Detecting a Tweet's Topic within a Large Number of Portuguese Twitter Trends

Hugo Rosa[1], João Paulo Carvalho[2], and Fernando Batista[3]

1 **INESC-ID Lisboa**
IST - Universidade de Lisboa, Portugal
hugohrosa@gmail.com
2 **INESC-ID Lisboa**
IST - Universidade de Lisboa, Portugal
joao.carvalho@inesc-id.pt
3 **INESC-ID Lisboa**
ISCTE-IUL, Lisboa, Portugal
fernando.batista@iscte.pt

Abstract

In this paper we propose to approach the subject of Twitter Topic Detection when in the presence of a large number of trending topics. We use a new technique, called Twitter Topic Fuzzy Fingerprints, and compare it with two popular text classification techniques, Support Vector Machines (SVM) and k-Nearest Neighbours (kNN). Preliminary results show that it outperforms the other two techniques, while still being much faster, which is an essential feature when processing large volumes of streaming data. We focused on a data set of Portuguese language tweets and the respective top trends as indicated by Twitter.

1998 ACM Subject Classification I.2.7 Natural Language Processing; H.2.8 Database Applications; I.5.4 Applications

Keywords and phrases topic detection; social networks data mining; Twitter; Portuguese language

Digital Object Identifier 10.4230/OASIcs.SLATE.2014.0

1 Introduction

No one can deny the importance of public social networks in current modern world society. From event advertising or idea dissemination, to commenting and analysis, 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. The Arab Spring [11], the Indignant movement protest [17], or presidents tweeting and posting messages on Facebook instead of using official public addressing are just a few examples of how influential social networks have become. Nowadays important events are often commented in social networks even before they become "public news", and even news agencies and networks had to adapt and start using social networks as sources of information.

Among public social networks, Twitter has become a major tool for sociological analysis. However, in order to properly analyse Twitter data, it is necessary to filter which tweets are relevant for a given subject or topic. This is not a trivial problem since there are currently more than 340 millions of daily tweets covering thousands of different topics [10]. Twitter already helps by providing a list of top trends [22] and the hashtag # mechanism: when referring to a certain topic, users are encouraged to indicate it through the use of a hashtag,

3rd Symposium on Languages Technologies and Applications (SLATE'14).
Editors: Maria João Varanda Pereira, José Paulo Leal and Alberto Simões; pp. 0–14

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

e.g., "#Obamacare has been approved!" indicates the topic of the tweet is Obamacare. Websites such as #hastags.org make good use of this information to present Twitter trends, e.g., `http://www.hashtags.org/analytics/Obamacare/`.

Other tools such as Twittermonitor [15] can also be used to obtain Twitter trends. However not all tweets related to a given topic are hashtagged. In fact, only roughly 16% of all tweets are hashtagged [16]. These numbers have been confirmed by our (assumedly) small experiments. Therefore, in order to properly analyse a given topic, it is essential to include the most of the remaining 84% of the untagged information.

This task, which we shall refer to as Tweet Topic Detection, involves deciding if a given tweet is related to a given #hashtagged topic. Basically this can be categorized as a classification problem, albeit one with some particular characteristics that need to be addressed specifically: it is a text classification problem, with an unknown and large number of categories, where the texts to be classified are very short texts (up to 140 characters), and it is a problem that fits the Big Data paradigm due to the huge amounts of streaming data.

We distinguish between Topic Classification and Topic Detection. The former defines a short and generic set of categories, ranging from politics to sports and the documents will often belong to at least one of those categories. It is very rare that a tweet does not fit into any topic. The latter takes on a more detailed approach, where an attempt is made to determine the topic of the document, given a predetermined large set of possible topics. 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.

When considering this difference, the most similar works on Topic Detection within Twitter are those related with emerging topics or trends, for example [15, 3, 12, 19]. In these works the authors use a wide variety of techniques regarding text analysis to find the most common related words and hence detect topics. In our work we already assume the existence of trending topics and we aim at efficient detecting tweets that are related to them, despite not being explicitly marked as so.

It is also possible to find several works regarding Topic Classification. In [14], an attempt is made to classify Twitter Trending Topics into 18 broad categories, such as: sports, politics, technology, etc, and their experiments on a database of randomly selected 768 trending topics (over 18 classes) show that, using text-based and network-based classification modelling, a classification accuracy up to 65% and 70% can be achieved, respectively. Another interesting article, despite not on the theme of Topic Detection, demonstrates how to use Twitter to automatically obtain breaking news from the tweets posted by Twitter users [20]. In 2009, when Michael Jackson passed away, "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". This further enforces the importance of automatically analysing the massive amount of information on Twitter.

In what concerns text classification, K-Nearest Neighbours (kNN) and the Support Vector Machine (SVM) are amongst the most widely used and best performing classifiers. In [23], Yang and Liu, performed several tests in a controlled study and reported that SVM and kNN are at least comparable to other well-known classification methods, including Neural Networks and Naive Bayes, and that significantly outperform the other methods when the number of positive training instances per category are small.

In this paper we propose a new approach to the subject of Twitter Topic Detection when in the presence of a large number of Portuguese trending topics: the use of and adaptation of the Fuzzy Fingerprints introduced in [9], associated with the use of Filtered Space Saving algorithm [8][10] and the fuzzy based automatic error correction mechanism presented in

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

Figure 1 Binary Bag-of-Words Representation.

[2]. This work is integrated within the MISNIS framework, being developed with the goal of Intelligent Mining of Public Social Networks' Influence in Society [9]. We present some preliminary results that show that the adapted Fuzzy Fingerprints outperform some of the most commonly classifiers (SVM and kNN) when applied to this particular problem. Additionally, in conjunction with the other techniques, the proposed Twitter topic detection process has additional advantages over the existing methods. In particular, it is significantly faster, and the resulting models are much smaller than SVM's.

The paper is organized as follows: First, we discuss several "Related Techniques" from similar fields of study such as Text Categorization and Document Representation. Secondly, we explain how our proposed method (Twitter Topic Fuzzy Fingerprints) works and how it stems from the Author Fuzzy Fingerprint in [9]. Then, we present the characteristics of the used data set and how evaluations were performed. Finally we evaluate the Twitter Topic Fuzzy Fingerprints with our data set of Portuguese tweets and present the comparison results to SVM and kNN.

2 Related Techniques

The goal of this work is essentially to automatically classify tweets into a set of trending topics. This process is broadly known in Natural Language Processing (NLP) as Text Categorization, and consists of finding the correct topic (or topics) for each document, given a set of categories (subjects, topics) and a collection of text documents [5].

The text contained in each tweet is the most relevant source of information for classification. However, text is an unstructured form of data that classifiers and learning algorithms cannot directly process [5]. For that reason, our documents/tweets must be converted into a more manageable form, during a preprocessing step.

2.1 Document Representation

One of the simplest and commonly used representation is the *bag-of-words* model. Frequently used in NLP and Information Retrieval (IR), it consists of representing a document as a set (bag) of its words, ignoring the syntax and even the word order, but keeping the frequency of each word. It uses all words in a document as features. Thus, the dimension of the feature space is equal to the number of different words in all documents [5]. This form of representation can be illustrated with the following example:

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

Based of these three texts, a dictionary of unique words can be constructed: {John, bought, a, car, I, love, driving, my}. The text collection can then be represented as a binary matrix, containing 8 columns, one per dictionary word, and 3 rows, one per text entry:

The matrix presented in Figure 1 indicates whether a given term exists in the document, without detailing on its importance to the collection of documents. An extended representation is known as the TF-IDF scheme and combines the concept of the term frequency with the inverse document frequency. The TF-IDF scheme is a scoring method that can tell the importance of a word in a collection of documents, and can be calculated as a simple multiplication:

$$tfidf = tf \times idf \tag{1}$$

The concept of term frequency (tf) is simply the number of occurrences 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).

$$idf = \log \frac{N}{n_i} \tag{2}$$

By combining the Eqs. (1) and (2), the TF-IDF of word in a document can be expressed by Eq. (3).

$$tfidf_i = tf \times \log \frac{N}{n_i} \tag{3}$$

As succinctly explained in [18], $tfidf$ assigns a weight to a term 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;

Different categorization methods can be applied to a structured document representation. In general, categorization algorithms follow the following four steps [5]:

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;

2.2 k-Nearest Neighbors Algorithm - kNN

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" [5]. 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 large proportion of them, a positive decision is made."

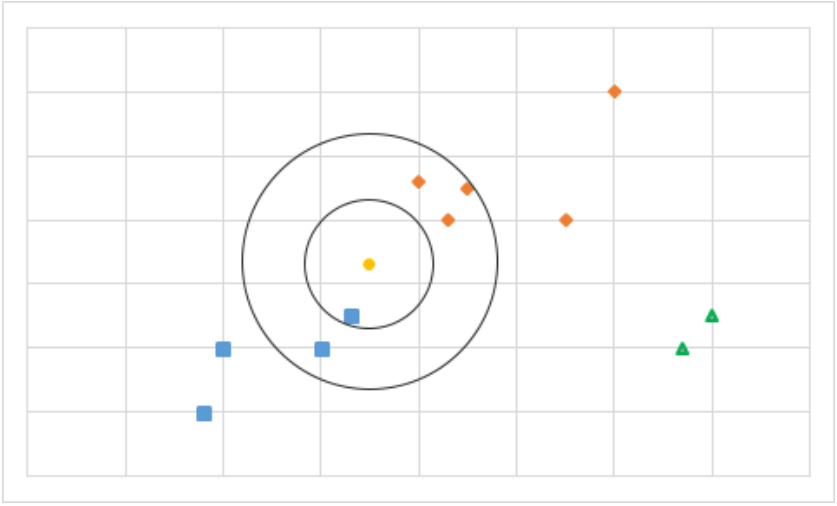

Figure 2 Example of the K Nearest Neighbour Algorithm.

In Figure 2, 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 neighbour 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 neighbours belong to the orange category and 2 to the blue category. By a majority rule, the document will be classified as orange.

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 neighbour, a high value of k can have too much noise in it and favour dominant categories. In fact, this algorithm is known to be affected by noisy data.

The kNN is considered to be one of the simplest and 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" [5]. 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" [13].

2.3 Support Vector Machines - SVM

A support vector machine (SVM) is a very fast and effective binary classifier. According to [13] "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, [5] 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".

Consider Figure 3 as a two dimensional example of SVM. As one would expect, in this scenario, the hyperplanes are lines. The figure reveals that 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

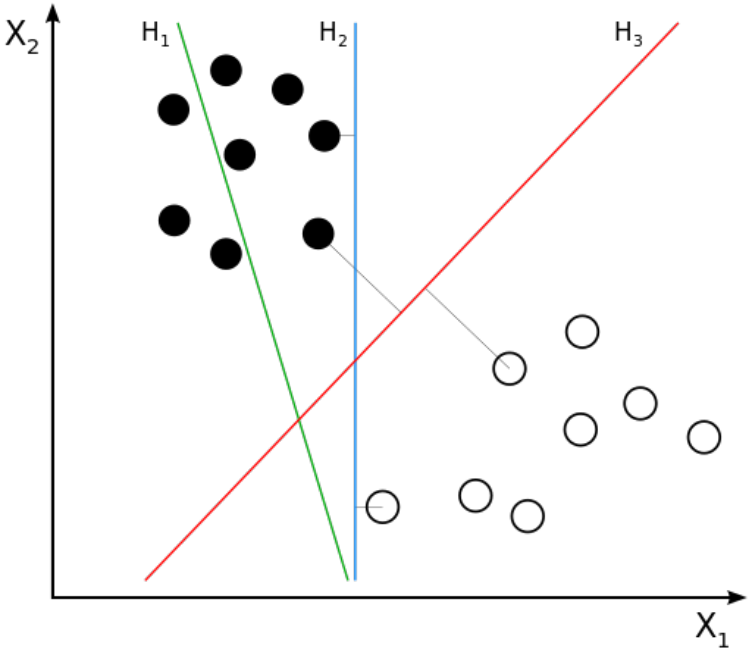

Figure 3 Two dimensional Support Vector Machine.

instances, which are called the support vectors" [5].

According to [13], SVM has at least three major differences with the previous categorization method:

1. Not all training documents are used. The SVM function is built only by documents near to the classification border;
2. An SVM can construct an irregular border to separate positive and negative training documents;
3. Not all features (unique words) from training documents are necessary for classification;

SVM methods for text categorization have recently attracted some attention since they are amongst the most accurate classifiers [13].

3 Twitter Topic Fuzzy Fingerprints

In this work we propose the use of an adaptation of the Fuzzy Fingerprints classification method described in [9] to tackle the problem of Topic Detection in Twitter. In [9] the authors approach the problem of text authorship by using the crime scene fingerprint analogy to claim that a given text has its authors writing style embedded in it. If the fingerprint is known, then it is possible to identify whether a text whose author is unknown, has a known author's fingerprint on it.

The algorithm itself works as following:

1. Gather the top-k word frequencies in all known texts of each known author;
2. Build the fingerprint by applying a fuzzifying function to the top-k list. The fuzzified fingerprint is based on the word order and not on the frequency value;

Listing 1 Pseudo-Code to explain data structure used.

```
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]++
```

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

The proposed fuzzy fingerprint method for Tweet Topic Detection, 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 [9] 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.

3.1 Building the Fingerprint Library

In order to build the fingerprint library, the proposed method goes over the training set, which, in this situation, are tweets containing the Trending Topics of the day. For each tweet, it acknowledges the existence of the # and adds each word in the tweet to a #topic table alongside with its counter of occurrences. Only the top-k most frequent words are considered. The algorithm presented in Figure 1 presents further details the this process.

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 account for of its Inverse Topic Frequency (ITF), an adaptation of the Inverse Document Frequency in Eq. (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 1 shows an example of a possible top-k output produced by the algorithm Figure

Table 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

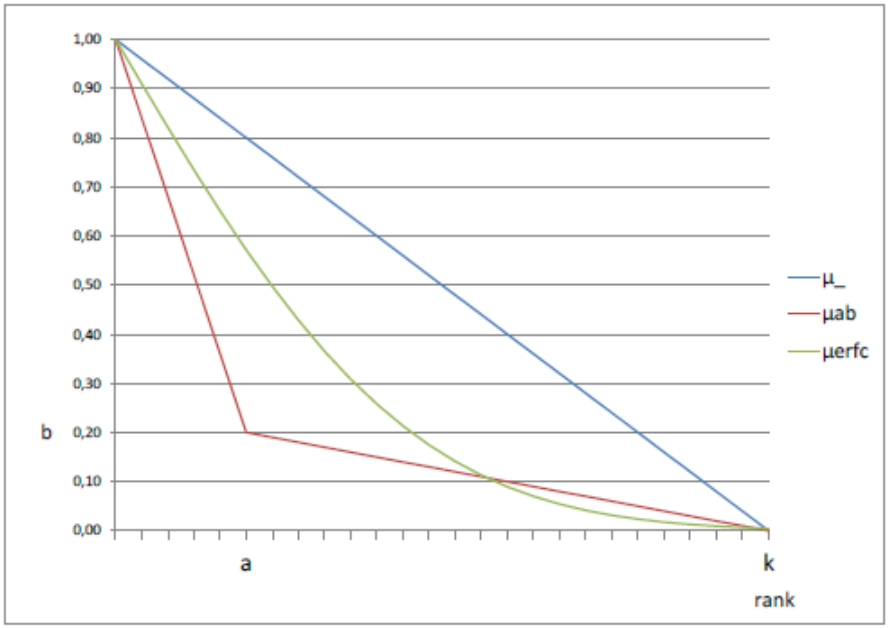

Figure 4 Fuzzyfing functions.

1 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 1. 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".

After obtaining the top-k list for a given #topic, we take the same approach as the original method, and use the membership Eq. 4 to build the fingerprint, where k is the size of the top-k fingerprint and i represents the membership index.

$$\mu_{ab}(i) = \begin{cases} 1 - (1-b)\frac{i}{kb} & i < a \\ \frac{a(1-\frac{i-a}{k-a})}{k} & i \geq a \end{cases} \tag{4}$$

Figure 4 shows the three membership functions that were considered and the impact of the parameters a and b on it. Much like in the original method, Eq. (4) was the best performing function and thus, the chosen one.

The fingerprint is a k sized bi-dimensional array containing in the first column the list of the top-k words, and in the second column its membership value $\mu_{ab}(i)$ obtained by the application of Eq. (4).

3.2 Tweet-Topic Similarity Score

In the original method, Eq. (4), in order to check the authorship of a given text, 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), 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).

$$T2S2(\Phi, T) = \frac{\sum_{v} \mu_{\Phi}(v) : v \in (\Phi \cap T)}{\sum_{i=0}^{j} \mu_{\Phi}(w_i)} \tag{5}$$

In (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 $w \in (\Phi)$. Eq. 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.

4 Twitter Data

Using Twitter's developer tools [21], we extracted samples of the public data flowing through Twitter by establishing a connection to a Twitter streaming endpoint. It is important to note that, using the sample API, only 1% of the actual public tweets can be retrieved [4]. Using this method, we obtained just over 1.2 million Portuguese tweets, from March, 14th to March 20th, 2014.

By executing Twitters DEV "GET Trends/place" method, one can obtain the top 10 trending topics of the moment in a given place. Using the WOEID (Where On Earth ID) for Brazil and Portugal, we extracted the top trends on the 17th of March mid-afternoon, 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. This can be explained by Twitter's view on what constitutes a trending topic.

According to [22], "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 favours novelty over popularity".

4.1 Training Data Set

Even though we only used 2 topics for testing purposes (due to the difficulty in annotating a high 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 600,000 tweets in Portuguese language, where the most popular trend is #kca (18,000 tweets) and the rarer is #1dnamix (139 tweets).

The fact that not all categories are trained with the same amount of samples makes for what is known as an unbalanced dataset. In this case, it may happen that one single category dominates the training set in such fashion, that some classifiers will incorrectly categorize most of the test set as belonging to that one category.

4.2 Test Data Set

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, i.e., manually assigned to one of the two target trends despite not the trend itself in the tweets' text.

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.

5 Evaluation Metrics

In this section, we take a look at the metrics used to determine how good or poorly a classifier performs. Typically there are three key concepts: 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 detection:

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:

$$Precision = \frac{\#TP}{\#TP + \#FP} \tag{6}$$

$$Recall = \frac{\#TP}{\#TP + \#FN} \tag{7}$$

$$F - Measure = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{8}$$

6 Results

Here we compare the Twitter Topic Fuzzy Fingerprint method up against the two algorithms we presented earlier: k-Nearest Neighbour (kNN) and Support Vector Machine (SVM). The exact same training data sets and test data sets were used for all methods. Several test scenarios were built to find each algorithm optimal performance setting.

6.1 Twitter Topic Fuzzy Fingerprint Performance

The following parameters were considered for the Twitter Topic Fuzzy Fingerprint:

1. k, size of the fuzzy fingerprint. Several increasing values were taken into to account, in order to determine whether a higher or lower 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. The stopword list was provided by the Natural Language Toolkit, [1];
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 work, values of 0.5, 0.25, 0.15 and 0.10 were tested;

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 tweet 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;

While the removal of stopwords provided better results (approximately 2%), the stemming technique provided literally no improvement. 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 fashion that a formal stemmer cannot process. Twitter's lingo is very unique due to the informal nature of social networking communication, and a Stemming algorithm can only truly be effective with formal and well written texts.

Table 2 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 $k = [5; 10; 15]$ peaking at $k = 40$, when no False Negative scenarios are identified.

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

Table 2 Twitter Topic Fuzzy Fingerprint Performance, with stopword removal but no stemming.

j	k	Precision	Recall	F-Measure
4	5	1.000	0.492	0.659
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

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 [6], the exact same Twitter Fuzzy Fingerprints method reached f-measure= 0.840 for a larger multi-language data set and with more target top trends (35).

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. On the other hand, this experiment also shows how the competing techniques are not as scalable: in [6], kNN also performed poorly, but the SVM f-measure= 0.79 was very close to the one obtained using the Twitter Fuzzy Fingerprints, albeit much less efficient in what concerns execution time. Here the performance of the competing techniques degrade to the point of being unusable.

6.2 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 1, with TF-IDF weighting, Eq. (3). The tests were performed using the WEKA framework [7].

kNN was executed with the $k = [3, 5, 10, 30, 50]$ and, due to the limitations of WEKA, the distance measure considered was the Euclidean distance as opposed to the more common cosine similarity. Despite extensive testing and parameter tunning, 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. In [6], when $k = 5$, an f-measure= 0.445 could be achieved, despite also being a consequence of classifying all test tweets in the majority class.

For SVM, a number of different parameters were tested and optimized, but [6] suggests that the best performance was achieved using a linear kernel and a small soft-margin value: $C = 0.01$, providing competitive results with the Twitter Fuzzy Fingerprints method.

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 three possible explanations for such a poor performance. Firstly, this is a very specific problem to which such broad and well known algorithms may not apply. Secondly 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. Finally, there is the unbalanced nature of the training data set, as explained by Zang and Inderjeet in [24]. When dealing with unbalanced data sets, kNN completely ignores the minority classes and will often mistakenly classify a tweet to the majority category.

6.3 Method Comparison

When comparing all 3 methods, it is evident that the Twitter Topic Fuzzy Fingerprint algorithm outperforms both kNN and SVM, in this particular case.

Table 3 Execution Speed Comparison.

Method	Preprocessing	Build Model	Evaluate
Fuzzy Fp	124.0s		0.02s
kNN	$>$ 1800s	0.02s	89.7s
SVM	$>$ 1800s	$>$ 14400s	

kNN is a lazy algorithm, where all computation is deferred until classification, which justifies that it takes so much longer evaluating such a small test data set. In what concerns to SVM, a significant part of the time is attributed to creating the model after the preprocessing stage. In our approach, building the model is just a linear function of the number of words being considered for training.

Finally, the size of the 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 prunning the list of relevant words at k.

7 Conclusions and Future work

We proposed a method for topic detection for micro bloging content, such as Twittter, when in the presence of a large number of trending topics. This method, known as Twitter Topic Fuzzy Fingerprints, outperforms two other commonly used algorithms, kNN and SVM.

Even when ignoring the obtained f-measure results, which can be biased by the size of the test dataset, the proposed method still presents several advantages that make it an undeniable alternative for on-the-fly, and possibly distributed, topic detection: 1) the size of the resulting model is significantly smaller than SVM models, which is an important issue to consider when performing distributed computation in different machines; 2) the classification is significantly faster than the other two methods, making it an interesting solution for on-the-fly processing of Big Data streams.

Since the annotated Portuguese data used in the paper is undeniably small, extended manually annotated test sets must be created in a near future in order to further confirm and validate the results here presented.

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

References

1 Steven Bird. NLTK: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, COLING-ACL'06, pages 69–72, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

2 J. P. Carvalho and L. Coheur. Introducing UWS - a fuzzy based word similarity function with good discrimination capability: Preliminary results. In *FUZZ-IEEE*, pages 1–8, 2013.

3 Mario Cataldi, Luigi Di Caro, and Claudio 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, pages 4:1–4:10, New York, NY, USA, 2010. ACM.

4 Sunil D M et al. Twitter developers - limit on streaming tweets. `https://dev.twitter.com/discussions/6789`. Accessed: 2014-03-28.

5 Ronen Feldman and James Sanger. *Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press, New York, NY, USA, 2006.

6 J. P. Carvalho H. Rosa and F. Batista. Twitter topic fuzzy fingerprints. *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2014)*, 2014.

7 Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.

8 N. Homem and J. P. Carvalho. Finding top-k elements in data streams. *Inf. Sci.*, 180(24):4958–4974, December 2010.

9 N. Homem and J. P. Carvalho. Authorship identification and author fuzzy fingerprints. In *30th Annual Conference of the North American Fuzzy Information Processing Society*, NAFIPS2011, 2011.

10 N. Homem and J. P. Carvalho. Finding top-k elements in a time-sliding window. *Evolving Systems*, 2(1):51–70, 2011.

11 Carol Huang. Facebook and Twitter key to Arab Spring uprisings: report. *The National*, 6 June 2011. `http://www.thenational.ae/news/uae-news/facebook-and-twitter-key-to-arab-spring-uprisings-report`.

12 Shiva Prasad Kasiviswanathan, Prem Melville, Arindam Banerjee, and Vikas Sindhwani. Emerging topic detection using dictionary learning. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 745–754, New York, NY, USA, 2011. ACM.

13 Manu Konchady. *Text Mining Application Programming*. Charles River Media, 2006.

14 Kathy Lee, Diana Palsetia, Ramanathan Narayanan, Md. Mostofa Ali Patwary, Ankit Agrawal, and Alok Choudhary. Twitter trending topic classification. In *Proceedings of the 2011 IEEE 11th International Conference on Data Mining Workshops*, ICDMW '11, pages 251–258, Washington, DC, USA, 2011. IEEE Computer Society.

15 Michael Mathioudakis and Nick Koudas. Twittermonitor: Trend detection over the twitter stream. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 1155–1158, New York, NY, USA, 2010. ACM.

16 Allie Mazzia and James Juett. Suggesting hashtags on twitter. Master's thesis, University of Michigan, 2010.

17 El País. El 15-M sacude el sistema. `http://politica.elpais.com/politica/2011/05/21/actualidad/1305999838_462379.html`, May 2011.

18 Anand Rajaraman, Juri Leskovec, and Jeffrey Ullman. *Mining of Massive Datasets*. Cambridge University Press, 2011.

19 Ankan Saha and Vikas 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, pages 693–702, New York, NY, USA, 2012. ACM.

20 Jagan Sankaranarayanan, Hanan Samet, Benjamin E. Teitler, Michael D. Lieberman, and Jon Sperling. Twitterstand: News in tweets. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '09, pages 42–51, New York, NY, USA, 2009. ACM.

21 Twitter. Twitter developer tools. `https://dev.twitter.com/`. Accessed: 2014-03-28.

22 Twitter. To trend or not to trend... `https://blog.twitter.com/2010/trend-or-not-trend`, 8 December 2010.

23 Yiming Yang and Xin 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, pages 42–49, New York, NY, USA, 1999. ACM.

24 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.