# Topic Detection within Public Social Networks

Hugo Rosa
Instituto Superior Técnico
Universidade de Lisboa, Portugal
Email: hugohrosa@gmail.com

***Abstract*—In this paper we propose to approach the subject of Twitter Topic Detection using a new technique called Twitter Topic Fuzzy Fingerprints. A comparison is made with two popular text classification techniques, Support Vector Machines (SVM) and $k$-Nearest Neighbors ($k$NN). Preliminary results show that Twitter Topic Fuzzy Fingerprints outperforms the other two techniques achieving better F-Measure, while still being much faster, which is an essential feature when processing large volumes of streaming data.**

## I. 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 [1], the Indignant movement protest [2], 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 analyze 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 [3]. Twitter already helps by providing a list of top trends [4] and the hashtag # mechanism: when referring to a certain topic, users are encouraged to indicate it through the use of a hashtag, e.g., "#Obamacare has been approved!" indicates the topic of the tweet is Obamacare. Websites such as #hastags.org make good use of these information to present twitter trends, e.g., *http://www.hashtags.org/analytics/Obamacare/*.

Other tools such as Twittermonitor [5] 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 [6]. These numbers have been confirmed by our (assumedly) small experiments. Therefore, in order to properly analyze 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 [5], [8]–[10]. 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 [11], 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 modeling, 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 [12]. 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 analyzing the massive amount of information on Twitter.

In what concerns text classification, K-Nearest Neighbors ($k$NN) and the Support Vector Machine (SVM) are amongst the most widely used and best performing classifiers. In [7], Yang and Liu, performed several tests in a controlled study and

reported that SVM and $k$NN 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: the use of and adaptation of the Fuzzy Fingerprints introduced in [13], associated with the use of Filtered Space Saving algorithm [14] [15] and the fuzzy based automatic error correction mechanism presented in [16]. This work is integrated within the MISNIS framework, being developed with the goal of Intelligent Mining of Public Social Networks' Influence in Society [13]. In the paper we present some preliminary results that show that the adapted Fuzzy Fingerprints outperform some of the most commonly classifiers (SVM and $k$NN) 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 [13]. Then, we present the characteristics of the used data set and how evaluations were performed. Finally we evaluate the Twitter Topic Fuzzy Fingerprints method and present the comparison results to SVM and $k$NN.

## II. 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 [17].

The text contained in each 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 [17]. For that reason, our documents/tweets must be converted into a more manageable form, during a preprocessing step.

### A. 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 [17]. This form of representation can be illustrated with the following example:

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

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

Fig. 1. Binary Bag-of-Words Representation

Based of these three texts, a dictionary of unique words can be constructed: {John, bought, a, car, I, love, driving, my}.

As shown in Figure 1, 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 ($tf$) can be 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;

There are several interesting variations of $tfidf$. In Figure 1 we showed an example of a the binary representation and, with Eq. (3), we presented $tfidf$ in its most common form.

In [19], 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. (4), where $w$

is the number of occurrences of a given term in a document with a total of $W$ tokens.

$$tfidf_i = \frac{w}{W} \times \log \frac{N}{n_i} \tag{4}$$

Different categorization methods can be applied to a structured document representation. In general, categorization algorithms follow the following four steps [17]:

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;

### B. $k$-Nearest Neighbors Algorithm - $k$NN

The $k$NN 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" [17]. 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 $k$NN to "decide whether a document $d$ belongs to a category $c$, $k$NN 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."

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. In fact, this algorithm is known to be affected by noisy data.

The $k$NN 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" [17]. In $k$NN, "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" [19].

### C. Support Vector Machines - SVM

A support vector machine (SVM) is a very fast and effective binary classifier. According to [19] "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, [17] 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 2 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 instances, which are called the support vectors" [17].

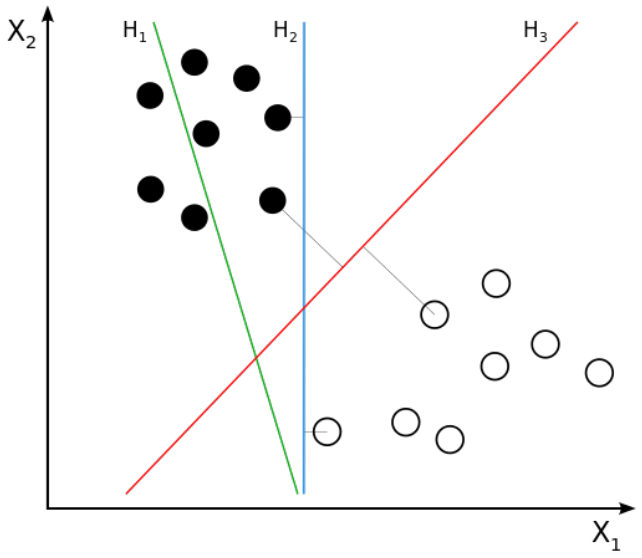


Fig. 2. Two dimensional Support Vector Machine

According to [19], 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 [19].

## III. Twitter Topic Fuzzy Fingerprints

In this work we propose the use of an adaptation of the Fuzzy Fingerprints classification method described in [13] to tackle the problem of Topic detection in Twitter. In [13] 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;

```
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]++
```

Fig. 3. Pseudo-Code to explain data structure used

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 all the topics' fingerprints, each unclassified (un-hashtagged) 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 [13] 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.

### A. 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 3 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 I shows an example of a possible top-$k$ output produced by the algorithm Figure 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 I. 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".

TABLE I
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 |

After obtaining the top-$k$ list for a given #topic, we take the same approach as the original method, and use the membership Eq. 5 to build the fingerprint, where $k$ is the size of the top-$k$ fingerprint and $i$ represents the membership index.

$$\mu_{ab}(i) = \begin{cases} 1 - (1-b)\frac{i}{kb} & i < a \\ \frac{a(1-\frac{i-a}{k-a})}{k} & i \geq a \end{cases} \quad (5)$$

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. (5).

### B. Tweet-Topic Similarity Score

In the original method, Eq. (5), 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. (6), 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).

$$T2S2(\Phi, T) = \frac{\sum_{v} \mu_{\Phi}(v) : v \in (\Phi \cap T)}{\sum_{i=0}^{j} \mu_{\Phi}(w_i)} \quad (6)$$

In Eq. (6) $\Phi$ 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. 6 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.

## IV. Data Set

Using Twitter's developer tools [20], 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 [21]. Nonetheless, with this method, we obtained nearly two million tweets per day from the 18th of May to the 21st of May, 2013.

By using Twitters DEV "GET Trends-Weekly" method, one can obtain the top trending topics for each day in a given week. The method belongs to the REST API, version 1.0. It should be noted that in August of 2013, the previous method was deprecated and replaced in version 1.1 by "GET Trends/place" where one can get the 10 top trending topics at the current time, filtered by geographical location.

From the above set of tweets, and by using the mentioned methods, we obtained our Training Data Set, which consists of approximately 21000 tweets containing the hashtag for the top trending English, Spanish and Portuguese topics of the day (Table II).

By analyzing the distribution of top trend hashtags across these tweets, 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.

The Test Data Set consists of 585 tweets that do not contain any of the top trending hashtags in Table II, although they may contain others. As shown in Table III, all 585 tweets were impartially annotated as belonging to one of topics above, in spite of not having the # it is claimed to belong too.

TABLE II
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 III
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 |

## V. 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:

$$Precision = \frac{\#TP}{\#TP + \#FP} \tag{7}$$

$$Recall = \frac{\#TP}{\#TP + \#FN} \tag{8}$$

$$F - Measure = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{9}$$

## VI. Results

Here we compare the Twitter Topic Fuzzy Fingerprint method up against the two algorithms we presented earlier: $k$-Nearest Neighbor ($k$NN) 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.

### A. Twitter Topic Fuzzy Fingerprint Performance

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

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.

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 trend (0.10)
- configuring a low value of $k$ for the size of the list of the fingerprint ($k = 20$)

TABLE IV
Twitter Topic Fuzzy Fingerprint Performance

| $j$ | $k$ | Precision | Recall | F-Measure |
|---|---|---|---|---|
| 3 | 10 | 0.741 | 0.769 | 0.755 |
| 3 | 15 | 0.748 | 0.816 | 0.780 |
| 3 | 20 | 0.756 | 0.928 | **0.833** |
| 3 | 25 | 0.746 | 0.936 | 0.831 |
| 3 | 30 | 0.725 | 0.941 | 0.819 |
| 3 | 50 | 0.704 | 0.948 | 0.808 |
| 3 | 100 | 0.693 | 0.952 | 0.802 |
| 3 | 250 | 0.615 | 0.964 | 0.751 |

- 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

Table IV resumes the algorithm's performance. Results reveal that the best value is f-measure=$0.833$. This represents a weighted average of precision=$0.756$ and recall=$0.928$. In other words, a precision of 75.6% means that of all the times our method said a topic was being correctly identified, only 24.4% were actually incorrect. In addition, a recall of 92.8% indicates that only 7.2% positive cases were left unidentified.

A slightly better performing configuration was found. It scored f-measure= $0.841$, when $j = 1$, $k = 25$ but with stopword removal activated. However, because of the step to remove stopwords, its execution time was 25 times larger than without stopword removal, which therefore made the scenario presented above the Most Effective Case Scenario on all accounts.

### B. kNN Performance

In order to test $k$NN, no stopwords were removed and 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 [22].

Table V contains $k$NN results for different values of $k$ and $j$, where $j$ is the minimum sized word, and $k$ is the number of neighbors that the algorithm will use to make its decision.

TABLE V
$k$NN's performance

| $j$ | $k$ | Precision | Recall | F-Measure |
|---|---|---|---|---|
| 1 | 1 | 0.479 | 0.135 | 0.211 |
| 1 | 3 | 0.466 | 0.321 | 0.380 |
| 1 | 5 | 0.467 | 0.280 | 0.350 |
| 1 | 10 | 0.332 | 0.229 | 0.271 |
| 2 | 1 | 0.509 | 0.137 | 0.216 |
| 2 | 3 | 0.453 | 0.324 | 0.378 |
| 2 | 5 | 0.422 | 0.285 | 0.340 |
| 2 | 10 | 0.369 | 0.227 | 0.281 |
| 3 | 1 | 0.486 | 0.137 | 0.214 |
| 3 | 3 | 0.448 | 0.319 | 0.373 |
| 3 | 5 | 0.441 | 0.299 | 0.356 |
| 3 | 10 | 0.360 | 0.247 | 0.293 |
| 4 | 1 | 0.634 | 0.142 | 0.232 |
| 4 | 3 | 0.466 | 0.341 | **0.394** |
| 4 | 5 | 0.470 | 0.324 | 0.384 |
| 4 | 10 | 0.463 | 0.247 | 0.322 |

The results are quite poor, with the best f-measure value peaking at an unimpressive 0.394, when 3 neighbors are consulted and $j = 4$, i.e., all words have a minimum size of 4 characters.

A possible explanation for this poor performance is the unbalanced nature of the training data set, as explained by Zang and Inderjeet in [23]. When dealing with unbalanced data sets, $k$NN completely ignores the minority classes and will often mistakenly classify a tweet to the majority category.

### C. SVM's Performance

In the case of SVM, thorough testing showed that the best performance was achieved when stopwords were removed and 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). We also used the SVM WEKA implementation. 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: $C = 0.01$.

Table VI contains the values obtained by using SVM's. $j$ represents the minimum size of the words kept in each scenario:

TABLE VI
SVM'S PERFORMANCE

| $j$ | Precision | Recall | F-Measure |
|---|---|---|---|
| 1 | 0.858 | 0.652 | 0.741 |
| 2 | 0.863 | 0.712 | **0.780** |
| 3 | 0.872 | 0.674 | 0.760 |
| 4 | 0.836 | 0.625 | 0.715 |

Overall, the performance is far better than $k$NN's, but still worse than Twitter Topic Fuzzy Fingerprints. When $j = 2$ it scores a f-measure=$0.780$.

### D. Results Comparison

When comparing all 3 methods, (Table VII), it is evident that the Twitter Topic Fuzzy Fingerprint algorithm outperforms both $k$NN and SVM. The proposed method achieves the best recall and f-measure value. The SVM strategy, while achieving better precision achieves almost 32% lower recall. The results achieved by $k$NN are definitely worse than the other two methods.

The reason why SVM performs outperforms $k$NN 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 $k$NN, exactly for the same reason of being unable to handle an unbalanced data set.

TABLE VII
OVERALL COMPARISON

| Method | Precision | Recall | F-Measure |
|---|---|---|---|
| Twitter Topic FFP ($j = 1$,$k = 25$) | 0.756 | 0.928 | 0.833 |
| $k$NN ($j = 4$,$k = 3$) | 0.466 | 0.341 | 0.394 |
| SVM ($j = 2$) | 0.863 | 0.712 | 0.780 |

The difference in performance is further accentuated if considering the same configuration for the commons parameters, i.e., the same value of $j = 1$, no stopwords removal and no stemming.

TABLE VIII
OVERALL COMPARISON II - $j = 1$

| Method | Precision | Recall | F-Measure |
|---|---|---|---|
| Twitter Topic FFp ($j = 1$,$k = 20$) | 0.756 | 0.928 | 0.833 |
| $k$NN ($j = 1$,$k = 20$) | 0.466 | 0.321 | 0.380 |
| SVM ($j = 1$) | 0.786 | 0.700 | 0.741 |

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 $k$NN 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, in our current conditions, the training time performance is several times better for fuzzy fingerprints. The classification is also straightforward. Table IX shows the processing time comparison. The proposed method preprocesses the training data, builds its model/fingerprint and classifies the test set several times faster than $k$NN and SVM. It is specially faster when evaluating the test set, which further reinforces the idea that the Twitter Fuzzy Fingerprints can be used as an on-the-fly method over a Twitter stream.

$k$NN 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 $k$.

TABLE IX
EXECUTION SPEED COMPARISON

| Method | Preprocessing | Build Model | Evaluate Test | Total |
|---|---|---|---|---|
| Fuzzy Fp | 2.030 s | | 0.507 s | 2.537 s |
| $k$NN | 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 |

## VII. Conclusions and Future work

We proposed a method for topic detection for micro bloging content, such as Twitter, that outperforms two other commonly used methods, $k$NN and SVM. The proposed method also presents several advantages that make it an undeniable alternative for on-the-fly, and possibly distributed, topic detection: 1) the training time can be comparable to $k$NN, which is a lazy algorithm, but is several times faster than SVM; 2) the size of the resulting model is also significantly smaller than SVM models, and that is an important issue to take into consideration when performing distributed computation in different machines; 3) the classification is also significantly faster than the other two methods, making it an interesting solution for on-the-fly processing of Big Data streams.

It should be noted that the data used here is still a rather small set when compared with the daily amount of Twitter data. In the future, extended manually annotated test sets must be created in order to further validate the results here presented.

## Acknowledgment

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

## References

[1] Huang, C. (June 6, 2011); *Facebook and Twitter key to Arab Spring uprisings: report*; The National. Retrieved April 13, 2012.
[2] *El 15 -M sacude el sistema*; El Pas. Retrieved 26 May 2011.
[3] Homem, N. Carvalho, J.P.; *Finding top-k elements in a time-sliding window*; Evolving Systems, 2(1), pp. 51-71, Jan. 2011, Springer
[4] *https://blog.twitter.com/2010/trend-or-not-trend*
[5] Mathioudakis, M., Koudas, N.; *TwitterMonitor: trend detection over the twitter stream*; Proc. of the 2010 international conference on Management of data, Indianapolis, Indiana, USA, pp.11551157, 2010
[6] Mazzia A., Juett J.; *Suggesting Hashtags on Twitter*; In EECS 545m, Machine Learning, Computer Science and Engineering, University of Michigan
[7] Yang Y., Liu X.; *A re-examination of text categorization methods*; Carnegie Mellon University; Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, pp42-49, 1999
[8] 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. 4:14:10, ACM, 2010.
[9] S. P. Kasiviswanathan, P. Melville, A. Banerjee, and V. Sindhwani, *Emerging topic detection using dic- tionary learning*. in Proceedings of the 20th ACM International Conference on Information and Knowledge Management, CIKM 11, (New York, NY, USA), pp. 745754, ACM, 2011.
[10] 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. 693702, ACM, 2012.
[11] Lee K., Palsetia D., Narayanan R.; *Twitter Trending Topic Classification*; Northwestern University, Evanston; Proceedings of the 2011 IEEE 11th International Conference on Data Mining Workshops,pp.251-258, 2011
[12] Sankaranarayanan J., Samet H., Teitler B. E., Lieberman M. D., Sperling J.; *TwitterStand: News in Tweets*; Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp.42-51, 2009
[13] Homem N., Carvalho J.; *Authorship Identification and Author Fuzzy Fingerprints*; Proc. of the NAFIPS2011 - 30th Annual Conference of the North American Fuzzy Information Processing Society, 2011, IEEE Xplorer
[14] Homem, N. Carvalho, J.P.; *Finding top-k elements in data streams*; Information Sciences, 180(24), pp. 4958-4974, Dec. 2010, Elsevier
[15] Homem, N. Carvalho, J.P.; *Finding top-k elements in a time-sliding window*; Evolving Systems, 2(1), pp. 51-71, Jan. 2011, Springer
[16] Carvalho, J.P., Coheur, L.; *Introducing UWS A Fuzzy Based Word Similarity Function with Good Discrimination Capability: Preliminary results*; Proc. of the FUZZ-IEEE 2013, Hyderabad, India, 2013, IEEE Xplorer
[17] Feldman R., Sanger J.; *The Text Mining Handbook*; Cambridge University Press, 2007
[18] Rajaraman, A., Ullman, J.D.; *Mining of Massive Datasets*; Cambridge University Press, 2011
[19] Kondachy M.; *Text Mining Application Programming*; Charles River Media, 2006
[20] https://dev.twitter.com/
[21] *https://dev.twitter.com/discussions/6789*
[22] *WEKA*; http://www.cs.waikato.ac.nz/ml/weka/
[23] Zang J., Inderjeet M.; *kNN Approach to Unbalanced Data Distributions: A Case Study involving Information Extraction*
[24] Carvalho,J.P., Pedro,V.C., Batista,F.; *Towards intelligent mining of public social networks' influence in society*; IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS), 2013, 478-483
[25] Twitter Team; *Twitter Turns Six*; Twitter Blog (blog of Twitter), March, 21, 2012