One Day in Twitter: Topic Detection Via Joint Complexity

Gérard Burnside Dimitris Milioris Philippe Jacquet

Bell Labs, Alcatel-Lucent, Centre de Villarceaux, 91620 Nozay, France
{gerard.burnside, dimitrios.milioris, philippe.jacquet}@alcatel-lucent.com

Abstract

In this paper we introduce a novel method to perform topic detection in Twitter based on the recent and novel technique of Joint Complexity. Instead of relying on words as most other existing methods which use bag-of-words or n-gram techniques, Joint Complexity relies on String Complexity which is defined as the cardinality of a set of all distinct factors, subsequences of characters, of a given string. Each short sequence of text is decomposed in linear time into a memory efficient structure called Suffix Tree and by overlapping two trees, in linear or sublinear average time, we obtain the Joint Complexity defined as the cardinality of factors that are common in both trees. The method has been extensively tested for Markov sources of any order for a finite alphabet and gave good approximation for text generation and language discrimination. One key take-away from this approach is that it is language-agnostic since we can detect similarities between two texts in any loosely character-based language. Therefore, there is no need to build any specific dictionary or stemming method. The proposed method can also be used to capture a change of topic within a conversation, as well as the style of a specific writer in a text. In this paper we exploit a dataset collected by using the Twitter streaming API for one full day, and we extract a significant number of topics for every timeslot.

In: S. Papadopoulos, D. Corney, L. Aiello (eds.): Proceedings of the SNOW 2014 Data Challenge, Seoul, Korea, 08-04-2014, published at http://ceur-ws.org

1 Introduction

During recent years the online social media services have seen a huge expansion, and as a result, the value of information within these networks has increased dramatically. Interactions and communication between users help predict the evolution of information in online social networks, and the ability to study these networks can provide relevant information in real time.
The study of social networks has several research challenges:

- searching in blogs, tweets and other social media is still an open problem since posts are individually small in size but there is a tremendous quantity of them delivered in real time, with little and sometimes transient contextual information. Real time search has to balance between quality, authority, relevance and timeliness of the content,
- the information of the correlation between groups of users can help predict media consumption, network resources and traffic. This can be used in order to improve the quality of service and experience,
- analyzing the relationship between members of a group or a community within a social network can reveal the important teams in order to use them for companies marketing plans,
- spam and advertisement detection is another important challenge, since with the growth of social networks we also have a continuously growing amount of irrelevant information over the network.

In order to address these challenges, we need to extract the relevant information from online social media in real time.

In this paper we use the theory of Joint Complexity to detect bursty trends among short pieces of information. Our method is simple, context-free, with no grammar and no language assumptions, and it does not use semantics.

The Suffix Tree contains the algorithmic signature of the text, and we use that to have a fast and massive – without human intervention – trend sensing in social communities.

In a prior work, we proposed a method based on Joint Complexity along with pattern matching on URLs extracted from tweets. That method was compared with a version of a Document-Pivot method described in [APM+13], which was modified in order to perform classification of tweets, *i.e.* assigning tweets to categories such as Sports, Politics, Economics, etc. The results showed a good performance of the method compared to Document-Pivot and motivated us to use a more simplified version of it based on Joint Complexity for the Snow Data Challenge 2014.

The paper is organized as follows: Section 2 introduces the Joint Complexity method, while Section 3 briefly discusses the Snow Data Challenge dataset. Section 4 describes the implementation of the proposed method for topic detection in Twitter, while Section 5 evaluates the performance with real data obtained from the mentioned dataset. Finally, Section 6 summarizes our main results and provides directions for future work.

2 Joint Complexity

In the last decades, several attempts have been made to mathematically capture the concept of complexity of a sequence, i.e, the number of distinct factors contained in a sequence. In other words, if X is a sequence and $I(X)$ its set of factors, then $|I(X)|$ is the complexity of the sequence. For example, if X = "*apple*" then $I(X) = \{$a, p, l, e, ap, pp, pl, le, app, ppl, ple, appl, pple, apple, $v\}$ and $|I(X)| = 15$ (v denotes the empty string). The complexity of a string is also called the $I-$complexity [BH11]. The notion is connected with deep mathematical properties, including the rather elusive concept of randomness in a string [IYZ02, LV93, Nie99].

In general, the information contained in a string may be revealed by comparing with a reference string, since its entropy or I-complexity does not give much insight. In a previous work [Jac07] we introduced the concept of *Joint* Complexity, or J-complexity, of two strings. The J-complexity is the number of common distinct factors in two sequences. In other words the J-complexity of sequence X and Y is equal to $J(X,Y) = |I(X) \cap I(Y)|$.

The J-complexity is an efficient way to estimate similarity degree of two sequences. Two texts written in similar languages have more sequences in common than texts written in very different languages. Furthermore, texts in the same language but on different topics have smaller Joint Complexity than texts on the same topic.

The analysis of a sequence in subcomponents is done by Suffix Trees, which come with a simple, fast and low complexity method to store and recall them from memory, especially for short sequences. Suffix Trees are frequently used in DNA sequence analysis. The construction of the Suffix Tree for the words *apple* and *maple*, as well as the comparison between them is shown in Fig. 1.

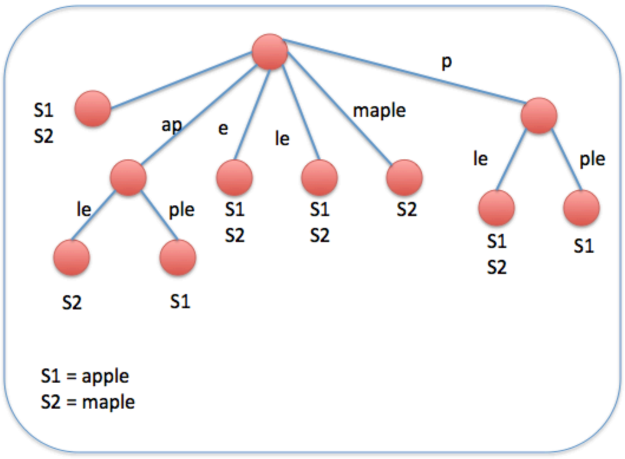

Figure 1: Suffix Tree of words *apple*, *maple*. $JC(apple, maple) = 9$.

In our previous work [Jac07] it is proved that the J-complexity of two texts of size m built from two *different* binary memoryless sources grows like

$$\gamma \frac{m^{\kappa}}{\sqrt{\alpha \log m}}, \tag{1}$$

for some $\kappa < 1$ and $\gamma, \alpha > 0$ which depend on the parameters of the two sources. When the sources are identical, then the J-complexity growth is $O(m)$, hence $\kappa = 1$. When the texts are identical (i.e, $X = Y$), then the J-complexity is identical to the I-complexity and it grows as $\frac{m^2}{2}$ [JLS04]. Indeed the presence of a common factor of length $O(m)$ would inflate the J-complexity by a term $O(m^2)$.

We should point out that experiments show that the complexity estimated as above for memory-less sources converges very slowly. Furthermore, memory-less sources are not appropriate for modeling text generation. In a prior work, we extend the J-complexity estimate to Markov sources of any order on a finite alphabet. Markov models are more realistic and have better approximation for text generation, *i.e.* for DNA sequences, than memory-less sources [JMS13, MJ14].

Joint string complexity has a variety of applications, such as detection of the similarity degree of two sequences, for example the use of "copy-paste" in texts or documents or the identification of authors and era of texts. We want to show here that it can also be used in analysis of social networks (*e.g.* tweets which are limited to 140 characters) and classification. Therefore it may be a pertinent tool for automated monitoring of social networks. Real time search in blogs, tweets and other social media has to balance between quality and relevance of the content, which due to the very short size of the texts and the huge amount of those, is still an unsolved problem.

In a prior work [JMS13, MJ14] we derive a second order asymptotics for J-complexity of the following form

$$\gamma \frac{m^\kappa}{\sqrt{\alpha \log m + \beta}}, \tag{2}$$

for some $\beta > 0$. This new estimate converges more quickly, and usually works for texts of order $m \approx 10^2$; In fact, for some Markov sources our analysis indicates that J-complexity oscillates with m. This is outlined by the introduction of a periodic function $Q(log\ m)$ in the leading term of our asymptotics. This additional term even further improves the convergence for small values of m.

In view of these facts, we can use the J-complexity to discriminate between two identical/non-identical Markov sources [Ziv88]. We introduce the discriminant function as follows

$$d(X,Y) = 1 - \frac{1}{\log m} \log J(X,Y), \tag{3}$$

for two sequences X and Y of length m. This discriminant allows us to determine whether X and Y are generated by the same Markov source or not by verifying whether

$$\begin{aligned} d(X,Y) &= O(1/\log m) \to 0 \ or \\ d(X,Y) &= 1 - \kappa + O(\log\log m/\log m) > 0, \end{aligned}$$

respectively when the length of X and Y is equal to m.

In [JMS13, MJ14] we show some experimental evidence of usage of our discriminant function on real and simulated texts by different Markov orders. We compare the J-complexity of a simulated English text with same length texts generated in French, Polish, Greek and Finnish by a third order Markov model to our theoretical results. Even for texts of lengths smaller than a thousand characters, one can easily discriminate between these languages. Therefore, Joint Complexity is an efficient method to capture the similarity degree of short texts.

3 Snow Data Challenge Dataset

According to the Social Sensor dataset for the Snow Data Challenge, we collected tweets for 24 hours; between Tuesday Feb. 25, 18:00 and Wednesday Feb. 26, 18:00 (GMT). The result of crawling was over 1,041,062 tweets between the Unix timestamps 1393351200000 and 1393437600000 and was constructed through the use of the *Twitter* Streaming API by following 556,295 users and also looking for four specific keywords: *Syria; terror; Ukraine; bitcoin.* The dataset used for the experiments consists of 96 timeslots, where each timeslot contains tweets for every 15 minutes, starting at 18:00 on Tuesday 25th 2014. The challenge then consisted in providing a minimum of one and a maximum of ten different topics per timeslot, along with a headline, a set of keywords and a URL of a relevant image for each detected topic. The test dataset activity and the statistics of the dataset crawl are described more extensively in [PCA].

4 Topic Detection Method

Until the present, the main methods used for text classification are based on *keywords* detection and Machine Learning techniques. Using keywords in tweets has several drawbacks because of wrong spelling or distorted usage of the words – it also requires lists of stop-words for every language to be built – or because of implicit references to previous texts or messages. Machine Learning techniques are generally heavy and complex and therefore may not be good candidates for real time text processing, especially in the case of Twitter where we have natural language and thousands of tweets per second to process. Furthermore, Machine Learning processes have to be manually initiated by tuning parameters, and it is one of the main drawbacks for the kind of application, where we want minimum if any human intervention. Some other methods are using information extracted by visiting the specific URLs on the text, which makes them a heavy procedure, since one may have limited or no access to the information, *e.g.* because of access rights, or data size/rate.

In our method we use an analysis of the messages (tweets) based on Suffix Trees [THP04] and we use the Joint Complexity as a metric to quantify the similarity between these messages.

According to the dataset described in Section 3 and in [PCA] we have $N = 96$ timeslots with $n = 1 \dots N$. For every tweet t_i, where $i = 1 \dots M$, with M being the total number of tweets, in the n-th timeslot, $i.e\ t_i^n$, we build a Suffix Tree, $ST(t_i^n)$, as described in Section 2. Building a Suffix Tree is an operation that costs $O(m \log m)$ and takes $O(m)$ space in memory, where m is the length of the tweet.

Then we compute the Joint Complexity metric, $JC(t_i^n, t_j^n)$ of the tweet t_i^n with every other tweet t_j^n of the n-th timeslot, where $j = 1 \dots M$, and $j \neq i$ (by convention we choose $JC(t_i^n, t_i^n) = 0$). The Joint Complexity between two tweets is the number of the common factors defined in language theory and can be computed efficiently in $O(m)$ operations (sublinear on average) by Suffix Tree superposition. For the N timeslots we store the results of the computation in the matrices $T_1, T_2, \dots, T_N$ of $M \times M$ dimensions.

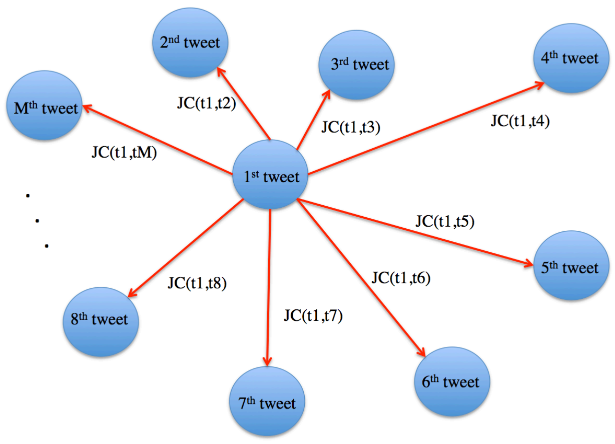

Figure 2: Representation of the first row of the n-th timeslot via weighted graph.

We represent timeslots by fully-connected weighted graphs. Each tweet is a node in the graph and the two-dimensional array T_n holds the weight of each edge, as shown in Fig. 2. Then, we calculate the score for each node in our graph by summing all the edges that are connected to the node. The node that gives the highest score is the most representative and central tweet of the timeslot.

$$T_n = \begin{pmatrix} 0 & JC_{t_1^n,t_2^n} & JC_{t_1^n,t_3^n} & \cdots & JC_{t_1^n,t_M^n} \\ JC_{t_2^n,t_1^n} & 0 & JC_{t_2^n,t_3^n} & \cdots & JC_{t_2^n,t_M^n} \\ JC_{t_3^n,t_1^n} & JC_{t_3^n,t_2^n} & 0 & \cdots & JC_{t_3^n,t_M^n} \\ \vdots & \vdots & & \ddots & \vdots \\ JC_{t_M^n,t_1^n} & JC_{t_M^n,t_2^n} & JC_{t_M^n,t_3^n} & \cdots & 0 \end{pmatrix}$$

Most of the timeslots have $M = 5,000$ tweets, so matrices $T_1, T_2, \dots, T_N$ have approximately $25,000$ entries for every timeslot. Since they are symmetric, only half of these entries could be used, *i.e* the upper triangular or the lower triangular of matrices $T_1, T_2, \dots, T_N$, as shown below, which save space and make the structure more efficient.

Algorithm 1 Method to retrieve row i of an upper triangular matrix

```
// data is the internal TIntArrayList object
int[] row = new int[data.length+1];
// read the k-th column up to i:
for $k = 0$ to $i - 1$ do
    $row[k] \leftarrow data[k].get(i - k - 1)$;
end for
$row[i] \leftarrow 0$; // by convention, not computed
if $i < data.length - 1$ then
    // read the i-th row until the end:
    for $j = 0$ to $data.length - i$ do
        $row[i + j + 1] \leftarrow data[i].get(j)$;
    end for
else do nothing; // the last tweet does not have a row!
end if
return row;
```

$$T_n^{up\ triang} = \begin{pmatrix} \mathbf{JC_{t_1^n,t_2^n}} & \mathbf{JC_{t_1^n,t_3^n}} & \cdots & \mathbf{JC_{t_1^n,t_M^n}} \\ & \mathbf{JC_{t_2^n,t_3^n}} & \cdots & \mathbf{JC_{t_2^n,t_M^n}} \\ & & \ddots & \vdots \\ & & & \mathbf{JC_{t_{M-1}^n,t_M^n}} \end{pmatrix}$$

The actual implementation of the upper triangular data structure is based on an efficient *TIntArrayList* data structure provided by the *Trove4j* library (it uses arrays of int instead of Java objects) encapsulated in a class which provides methods to access individual elements thanks to their row and column index or a whole row by providing its index. Algorithm 1 illustrates the *getRow*() method of this class.

The whole Cross Joint Complexity computation was run in a multithreaded way on a 24 processor machine: $k = 23$ threads are started and each thread works on a disjoint set of rows. Note that because each row of the triangular matrix does not have the same number of elements, the number of rows assigned to each thread should not be divided evenly, otherwise the first threads would have much more work to do than the last ones (remember that the input is an upper triangular matrix); instead, because the total number of non-zero elements of the $n*n$ matrix is $n(n-1)/2$, each thread should work on approximately $n(n-1)/(2*k)$ elements so the implementation kept assigning rows to a thread until that number of elements was reached.

This implementation allowed the program to run in on average 95 seconds in order to compute a 15-minutes timeslot.

These computations were only run once, as soon as the data was properly divided into 15-minutes timeslots, and the results were saved in files which were subsequently used to perform the rest of implementation.

When we finally get the scores of the Joint Complexity metric, we try to find the R most representative and central tweets of every timeslot. At first we get the sum of the Joint Complexity, $S_i^n = \sum_{j=1\ldots M, j\neq i} JC_{t_i^n, t_j^n}$, of the i-th tweet with every other tweet j in the n-th timeslot, and finally we get the vector $S^n = [S_1^n, S_2^n, \ldots, S_M^n]$ for all timeslots.

We sort the elements of each vector S^n in descending order and we get the R most representative and central tweets in the following way: The best-ranked tweet is chosen unconditionally, the second one is picked only if its JC score with the first one is below a chosen threshold Thr_{low}, otherwise it is added to the list of related tweets of the first tweet; similarly, the third one is picked only if its JC score with the first two is below Thr_{low}, etc. This ensures that the topics are dissimilar enough and it classifies best ranked tweets into topics at the same time.

Then by removing punctuation, special characters, etc., of each selected tweet, we construct the headlines of each topic and we run through the list of related tweets to keep only tweets that are different enough from the selected one (we do not want duplicates), we do so by keeping only the tweets whose JC score with the selected tweet and all previous related tweets is above a chosen threshold Thr_{max}. We first chose empirical values 400 and 600 for Thr_{low} and Thr_{max} respectively, then a few hours before submission time we noticed that many topics had only one related tweet (all the others were retweets), so we decided to lower that threshold to 240 but did not have time to recompute the results on the whole dataset so only a handful of timeslots benefited from this better $Thr_{low} = 240$ value. The final plan is to come up with a formula to have the system determine those thresholds automatically depending on the number of characters of each tweet.

While running through the list of related tweets we compute the bag of words used to construct the list of keywords and we also check the original $json$ data to find a URL pointing to a valid image related to the topic.

We chose to print the first top 8 topics for each timeslot.

4.1 Keywords

In the bag of words constructed from the list of related tweets, we remove articles (stop-words), punctuation, special characters, etc., and we get a list of words, and then we order them by decreasing frequency of occurrence. Finally we report the k most frequent words, in a list of keywords $K = [K_{1\ldots k}^1, K_{1\ldots k}^2, \ldots, K_{1\ldots k}^N]$, for the N total number of timeslots.

4.2 Media URLs

The body of a tweet (in the *.json* file format), contains a URL information for links to media files such as pictures or videos, when available this information is stored in the following subsection of the $json$ object: *entities* $\rightarrow$ *media* $\rightarrow$ *media_url*.

While reporting the most representative and central tweets, we scan the original json format in order to retrieve such a URL, from the most representative tweet or any of its related tweets, pointing to valid photos or pictures in a *.jpg*, *.png* or *.gif* format. Then, we report these pictures along with the headlines and the set of keywords, as shown in Algorithm 2 .
Almost half of the headlines (47%) produced by our method had an image retrieved from the original tweet. When no such URL is provided within the collected json objects we planned to visit the URLs of websites and retrieve images from the website but did not have sufficient time to implement this in a way that enforced that the retrieved image was indeed relevant for the topic. It is important to point out that the goal of the challenge was to detect newsworthy items before they hit mainstream news websites, so it was decided that parsing images from such websites was not interesting in that context.

5 Evaluation

Twitter is one of the most popular social networks and micro-blogging service in the world, with more than 540 million users connected by 24 billion links [GL12]. It allows the information spread between people, groups and advertisers, and since the relation between its users is unidirectional, the information propagation within the network is similar to the way that the information propagates in real life.

Apart from the specific implementation for the Snow Data Challenge, the main benefits of our method are that we can both classify the messages and identify the growing trends in real time, without having to manually identify lists of keywords for every language. We can track the information and timeline within a social network and find groups of users that agree or have the same interests, *i.e*, perform trend sensing.

The official evaluation results of our method in the Data Challenge are included in [PCA], therefore here we will only discuss a quick comparison of our method with simply counting the number of retweets within a chosen timeslot, namely Feb.

Algorithm 2 Topic detection based on Joint Complexity

```
// $N$ = # timeslots, $M$ = # tweets in the n-th timeslot
for $n = 1$ to $N$ do
    for $t = 1$ to $M$ do
        $t \leftarrow t_{json}.getText();$
        $t_{ST} \leftarrow suffixTreeConstruction(t);$
        $JCScores \leftarrow JCMetric();$
    end for
    // Find the most representative & central tweets
    $S^n \leftarrow sum(JCScores);$
    // Get headlines for the central tweets
    $R^n \leftarrow descendingOrder(S^n);$
    // Get set of keywords
    $K^n \leftarrow keywords(R^n);$
    // Get URLs of pictures from the .json file
    $P^n \leftarrow mediaURL(R^n);$
    // Print the results in appropriate format
    $Print(R^n);$
    $Print(K^n);$
    $Print(P^n);$
end for
```

25th from 8:15 pm until 8:30 pm, comprised of just under 16,000 tweets, which is about 50% higher than the average number of tweets for the whole 24h period.

The ranking produced by our method for the timeslot 25-02-14 20:15 is listed below:

```
(JC1) BarackObama: LIVE: President Obama is
announcing a new plan to boost job growth
for middle-class Americans.

(JC2) WhiteHouse: "I'm here to announce that
we're building Iron Man | Not really. Maybe.
It's classified." President Obama ActOnJobs

(JC3) Madonna: Fight for the right to be free!
Fight Fascism and discrimination everywhere!
Free Venezuela the Ukraine...

(JC4) Karinabarbosa: civicua: Free Venezuela !
Ukraine is with you!  euromaidan Photo by
Liubov Yeremicheva

(JC5) TheEllenShow: I'm very excited Pharrell's
performing his big hit "Happy" at the Oscars.
Spolier alert: I'll be hiding in his hat.

(JC6) BBCSport: Olympiakos take the lead vs
Manchester United, a hideous deflection off
Alejandro Dominguez beats De Gea.

(JC7) WhatTheBit: It's going to be pretty damn
near impossible to top this as photo of the week.

(JC8) Al Qaeda branch in Syria issues ultimatum
to splinter group: The head of an al
Qaeda-inspired militia fighting...
```

A small program was written to compare the number of retweets observed on the first occurence of a retweet and substract that to the number of retweets observed on the last occurence of the same retweet. All tweets were then sorted in descending order of the number of retweets obtained, and the result is shown below:

```
(RT1) RT @OllieHolt22: Ashley Young berating
someone for going down too easily - that's funny

(RT2) RT @WhiteHouse: "I'm here to announce that
we're building Iron ManNot really. Maybe. It's
classified." @President Obama #ActOnJobs

(RT3) RT @BarackObama: LIVE: President Obama is
announcing a new plan to boost job growth for
middle-class Americans. http://t.co/d6x1Bous1S

(RT4) RT @TheEllenShow: I'm very excited @Pharrell's
performing his big hit "Happy" at the #Oscars.
Spolier alert: I'll be hiding in his hat.

(RT5) RT @Madonna: Fight for the right to be
free! Fight Fascism and discrimination everywhere!
Free Venezuela the Ukraine...

(RT6) RT @civicua: Free #Venezuela !  #Ukraine is
with you! #euromaidan Photo by Liubov Yeremicheva

(RT7) RT @russellcrowe: Holy Father @Pontifex , it
would be my deepest pleasure to bring the
@DarrenAronofsky film to you to screen. [...]

(RT8) RT @DjKingAssassin: $myry recovering
on #bitcoin market update on #mtgox
```

The following differences can be observed between the two rankings:

- The best ranked retweet does not appear in the list produced by our method! In fact a plausible explanation for $RT1$ and $RT8$ not being picked up by our method is a weak centrality due to a shorter size of text, as briefly explained in Section 6. For $RT7$ however the size of the text is not a factor and the weak centrality is most probably explained by the unlikely co-occurrence of terms within the sentence. Although this is highly subjective it is worth noting that those tweets do not appear to be very newsworthy.

- Five out of eight items produced by our method ($JC1$ to $JC5$) are listed in the retweet ranking, this is rather expected as there is a mechanical link between our method and the number of occurrences of the same text.
- The eighth item produced by our method ($JC8$) seems to be very newsworthy but appears only past the 100^{th} position in the retweet count method. Again, although this is subjective it appears to be an interesting result.

Finally, although the dataset that was used for this challenge did not allow to show this properly, the authors strongly believe that one key advantage of using Joint Complexity is that it can deal with languages other than English [JMS13, MJ14] without requiring any additional human intervention.

6 Conclusion and Future Work

In this short paper we presented an implementation of a topic detection method applied to a dataset of tweets emitted during a 24 hour period. The implementation relies heavily on the concept of Joint String Complexity which has the benefit of being language agnostic and not requiring humans to deal with list of keywords. The results obtained seemed satisfactory and a final evaluation in the context of the SNOW Data Challenge 2014, can be found in [PCA].

As the timeframe available to participate in the challenge was quite short, a - probably inexhaustive - number of items have been identified as possible future work, and these are listed below:

- As mentioned in Section 4, although the current empirical values chosen as thresholds to determine whether two texts are similar and whether two texts are near-duplicates seemed to be quite satisfactory on the dataset that our method was tested against, it would be more useful to compute values that would adjust to the size of the texts.
- The current implementation systematically picks the top 8 topics from each timeslot whereas a better approach might be to identify a threshold under which topics may be considered not significant enough, and thus allowing some not very active timeslots to contain less than 8 topics while other very active timeslots may contain more than 8.
- Dividing the timeslots every quarter of an hour is arbitrary as some topics may be cut in half if they started after the middle of the previous timeslot and may therefore not acquire enough significance on the current timeslot. Before aggregating the tweets into 15-minutes timeslots, our implementation first divided the data into 5-minutes timeslots and we considered building 20 minutes timeslots (each time including the 5 minutes preceding the official 15-minutes slot) to remedy the above issue, but this is now left as future work to check if it provides significant improvements.
- The current implementation which sorts the topics by finding the most central tweets is a bit unfair to shorter tweets because longer tweets have mechanically a better chance of obtaining a higher centrality score (Joint Complexity counts the number of common factors), therefore a future work should be to somehow mitigate this. One idea may be to modify the Joint Complexity metric in order for it to behave like a distance and to perform clustering based on this distance.

Acknowledgment

This work is supported by the SocialSensor and REVEAL FP7 project under contract number 287975 and 610928 respectively. Dimitris Milioris would like to thank INRIA, École Polytechnique ParisTech and LINCS.

References

[APM+13] L. Aiello, G. Petkos, C. Martin, D. Corney, S. Papadopoulos, R. Skraba, A. Goker, I. Kompatsiaris, and A. Jaimes. Sensing trending topics in twitter. 15(6):1268–1282, October 2013.

[BH11] V. Becher and P. A. Heiber. A better complexity of finite sequences. *8th Int. Conf. on Computability and Complexity in Analysis and 6th Int. Conf. on Computability, Complexity, and Randomness*, page 7, 2011.

[GL12] M. Gabielkov and A. Legout. The complete picture of the twitter social graph. In *ACM CoNEXT Student Workshop*, 2012.

[IYZ02] L. Ilie, S. Yu, and K. Zhang. Repetition complexity of words. pages 320–329, 2002.

[Jac07] P. Jacquet. Common words between two random strings. In *IEEE International Symposium on Information Theory*, Nice, France, 2007.

[JLS04] S. Janson, S. Lonardi, and W. Szpankowski. On average sequence complexity. pages 213–227, 2004.

[JMS13] P. Jacquet, D. Milioris, and W. Szpankowski. Classification of markov sources through joint string complexity: Theory and experiments. 2013.

[LV93] M. Li and P. Vitanyi. *Introduction to Kolmogorov Complexity and Its Applications*. Springer–Verlag, Berlin, August, 1993.

[MJ14] D. Milioris and P. Jacquet. Joint sequence complexity analysis: Application to social networks information flow. *Bell Laboratories Technical Journal*, 18(4), 2014. Issue on Data Analytics.

[Nie99] Harald Niederreiter. Some computable complexity measures for binary sequences. In C. Ding, T. Helleseth, and H. Niederreiter, editors, *Sequences and their Applications*, Discrete Mathematics and Theoretical Computer Science, pages 67–78. Springer London, 1999.

[PCA] S. Papadopoulos, D. Corney, and L. Aiello. Snow 2014 data challenge: Assessing the performance of news topic detection methods in social media. In *Proceedings of the SNOW 2014 Data Challenge*.

[THP04] S. Tata, R. Hankins, and J. Patel. Practical suffix tree construction. In *Proceedings of the 30th VLDB Conference*, 2004.

[Ziv88] J. Ziv. On classification with empirically observed statistics and universal data compression. *IEEE Transactions on Information Theory*, 34:278–286, 1988.