Nevertheless, since our Most Effective Case Scenario is without stopword removal and stemming, the preprocessing stage for kNN and SVM consists solely of: reading the tweets, removing all words with less than $j = 3$ characters, calculate $tfidf$ and build the bag-of-words representation in ARFF for WEKA.

Table 6.16 shows the processing time comparison, in seconds, for $j = 3$ when neither stopwords nor stemming is performed.

Table 6.16: Execution Speed Comparison

Method	Preprocessing	Build Model	Evaluate Test Set	Total
Fuzzy Fp	2.030 s		0.507 s	2.537 s
kNN	416.441 s	0.070 s	32.730 s	449.241 s
SVM	416.441 s	642.830 s	4.170 s	1063.441 s

Table 6.16 clearly shows how much faster the Twitter Topic Fuzzy Fingerprint algorithm is. It is 177 times faster than kNN and 419 times faster than SVM.

Even if you consider that the disparity in the preprocessing method may be harmful to an unbiased comparison, it still evaluates the data set 64 times faster than kNN and 8 times faster than SVM.

These results can be explained by the different nature of the algorithms. kNN is a lazy algorithm, where all computation is deferred until classification, which means that the reported time relates to preprocessing and loading time. In what concerns to SVM, a significant part of the time is attributed to creating the model. In our approach, building the model is just a linear function of the number of words being considered for training.

Finally, the size of model is also a significant issue, specially when one aims at processing big quantities of data, in a distributed fashion. The fuzzy fingerprint model for a given topic corresponds to a vector of k fixed elements, each one containing the value of a feature (e.g. the word) and its score. Therefore it is fixed in size and corresponds to pruning the list of relevant words at k.

6.3 Data Set II

The second data set used in this thesis is composed exclusively of Portuguese tweets. It was used to study the behavior of the Twitter Topic Fuzzy Fingerprints method with an increasing number of fingerprints in detecting fewer topics.

6.3.1 Training Set

Using the same method as with Data Set I, we obtained just over 1.2 million Portuguese tweets, from for March, 14th to March 20th, 2014. We extracted the top trends on the 17th of March and among them, we selected two topics that seemed to have the most interesting content:

- #AnittaNarizDeCapivara, regarding Anitta's (Brazilian singer) new nose job which made an impact during the annual award show "Melhores do Ano";
- #FicaVanessa, for people supporting Big Brother's Brazil participant Vanessa who was at risk of leaving the show;

Despite being top trends, we found that these hashtags only occurred 289 and 822 times in our whole set of 1.2 million tweets.
Even though we only used 2 topics for testing purposes (due to the difficulty in annotating a higher number of topics), the training set was built using 100 different topics created after the most popular hashtags on the database. The training set is composed of over 600000 tweets in Portuguese language, where the most popular trend is #kca (18000 tweets) and the rarer is #1dnamix (139 tweets).

6.3.2 Experiments and Results

The test set was impartially built from uncategorized tweets belonging to the original set of 1.2 million documents.
Because of the TV broadcasting nature of the two target trends (#AnittaNarizDeCapivara and #FicaVanessa), where the owners of the trends encourage its use and propagation, it was very difficult to find many uncategorized tweets that clearly belonged to those topics. Only 82 and 43 respectively were annotated.
In order to increase the size of test set, a few more uncategorized tweets were added to it, making for a total of 210 samples. Whilst still short, it provides a chance for our algorithm to detect negative scenarios efficiently, since the added tweets belong to untrained top trends.
Through extensive testing, we found that the best results for the Twitter Topic Fuzzy Fingerprints Algorithm were achieved when:

- considering a low threshold value for acceptance of a twitter belonging to a topic (T2S2 = 0.10);
- configuring a value of $k = 40$ for the size of the list of the fingerprint;
- removing short words from the corpus, only keeping words with a minimum length of 4 characters($j = 4$);
- removing stopwords from the corpus;
- not performing Stemming operations;

6.3.3 Results Analysis

Table 6.17 summarizes the algorithm's performance. Regardless of the value of k, precision values are always high, which indicates that False Positive scenarios are rare or non-existent. However, recall is low for small values of $k = [10; 15]$ peaking at $k = 40$, when no False Negative scenarios are identified.

Table 6.17: Twitter Topic Fuzzy Fingerprint Performance

j	k	Precision	Recall	F-Measure
4	10	1.000	0.621	0.766
4	15	1.000	0.694	0.819
4	20	1.000	0.815	0.898
4	25	1.000	0.952	0.975
4	30	1.000	0.976	0.988
4	40	0.992	1.000	**0.996**
4	50	0.992	1.000	0.996
4	75	1.000	0.976	0.988
4	100	1.000	0.968	0.984
4	150	1.000	0.968	0.984
4	250	1.000	0.976	0.988
4	500	1.000	0.976	0.988

A low recall is also a consequence of typically small T2S2 similarity values, which means that Positive cases are not being identified because they were below the threshold= 0.10. Consider the example of a preprocessed tweet with 8 features, two of which match a given fingerprint: even if the matching words are the highest ranked membership terms, T2S2 would score approximately under $\frac{2}{8} = 0.25$.

As k increases, either more matching words between the tweet features and the fingerprint are found, or the same ranked words have a higher membership value which encourages a better

T2S2 score.

The best case scenario scores f-measure= 0.996, which, while extremely positive, is suspicious due to the fact that the data set is short and possibly over trained. In Data Set I, the Twitter Fuzzy Fingerprints method reached f-measure= 0.841 for a larger multi-language data set and with more target top trends.

Here we tested for 2 target topics out of the possible 100 training trends. The purpose behind this approach was to study how the Twitter Fuzzy Fingerprints method would behave when approaching a larger and more realistic number of different possible trends, and to study the impact of using the Inverse Topic Frequency (ITF) which should theoretically improve the results with the increase in the number of topics.

6.3.4 kNN and SVM Performance

In order to test kNN and SVM, stopwords were removed but stemming was not performed. The final representation of either training and test set is a bag-of-words type, Figure 2.1, with TF-IDF weighting, Eq. (2.3). The tests were performed using the WEKA framework.

kNN was executed with the following k values: [3,5,10,30,50]. In either of those scenarios, the algorithm was incapable of identifying a single True Positive case, i.e., precision= 0, recall= 0, f-measure= 0. Instead, it classified all the samples as a part of the majority trained class #kca, which is non-existent in the test set.

For SVM, a number of different parameters were tested and optimized. Even with the same configuration as that used with Data Set I (linear kernel and a small soft-margin value: $C = 0.01$), the results fell short. For our test set of 210 Portuguese tweets, SVM behaved exactly as kNN did, i.e., precision= 0, recall= 0, f-measure= 0.

There are two possible explanations for such a poor performance. The first is the fact that there were so many different classes for either method to train. In addition, the bag-of-words representation of the test data set was a very sparse matrix, with 210 documents (lines) and over 69000 unique features (columns). This would make these space-vector oriented algorithms highly ineffective.

The second explanation is, once again, the unbalanced nature of the training data set. When dealing with unbalanced data sets, kNN completely ignores the minority classes and will often mistakenly classify a tweet to the majority category.

6.4 Data Set III

The final data set used in this thesis is taken from TASS - the Workshop on Sentiment Analysis at SEPLN. TASS is a experimental evaluation workshop for sentiment analysis and online reputation analysis focused on the Spanish language, organized as a satellite event of the annual SEPLN Conference.

The corpus contains over 70000 tweets, written in Spanish by nearly 200 well-known personalities and celebrities of the world of politics, economy, communication, mass media and culture, between November 2011 and March 2012.

Using the 2012 twitter data set of this competition, the Twitter Topic Fuzzy FingerPrint Algorithm will be put to the test to see how it fares in classifying tweets into broad topics such as "política" ("politics"), "fútbol" ("soccer"), "literatura" ("literature") or "entretenimiento" ("entertainment"). In terms of the difference between Topic Classification and Topic Detection, which was established in Chapter 2, this data set falls into the first category. Therefore, we are attempting to see how this method adapts when tackling a related but yet intrinsically different kind of problem

Each message of the corpus has been semi automatically assigned to one or several of these topics (most messages are associated to just one topic, due to the short length of the text).

In the context of this work, we use the training set (file "general.train.xml") to build the fingerprint. Afterwards, we apply the method to the tweets in file "general.devel.xml" which are categorized, so we can match its performance.

6.4.1 Training Set

The training set consists of 5775 tweets, in XML format, as shown in Figure 6.5. For the purpose of this thesis, we will ignore the "sentiment" field and focus solely on user, content and topics.

6.4.2 Experiments and Results

The test data set consists of 1444 tweets, using the same format as Figure 6.5.

6.4.2.1 Scenario A - NO Stemming and NO Stopword Removal

In this section, we consider the several scenarios that k (size of the fingerprint) and j (minimum size of the tweet features) take neither stopword removal nor stemming is used. Please consider Table 6.18.

```
<tweet>
 <tweetid>142499355360903168</tweetid>
 <user>sevillajordi</user>
 <content><![CDATA[Un sistema económico q recorta dinero para
  prestaciones sociales y refuerza con billón
  y medio d euros a los bancos, no necesita
  repensarse?]]></content>
 <date>2011-12-02T08:04:28</date>
 <lang>es</lang>
 <sentiments>
  <polarity><value>P+</value><type>AGREEMENT</type></polarity>
 </sentiments>
 <topics>
  <topic>economía</topic>
  <topic>política</topic>
 </topics>
</tweet>
```

Figure 6.5: XML format of the Training Set TASS tweets

A quick glance at these results clearly show that they are worse than those obtained with Data Set I. For threshold= 0.5, there were even several "divide by zero" errors, due to inexistent True of False Positive cases. As the threshold value decreases, no more such cases occur. The best case scenario takes place when $j = 4$ and $k = 250$ ($f - measure = 0.401$). This is a consequence of very low T2S2 values.

6.4.2.2 Scenario B - NO Stemming and YES Stopword Removal

In this section, we consider the several scenarios that k (size of the fingerprint) and j (minimum size of the tweet features) stopwords are removed but stemming is not used.

In Scenario B, Table 6.19, performance is slightly better than in Scenario A, which reinforces the importance of stopword removal. The best case scenario occurs for f-measure= 0.404 when $j = 1$ and $k = 250$. However it is still below the performance achieved with Data Set I, due to very small T2S2 values.

Table 6.18: Results for Data Set II - no stemming and no stopword removal

		Threshold 0.5			Threshold 0.15			Threshold 0.10			Threshold 0.05		
1	10	0.429	0.002	0.003	0.642	0.078	0.140	0.642	0.080	0.143	0.586	0.127	0.208
1	15	0.250	0.001	0.001	0.647	0.083	0.147	0.623	0.105	0.180	0.588	0.134	0.219
1	20	0.000	0.000	—	0.641	0.094	0.163	0.623	0.113	0.191	0.586	0.131	0.215
1	25	0.000	0.000	—	0.640	0.093	0.163	0.609	0.117	0.196	0.586	0.138	0.224
1	30	0.000	0.000	—	0.684	0.088	0.156	0.600	0.119	0.199	0.572	0.146	0.233
1	50	—	0.000	—	0.564	0.034	0.063	0.613	0.123	0.204	0.574	0.163	0.254
1	100	—	0.000	—	0.495	0.023	0.045	0.628	0.082	0.146	0.579	0.203	0.300
1	250	—	0.000	—	0.567	0.028	0.053	0.593	0.092	0.160	0.567	0.262	0.359
2	10	0.429	0.002	0.003	0.642	0.078	0.140	0.642	0.080	0.143	0.585	0.127	0.208
2	15	0.250	0.001	0.001	0.650	0.086	0.152	0.623	0.105	0.180	0.590	0.135	0.220
2	20	0.000	0.000	—	0.641	0.094	0.163	0.624	0.113	0.191	0.586	0.132	0.215
2	25	0.000	0.000	—	0.641	0.094	0.163	0.605	0.117	0.196	0.584	0.140	0.226
2	30	0.000	0.000	—	0.669	0.089	0.156	0.597	0.121	0.201	0.574	0.148	0.235
2	50	—	0.000	—	0.600	0.040	0.074	0.605	0.126	0.209	0.573	0.164	0.256
2	100	—	0.000	—	0.490	0.026	0.049	0.638	0.088	0.155	0.580	0.207	0.305
2	250	—	0.000	—	0.591	0.033	0.063	0.580	0.096	0.165	0.571	0.280	0.376
3	10	0.500	0.002	0.004	0.642	0.078	0.140	0.641	0.081	0.144	0.578	0.130	0.213
3	15	0.400	0.001	0.002	0.640	0.090	0.157	0.623	0.105	0.180	0.578	0.138	0.223
3	20	0.000	0.000	—	0.638	0.098	0.170	0.623	0.115	0.194	0.589	0.138	0.224
3	25	0.000	0.000	—	0.642	0.101	0.175	0.589	0.120	0.199	0.577	0.146	0.233
3	30	0.000	0.000	—	0.642	0.099	0.171	0.589	0.123	0.204	0.574	0.151	0.239
3	50	—	0.000	—	0.629	0.063	0.115	0.581	0.134	0.218	0.568	0.169	0.260
3	100	—	0.000	—	0.586	0.048	0.089	0.602	0.119	0.198	0.558	0.221	0.317
3	250	—	0.000	—	0.586	0.066	0.118	0.582	0.138	0.223	0.548	0.308	0.394
4	10	0.500	0.002	0.004	0.642	0.078	0.140	0.640	0.081	0.145	0.581	0.131	0.214
4	15	0.500	0.002	0.003	0.640	0.095	0.166	0.622	0.105	0.180	0.579	0.142	0.227
4	20	0.000	0.000	—	0.632	0.101	0.175	0.620	0.116	0.196	0.587	0.143	0.230
4	25	0.000	0.000	—	0.634	0.105	0.180	0.597	0.120	0.200	0.582	0.146	0.234
4	30	0.000	0.000	—	0.631	0.105	0.181	0.598	0.123	0.204	0.585	0.151	0.240
4	50	—	0.000	—	0.642	0.083	0.147	0.582	0.137	0.222	0.575	0.172	0.264
4	100	—	0.000	—	0.592	0.062	0.112	0.603	0.144	0.232	0.566	0.229	0.326
4	250	—	0.000	—	0.584	0.087	0.151	0.595	0.182	0.278	0.543	0.317	**0.401**

6.4.2.3 Scenario C - YES Stemming and NO Stopword Removal

In this section, we consider the several scenarios that k (size of the fingerprint) and j (minimum size of the tweet features) stopwords are not removed but stemming used, Table 6.20.

The pattern is the same as with previous early analysis. Performance is much worse than that obtained with Data Set I. Additionally, the results are exactly the same as Scenario A,

Table 6.19: Results for Data Set II - no stemming and with stopword removal

		Threshold 0.5			Threshold 0.15			Threshold 0.10			Threshold 0.05		
1	10	0.455	0.003	0.005	0.642	0.078	0.140	0.642	0.080	0.143	0.582	0.131	0.214
1	15	0.400	0.001	0.002	0.642	0.091	0.160	0.625	0.106	0.181	0.583	0.140	0.226
1	20	0.000	0.000	—	0.637	0.099	0.172	0.619	0.118	0.198	0.590	0.142	0.228
1	25	0.000	0.000	—	0.642	0.102	0.177	0.591	0.121	0.201	0.583	0.147	0.234
1	30	0.000	0.000	—	0.643	0.103	0.177	0.595	0.125	0.206	0.580	0.151	0.240
1	50	0.000	0.000	—	0.637	0.072	0.130	0.588	0.140	0.226	0.572	0.171	0.263
1	100	0.000	0.000	—	0.619	0.057	0.104	0.613	0.135	0.222	0.568	0.230	0.328
1	250	0.000	0.000	—	0.599	0.083	0.146	0.603	0.171	0.266	0.542	0.322	**0.404**
2	10	0.455	0.003	0.005	0.642	0.078	0.140	0.644	0.081	0.144	0.582	0.131	0.214
2	15	0.400	0.001	0.002	0.642	0.092	0.161	0.625	0.106	0.181	0.583	0.140	0.226
2	20	0.000	0.000	—	0.630	0.100	0.172	0.618	0.118	0.198	0.589	0.142	0.228
2	25	0.000	0.000	—	0.639	0.104	0.179	0.591	0.121	0.201	0.581	0.147	0.235
2	30	0.000	0.000	—	0.647	0.104	0.180	0.592	0.125	0.206	0.581	0.152	0.241
2	50	0.000	0.000	—	0.643	0.074	0.133	0.586	0.141	0.227	0.574	0.172	0.265
2	100	0.000	0.000	—	0.626	0.061	0.110	0.610	0.138	0.226	0.563	0.231	0.328
2	250	0.000	0.000	—	0.595	0.088	0.153	0.610	0.178	0.275	0.545	0.321	**0.404**
3	10	0.455	0.003	0.005	0.642	0.078	0.140	0.643	0.081	0.145	0.572	0.135	0.218
3	15	0.400	0.001	0.002	0.639	0.095	0.165	0.625	0.106	0.181	0.574	0.143	0.228
3	20	0.000	0.000	—	0.625	0.100	0.173	0.614	0.118	0.198	0.582	0.145	0.232
3	25	0.000	0.000	—	0.633	0.105	0.181	0.588	0.121	0.201	0.577	0.148	0.235
3	30	0.000	0.000	—	0.637	0.105	0.181	0.590	0.125	0.207	0.576	0.153	0.241
3	50	0.000	0.000	—	0.646	0.084	0.148	0.580	0.142	0.228	0.568	0.171	0.263
3	100	0.000	0.000	—	0.606	0.068	0.123	0.610	0.150	0.241	0.564	0.233	0.329
3	250	0.000	0.000	—	0.591	0.096	0.165	0.598	0.186	0.284	0.538	0.322	0.403
4	10	0.455	0.003	0.005	0.642	0.078	0.140	0.641	0.082	0.145	0.585	0.135	0.220
4	15	0.500	0.002	0.003	0.643	0.097	0.169	0.623	0.106	0.181	0.576	0.143	0.229
4	20	0.000	0.000	—	0.629	0.104	0.179	0.618	0.118	0.198	0.586	0.147	0.235
4	25	0.000	0.000	—	0.631	0.109	0.186	0.597	0.120	0.200	0.582	0.148	0.236
4	30	0.000	0.000	—	0.630	0.107	0.184	0.596	0.125	0.207	0.584	0.152	0.241
4	50	0.000	0.000	—	0.644	0.090	0.158	0.582	0.142	0.228	0.573	0.172	0.265
4	100	0.000	0.000	—	0.605	0.073	0.131	0.604	0.156	0.248	0.565	0.234	0.331
4	250	0.000	0.000	—	0.589	0.101	0.172	0.591	0.200	0.299	0.537	0.322	0.402

Table 6.20: Results for Data Set II - with stemming and no stopword removal

		Threshold 0.5			Threshold 0.15			Threshold 0.10			Threshold 0.05		
1	10	0.429	0.002	0.003	0.642	0.078	0.140	0.642	0.080	0.143	0.586	0.127	0.208
1	15	0.250	0.001	0.001	0.647	0.083	0.147	0.623	0.105	0.180	0.588	0.134	0.219
1	20	0.000	0.000	—	0.641	0.094	0.163	0.623	0.113	0.191	0.586	0.131	0.215
1	25	0.000	0.000	—	0.640	0.093	0.163	0.609	0.117	0.196	0.586	0.138	0.224
1	30	0.000	0.000	—	0.684	0.088	0.156	0.600	0.119	0.199	0.572	0.146	0.233
1	50	—	0.000	—	0.564	0.034	0.063	0.613	0.123	0.204	0.574	0.163	0.254
1	100	—	0.000	—	0.495	0.023	0.045	0.628	0.082	0.146	0.579	0.203	0.300
1	250	—	0.000	—	0.567	0.028	0.053	0.593	0.092	0.160	0.567	0.262	0.359
2	10	0.429	0.002	0.003	0.642	0.078	0.140	0.642	0.080	0.143	0.585	0.127	0.208
2	15	0.250	0.001	0.001	0.650	0.086	0.152	0.623	0.105	0.180	0.590	0.135	0.220
2	20	0.000	0.000	—	0.641	0.094	0.163	0.624	0.113	0.191	0.586	0.132	0.215
2	25	0.000	0.000	—	0.641	0.094	0.163	0.605	0.117	0.196	0.584	0.140	0.226
2	30	0.000	0.000	—	0.669	0.089	0.156	0.597	0.121	0.201	0.574	0.148	0.235
2	50	—	0.000	—	0.600	0.040	0.074	0.605	0.126	0.209	0.573	0.164	0.256
2	100	—	0.000	—	0.490	0.026	0.049	0.638	0.088	0.155	0.580	0.207	0.305
2	250	—	0.000	—	0.591	0.033	0.063	0.580	0.096	0.165	0.571	0.280	0.376
3	10	0.500	0.002	0.004	0.642	0.078	0.140	0.641	0.081	0.144	0.578	0.130	0.213
3	15	0.400	0.001	0.002	0.640	0.090	0.157	0.623	0.105	0.180	0.578	0.138	0.223
3	20	0.000	0.000	—	0.638	0.098	0.170	0.623	0.115	0.194	0.589	0.138	0.224
3	25	0.000	0.000	—	0.642	0.101	0.175	0.589	0.120	0.199	0.577	0.146	0.233
3	30	0.000	0.000	—	0.642	0.099	0.171	0.589	0.123	0.204	0.574	0.151	0.239
3	50	—	0.000	—	0.629	0.063	0.115	0.581	0.134	0.218	0.568	0.169	0.260
3	100	—	0.000	—	0.586	0.048	0.089	0.602	0.119	0.198	0.558	0.221	0.317
3	250	—	0.000	—	0.586	0.066	0.118	0.582	0.138	0.223	0.548	0.308	0.394
4	10	0.500	0.002	0.004	0.642	0.078	0.140	0.640	0.081	0.145	0.581	0.131	0.214
4	15	0.500	0.002	0.003	0.640	0.095	0.166	0.622	0.105	0.180	0.579	0.142	0.227
4	20	0.000	0.000	—	0.632	0.101	0.175	0.620	0.116	0.196	0.587	0.143	0.230
4	25	0.000	0.000	—	0.634	0.105	0.180	0.597	0.120	0.200	0.582	0.146	0.234
4	30	0.000	0.000	—	0.631	0.105	0.181	0.598	0.123	0.204	0.585	0.151	0.240
4	50	—	0.000	—	0.642	0.083	0.147	0.582	0.137	0.222	0.575	0.172	0.264
4	100	—	0.000	—	0.592	0.062	0.112	0.603	0.144	0.232	0.566	0.229	0.326
4	250	—	0.000	—	0.584	0.087	0.151	0.595	0.182	0.278	0.543	0.317	**0.401**

further reinforcing the previous conclusion that stemming offers no value for Twitter Topic Fuzzy Fingerprint.

6.4.2.4 Scenario D - YES Stemming and YES Stopword Removal

In this section, we consider the several scenarios that k (size of the fingerprint) and j (minimum size of the tweet features) stopwords are removed and stemming used.
Please consider Table 6.21. Once again, performance is below par, when compared to Data Set I results and Scenario D provides the same results as Scenario B. The best case scenario occurs when f-measure= 0.404, for $j = 1$ and $k = 250$.

6.4.3 Result Analysis

At this point, it is obvious that the Twitter Topic Fuzzy Fingerprint algorithm performed much worse for Data Set II. This is a consequence of a few characteristics of this particular Data Set and its goal.
Firstly, the goal of TASS was to categorize the tweets according to broad subjects (politics, music, football,etc...) as opposed to specific topics such as the trends in Data Set I. This goes back to the idea put forth in Chapter 2, that there is a difference between Topic Classification and Topic Detection. The Twitter Topic Fuzzy Fingerprint clearly performs better when considering the latter.
Secondly, the quality of the data itself is, in our opinion, questionable. Proof of this theory can be shown with the following two tweets from user "David_Busta" (spanish musician), taken from the test set and that were expected to be classified as "música":

Buenas Noches! Y dulces sueños

Buenos días familia!!! Os deseo un feliz día!;-)

These are hardly music related tweets beyond the fact that they were posted by musician David Bustamante. The only real word that will match between these tweets and the "música" fingerprint is the username itself, which will result in a very low T2S2 score. This example is symptomatic of the quality of the data and also relates to the previous point that Topic Classification is too broad for the context of Tweet Topic Detection.
Thirdly, T2S2 values are so low that the threshold value had to be brought down all the way to 0.05. It is fair to assume that higher values of k and an even lower threshold value might

Table 6.21: Results for Data Set II - with stemming and with stopword removal

		Threshold 0.5			Threshold 0.15			Threshold 0.10			Threshold 0.05		
1	10	0.455	0.003	0.005	0.642	0.078	0.140	0.642	0.080	0.143	0.582	0.131	0.214
1	15	0.400	0.001	0.002	0.642	0.091	0.160	0.625	0.106	0.181	0.583	0.140	0.226
1	20	0.000	0.000	—	0.637	0.099	0.172	0.619	0.118	0.198	0.590	0.142	0.228
1	25	0.000	0.000	—	0.642	0.102	0.177	0.591	0.121	0.201	0.583	0.147	0.234
1	30	0.000	0.000	—	0.643	0.103	0.177	0.595	0.125	0.206	0.580	0.151	0.240
1	50	0.000	0.000	—	0.637	0.072	0.130	0.588	0.140	0.226	0.572	0.171	0.263
1	100	0.000	0.000	—	0.619	0.057	0.104	0.613	0.135	0.222	0.568	0.230	0.328
1	250	0.000	0.000	—	0.599	0.083	0.146	0.603	0.171	0.266	0.542	0.322	**0.404**
2	10	0.455	0.003	0.005	0.642	0.078	0.140	0.644	0.081	0.144	0.582	0.131	0.214
2	15	0.400	0.001	0.002	0.642	0.092	0.161	0.625	0.106	0.181	0.583	0.140	0.226
2	20	0.000	0.000	—	0.630	0.100	0.172	0.618	0.118	0.198	0.589	0.142	0.228
2	25	0.000	0.000	—	0.639	0.104	0.179	0.591	0.121	0.201	0.581	0.147	0.235
2	30	0.000	0.000	—	0.647	0.104	0.180	0.592	0.125	0.206	0.581	0.152	0.241
2	50	0.000	0.000	—	0.643	0.074	0.133	0.586	0.141	0.227	0.574	0.172	0.265
2	100	0.000	0.000	—	0.626	0.061	0.110	0.610	0.138	0.226	0.563	0.231	0.328
2	250	0.000	0.000	—	0.595	0.088	0.153	0.610	0.178	0.275	0.545	0.321	**0.404**
3	10	0.455	0.003	0.005	0.642	0.078	0.140	0.643	0.081	0.145	0.572	0.135	0.218
3	15	0.400	0.001	0.002	0.639	0.095	0.165	0.625	0.106	0.181	0.574	0.143	0.228
3	20	0.000	0.000	—	0.625	0.100	0.173	0.614	0.118	0.198	0.582	0.145	0.232
3	25	0.000	0.000	—	0.633	0.105	0.181	0.588	0.121	0.201	0.577	0.148	0.235
3	30	0.000	0.000	—	0.637	0.105	0.181	0.590	0.125	0.207	0.576	0.153	0.241
3	50	0.000	0.000	—	0.646	0.084	0.148	0.580	0.142	0.228	0.568	0.171	0.263
3	100	0.000	0.000	—	0.606	0.068	0.123	0.610	0.150	0.241	0.564	0.233	0.329
3	250	0.000	0.000	—	0.591	0.096	0.165	0.598	0.186	0.284	0.538	0.322	0.403
4	10	0.455	0.003	0.005	0.642	0.078	0.140	0.641	0.082	0.145	0.585	0.135	0.220
4	15	0.500	0.002	0.003	0.643	0.097	0.169	0.623	0.106	0.181	0.576	0.143	0.229
4	20	0.000	0.000	—	0.629	0.104	0.179	0.618	0.118	0.198	0.586	0.147	0.235
4	25	0.000	0.000	—	0.631	0.109	0.186	0.597	0.120	0.200	0.582	0.148	0.236
4	30	0.000	0.000	—	0.630	0.107	0.184	0.596	0.125	0.207	0.584	0.152	0.241
4	50	0.000	0.000	—	0.644	0.090	0.158	0.582	0.142	0.228	0.573	0.172	0.265
4	100	0.000	0.000	—	0.605	0.073	0.131	0.604	0.156	0.248	0.565	0.234	0.331
4	250	0.000	0.000	—	0.589	0.101	0.172	0.591	0.200	0.299	0.537	0.322	0.402

provide better results. However, given the previously mentioned contextual differences, it felt ineffective to further pursue that approach.

Lastly, according to [29], the best performing algorithm with the TASS development data set achieved only precision=0.59, recall= 0.69 and f-measure= 0.634. The results in [29] were achieved through a well tuned method aimed at the specificity of this data set, while we simply used a novel method perfected from a different context (Data Set I). Should we fine tune the Twitter Topic Fuzzy Fingerprints to perform with this data, better results could probably be achieved, as it is also a much lighter processing method than the one presented in [29].

Chapter 7

Conclusion

In this thesis, we proposed a novel method in Tweet Topic Detection - the Twitter Topic Fuzzy Fingerprint.

Inspired from Nuno Homem's and João Carvalho's work in [27], we build a fingerprint for the top Twitter trends of a given day and then set out to attempt a detect that topic in the approximately 85% of tweets that do not have any hashtags.

With an (assumedly) short test set, the algorithm provided very positive results and primes by both its speed and performance, in comparison to other well known classification methods (kNN and SVM). When considering a small fingerprint (top 20 keywords), without removing stopwords nor performing stemming and all words with 3 characters or more being kept in the corpus, an f-measure value of 0.833 was achieved for a decision threshold= 0.10.

The main setback of the Twitter Topic Fuzzy Fingerprint is that it is only really effective within the specific context of the Topic/Trend Detection. If broader categories are considered, such as those in Data Set III, the performance will drop considerately.

7.1 Future Work

There are several interesting tasks that can spawn from this thesis, in order to further attempt to improve the current Twitter Topic Fuzzy Fingerprint method.

On the one hand, the inclusion of the Filtered Space-Saving Algorithm used in the original method could provide faster trend detection, although in detriment of performance. On the other hand, new membership functions may be proposed to either enhance performance or simplify calculations and thus improve the algorithms quickness.

The T2S2 calculation, as it stands in Eq. (5.5), results in normalized values but with a very low threshold value (at 0.10 it decides that the tweet belongs to the topic). An overhaul of this

function should be considered, in order to try and reach a higher threshold value, such as 0.5. It should also be interesting to study how many tweets per trend are required for the training set to have, so that performance is optimized. Should fewer tweets be necessary, it could further validate the use of the Twitter Topic Fuzzy Fingerprint as a real-time classifier.

Given that trending topics will often relay some worldwide important events, it may also provide value to create the fingerprint from other sources such as blogs and news agencies sites. This may enrich the fingerprint enough so that better results are achieved when detecting the topic of a tweet.

Bibliography

[1] K. Wicker, *https://blog.twitter.com/2013/celebrating-twitter7*.

[2] A. Mazzia and J. Juett, "Suggesting hashtags on twitter," 2010.

[3] R. Feldman and J. Sanger, *Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. New York, NY, USA: Cambridge University Press, 2006.

[4] M. Konchady, *Text Mining Application Programming*. Charles River Media, 2006.

[5] A. Rajaraman, J. Leskovec, and J. Ullman, *Mining of Massive Datasets*. Cambridge University Press, 2011.

[6] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008.

[7] Y. Yang and X. Liu, "A re-examination of text categorization methods," in *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, (New York, NY, USA), pp. 42–49, ACM, 1999.

[8] J. Zhang and I. Mani, "KNN Approach to Unbalanced Data Distributions: A Case Study Involving Information Extraction," in *Proceedings of the ICML'2003 Workshop on Learning from Imbalanced Datasets*, 2003.

[9] A. B. Hur and J. Weston, "A User's Guide to Support Vector Machines," pp. 1–18.

[10] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[11] M. Cataldi, L. Di Caro, and C. Schifanella, "Emerging topic detection on twitter based on temporal and social terms evaluation," in *Proceedings of the Tenth International Workshop on Multimedia Data Mining*, MDMKDD '10, (New York, NY, USA), pp. 4:1–4:10, ACM, 2010.

[12] M. Mathioudakis and N. Koudas, "Twittermonitor: Trend detection over the twitter stream," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, (New York, NY, USA), pp. 1155–1158, ACM, 2010.

[13] S. P. Kasiviswanathan, P. Melville, A. Banerjee, and V. Sindhwani, "Emerging topic detection using dictionary learning," in *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, (New York, NY, USA), pp. 745–754, ACM, 2011.

[14] A. Saha and V. Sindhwani, "Learning evolving and emerging topics in social media: A dynamic nmf approach with temporal regularization," in *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, WSDM '12, (New York, NY, USA), pp. 693–702, ACM, 2012.

[15] J. Weng and B.-S. Lee, "Event detection in twitter," in *ICWSM*, 2011.

[16] K. Lee, D. Palsetia, R. Narayanan, M. M. A. Patwary, A. Agrawal, and A. Choudhary, "Twitter trending topic classification," in *Proceedings of the 2011 IEEE 11th International Conference on Data Mining Workshops*, ICDMW '11, (Washington, DC, USA), pp. 251–258, IEEE Computer Society, 2011.

[17] J. Sankaranarayanan, H. Samet, B. E. Teitler, M. D. Lieberman, and J. Sperling, "Twitterstand: News in tweets," in *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '09, (New York, NY, USA), pp. 42–51, ACM, 2009.

[18] M. Cheong and V. Lee, "Integrating web-based intelligence retrieval and decision-making from the twitter trends knowledge base," in *SWSM '09: Proceeding of the 2nd ACM workshop on Social web search and mining*, (New York, NY, USA), pp. 1–8, ACM, 2009.

[19] *https://dev.twitter.com/discussions/6789.*

[20] A. Roomann-Kurrik, *Introducing new metadata for Tweets - https://dev.twitter.com/blog/introducing-new-metadata-for-tweets.*

[21] *To trend or not to trend - https://blog.twitter.com/2010/trend-or-not-trend.*

[22] *Natural Language Toolkit - http://nltk.org/.*

[23] M. Porter, "An algorithm for suffix stripping," *Program: electronic library and information systems*, vol. 14, no. 3, pp. 130–137, 1980.

[24] C. Horn, "Analysis and classification of twitter messages," 2010.

[25] J. P. Carvalho and L. Coheur, "Introducing uws - a fuzzy based word similarity function with good discrimination capability: Preliminary results," in *FUZZ-IEEE*, pp. 1–8, 2013.

[26] *http://mwh.geek.nz/2009/04/26/python-damerau-levenshtein-distance/.*

[27] N. Homem and J. Carvalho, "Authorship identification and author fuzzy fingerprints," in *30th Annual Conference of the North American Fuzzy Information Processing Society*, NAFIPS2011, 2011.

[28] *Weka - http://www.cs.waikato.ac.nz/ml/weka/.*

[29] F. B. y Ricardo Ribeiro, "Sentiment analysis and topic classification based on binary maximum entropy classifiers," *Procesamiento del Lenguaje Natural*, vol. 50, no. 0, 2012.

Appendix A

Data Sets Distribution

Table A.1: Training Set Trend Distribution

Top Trend	#tweets
#askjennette	136
#assistacaradesantarestart	12
#b1a41stwin	255
#finalcopatve	30
#gobiernodecallerevolucionpopular	498
#gtmo17	17
#ilovegodbecause	989
#jedwardtvpov	104
#mtvonedirection	145
#muriovidela	128
#mydemitop3songs	497
#nowplaying	7959
#obrigadogioantonelliealexandrenero	198
#replacesonglyricswithnutsack	532
#thevampstwitcam	1
#tropaunidaconmaduro	6296
#ultimovoo	178
#vazajorge	350
#wecantstop	1590
#welcometoitalyguys	302
#whataboutlove	552

Table A.2: Test Set Trend Distribution

Top Trend	#tweets
#askjennette	22
#assistacaradesantarestart	5
#b1a41stwin	84
#finalcopatve	56
#gobiernodecallerevolucionpopular	28
#gtmo17	9
#ilovegodbecause	10
#jedwardtvpov	9
#mtvonedirection	20
#muriovidela	30
#mydemitop3songs	8
#nowplaying	65
#obrigadogioantonelliealexandrenero	40
#replacesonglyricswithnutsack	2
#thevampstwitcam	3
#tropaunidaconmaduro	15
#ultimovoo	51
#vazajorge	18
#wecantstop	48
#welcometoitalyguys	6
#whataboutlove	56

Appendix B

Lists of Words

B.1 English Stopwords

i, me, my, myself, we, our, ours, ourselves, you, your, yours, yourself, yourselves, he, him, his, himself, she, her, hers, herself, it, its, itself, they, them, their, theirs, themselves, what, which, who, whom, this, that, these, those, am, is, are, was, were, be, been, being, have, has, had, having, do, does, did, doing, a, an, the, and, but, if, or, because, as, until, while, of, at, by, for, with, about, against, between, into, through, during, before, after, above, below, to, from, up, down, in, out, on, off, over, under, again, further, then, once, here, there, when, where, why, how, all, any, both, each, few, more, most, other, some, such, no, nor, not, only, own, same, so, than, too, very, s, t, can, will, just, don, should, now

B.2 Spanish Stopwords

de, la, que, el, en, y, a, los, del, se, las, por, un, para, con, no, una, su, al, lo, como, más, pero, sus, le, ya, o, este, sí, porque, esta, entre, cuando, muy, sin, sobre, también, me, hasta, hay, donde, quien, desde, todo, nos, durante, todos, uno, les, ni, contra, otros, ese, eso, ante, ellos, e, esto, mí, antes, algunos, qué, unos, yo, otro, otras, otra, él, tanto, esa, estos, mucho, quienes, nada, muchos, cual, poco, ella, estar, estas, algunas, algo, nosotros, mi, mis, tú, te, ti, tu, tus, ellas, nosotras, vosostros, vosostras, os, mío, mía, míos, mías, tuyo, tuya, tuyos, tuyas, suyo, suya, suyos, suyas, nuestro, nuestra, nuestros, nuestras, vuestro, vuestra, vuestros, vuestras, esos, esas, estoy, estás, está, estamos, estáis, están, esté, estés, estemos, estéis, estén, estaré, estarás, estará, estaremos, estaréis, estarán, estaría, estarías, estaríamos, estaríais, estarían, estaba, estabas, estábamos, estabais, estaban, estuve, estuviste, estuvo, estuvimos, estuvisteis, estuvieron, estuviera, estuvieras, estuviéramos, estuvierais, estuvieran, estuviese, estuvieses, estuviésemos, estuvieseis, estuviesen, estando, estado, estada, estados, estadas, estad, he, has, ha, hemos,

habéis, han, haya, hayas, hayamos, hayáis, hayan, habré, habrás, habrá, habremos, habréis, habrán, habría, habrías, habríamos, habríais, habrían, había, habías, habíamos, habíais, habían, hube, hubiste, hubo, hubimos, hubisteis, hubieron, hubiera, hubieras, hubiéramos, hubierais, hubieran, hubiese, hubieses, hubiésemos, hubieseis, hubiesen, habiendo, habido, habida, habidos, habidas, soy, eres, es, somos, sois, son, sea, seas, seamos, seáis, sean, seré, serás, será, seremos, seréis, serán, sería, serías, seríamos, seríais, serían, era, eras, éramos, erais, eran, fui, fuiste, fue, fuimos, fuisteis, fueron, fuera, fueras, fuéramos, fuerais, fueran, fuese, fueses, fuésemos, fueseis, fuesen, sintiendo, sentido, sentida, sentidos, sentidas, siente, sentid, tengo, tienes, tiene, tenemos, tenéis, tienen, tenga, tengas, tengamos, tengáis, tengan, tendré, tendrás, tendrá, tendremos, tendréis, tendrán, tendría, tendrías, tendríamos, tendríais, tendrían, tenía, tenías, teníamos, teníais, tenían, tuve, tuviste, tuvo, tuvimos, tuvisteis, tuvieron, tuviera, tuvieras, tuviéramos, tuvierais, tuvieran, tuviese, tuvieses, tuviésemos, tuvieseis, tuviesen, teniendo, tenido, tenida, tenidos, tenidas, tened

B.3 Portuguese Stopwords

de, a, o, que, e, do, da, em, um, para, com, não, uma, os, no, se, na, por, mais, as, dos, como, mas, ao, ele, das, à, seu, sua, ou, quando, muito, nos, já, eu, também, só, pelo, pela, até, isso, ela, entre, depois, sem, mesmo, aos, seus, quem, nas, me, esse, eles, você, essa, num, nem, suas, meu, às, minha, numa, pelos, elas, qual, nós, lhe, deles, essas, esses, pelas, este, dele, tu, te, vocês, vos, lhes, meus, minhas, teu, tua, teus, tuas, nosso, nossa, nossos, nossas, dela, delas, esta, estes, estas, aquele, aquela, aqueles, aquelas, isto, aquilo, estou, está, estamos, estão, estive, esteve, estivemos, estiveram, estava, estávamos, estavam, estivera, estivéramos, esteja, estejamos, estejam, estivesse, estivéssemos, estivessem, estiver, estivermos, estiverem, hei, há, havemos, hão, houve, houvemos, houveram, houvera, houvéramos, haja, hajamos, hajam, houvesse, houvéssemos, houvessem, houver, houvermos, houverem, houverei, houverá, houveremos, houverão, houveria, houveríamos, houveriam, sou, somos, são, era, éramos, eram, fui, foi, fomos, foram, fora, fôramos, seja, sejamos, sejam, fosse, fôssemos, fossem, for, formos, forem, serei, será, seremos, serão, seria, seríamos, seriam, tenho, tem, temos, tém, tinha, tínhamos, tinham, tive, teve, tivemos, tiveram, tivera, tivéramos, tenha, tenhamos, tenham, tivesse, tivéssemos, tivessem, tiver, tivermos, tiverem, terei, terá, teremos, terão, teria, teríamos, teriam