International Journal on Natural Language Computing (IJNLC) Vol.11, No.5, October 2022

Hierarchical topics in texts generated by a stream

Achraf Lassoued

University of Paris II and IRIF-CNRS

Abstract. We observe a stream of text messages, generated by Twitter or by a text file and present a tool which constructs a dynamic list of topics. Each tweet generates edges of a graph where the nodes are the tags and edges link the author of the tweet with the tags present in the tweet. We consider the large clusters of the graph and approximate the stream of edges with a Reservoir sampling. We study the giant components of the Reservoir and each large component represents a topic. The nodes of high degree and their edges provide the first layer of a topic, and the iteration over the nodes provide a hierarchical decomposition. For a standard text, we use a Weighted Reservoir sampling where the weight is the similarity between words given by Word2vec. We consider dynamic overlapping windows and provide the topicalization on each window. We compare this approach with the Word2content and LDA techniques in the case of a standard text, viewed as a stream.

Keywords: NLP, Streaming algorithms, Clustering, Dynamic graphs.

1 Introduction

We observe streams of high volume of incoming text and study the classification and the sentiment analysis online, without storing the entire data. We consider messages generated by social network or by text servers. Twitter, for example, generates streams of tweets which are transformed into streams of graph edges where the nodes are the tags and edges link the author of the tweet with the tags present in the text. Text servers generate text sentences which are transformed into graph edges where the nodes are the words and the edges link two words of the same sentence.

We study the large clusters of these graphs in sliding windows by sampling a fixed number of edges with two different distributions, the *uniform distribution* or a *weighted distribution*. The analysis is based on the study of the *giant components* of these random subgraphs.

For Twitter applications, we typically receive 10^3 tweets per minute, approximately 3.10^3 edges per minute. We analyse random subgraphs online in sliding windows of length $\tau = 10$ minutes. We sample the edges *uniformly* for each sliding window using a *Reservoir sampling* [23], and analyse the giant components of the Reservoir of constant size K. We only keep the giant components of each window. We interpret the giant components as *topics* and follow the various topics in time. We can also correlate different streams, based on the comparison of their giant components.

For text applications, an RSS-stream of news articles may generate a large stream of sentences. The nodes of the generated graph are the words and an edge is a pair of words in the same sentence. If we sample uniformly the edges, the giant components are however not stable. We observed that if we sample the edges proportionally to the similarity of the words, given by Word2vec [14], the giant components become stable. We use the classical k-means algorithm to classify the text, with the Jaccard distance between giant components. Our main contributions are:

– an analysis of streaming texts by giant components of random graphs: *the uniform sampling* is efficient for Social media such as Twitter, whereas the *weighted sampling*

DOI: 10.5121/ijnlc.2022.11503

where the weight is the Word2vec similarity between words is efficient for streaming texts,
- a classification technique, based on a natural distance between components, useful for topicalization and an analysis of the nodes of high degree of the giant components which define the *central words* and *central sentences* of the text.
- the definition of a *sentiment index* associated with a text within a text reference L, the basis for a sentiment analysis.

In section 2, we introduce the main concepts. In section 3 we define the giant components in random graphs and the sampling methods. In section 4, we define a distance between giant components and use the k-means for classification. In section 5, we describe how detect offensive messages which have a high impact. In section 6, we describe experiments for analysing Twitter streams and for classifying and analysing standard texts.

2 Preliminaries

We first review Streaming algorithms, Social Networks and some of the classical statistical methods used in Natural language processing.

2.1 Streaming algorithms

These algorithms read data step by step and maintain a small memory, if possible constant, $poly(\log)$ or at least sublinear in the size of the stream. Classical algorithms may consider a stream of numerical values $x_i \in \{1, 2, ...n\}$, of words on an alphabet Σ, or of edges $e_i = (v_j, v_k)$ of a graph $G = (V, E)$ where $v_j, v_k \in V$.

An important technique called the *Reservoir sampling* [23] keeps K elements of the stream with a *uniform distribution*. In a stream of length m each element has probability K/m to be chosen in the Reservoir. The *weighted Reservoir sampling* keeps K elements of the stream with a *weighted distribution*, detailed in the appendix A. Each element e_i of weight w_i of the stream has probability $K \cdot w_i / \sum_i w_i$ to be chosen in the Reservoir. If K is sublinear, for example $O(\sqrt{m})$, we obtain a sublinear space algorithm.

2.2 Social Networks

In social networks and crowdsourcing we observe large streams of data online, mostly edges $e_1, ..., e_m$ of a graph. Given a set of tags such as {#Ethereum, #Bitcoin}, or {#Amazon}, Twitter provides a stream of tweets represented as Json trees whose content C (the text of the tweet) contains at least one of these tags. The *Twitter Graph* of the stream is the graph $G = (V, E)$ with multiple edges E where V is the set of tags $\#x$ or $@y$ and for each tweet sent by $@y$ which contains tags $\#x$,$@z$ we add the edges $(@y, \#x)$ and $(@y, @z)$ in E. In our approach, we consider the hypergraph where we add the content C to each edge. We have then the hyperedges $(@y, \#x, C)$ and $(@y, @z, C)$. The URL's which appear in the tweet can also be considered as nodes but we ignore them for simplicity. A stream of tweets is then transformed into a stream of edges $e_1, ...e_m, ...$, although each edge is an hyperedge, which also stores a timestamp.

Social networks such as Twitter evolve dynamically, and dense subgraphs appear and disappear over time as interest in particular events grows and disappears.

2.3 Large dense subgraphs

There are several appoaches to density in graphs with n nodes and m edges.

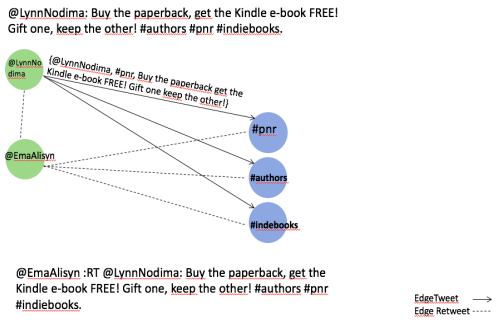

Fig. 1. Twitter graph for a tweet and a retweet

3 Large dense subgraphs

Let $S \subseteq V$ and $E(S)$ be the set of internal edges i.e. edges $e = (u, v)$ where $u, v \in S$. The classical density of S is the ratio $\rho = |E[S]|/|S|$. One may want to find subgraphs with S nodes which maximize ρ. In the case of a stream of edges, the approximation of dense subgraphs is well studied in [13] and an $\Omega(n)$ space lower bound is known [3]. In [1], another different objective is considered: a γ-*cluster* of domain S is a *subgraph* which depends on a parameter $\gamma \leq 1$ such that $|E(S)| \geq \gamma.|S|.|S-1|/2$. This problem is also hard to approximate when S is large. We consider $|S| > \delta\sqrt{n}$ for some other parameter δ, hence the (γ, δ)-*large dense subgraph* problem.

A family of graphs G_n has a (γ, δ)-cluster if for n large enough there exists S such that S is a (γ, δ)-cluster for G_n. Classical community detection algorithms take the entire graph as input and apply spectral techniques. We consider a specific regime of graphs defined by a stream of edges $e_1, ..., e_m, ...$ which follow a power-law degree distribution μ and use a sublinear space algorithm. A Reservoir sampling [23] with $K = O(\sqrt{n}.\log n/\gamma \cdot \delta)$ edges can detect (γ, δ)-large dense subgraphs (clusters) with high probability [12], on the specific class of graphs taken from μ. We use a *one-sided stochastic randomized* algorithm A to detect the existence of a cluster:

- If G has a cluster , $Prob_{\Omega}[A(x)\ accepts] \geq 1 - \epsilon$
- If G is a random graph drawn from μ with no cluster, $Prob_{\mu \times \Omega}[A(x)\ rejects] \geq 1 - \epsilon$

We are mainly interested in large clusters S and assume that $|S| > \delta\sqrt{n}$ for some parameter δ. The (γ, δ)-*large dense subgraph* problem, where $\gamma \leq 1$, takes as input a graph $G = (V, E)$ and decides whether there exists an induced subgraph $S \subseteq V$ such that $|S| > \delta\sqrt{n}$ and $|E[S]| > \gamma|S|(|S|-1)/2$. Social graphs defined by a stream of edges $e_1, ..., e_m, ...$ follow a specific regime for which a Reservoir of the size $K = O(\sqrt{n}.\log n)$ can detect (γ, δ)-large dense subgraphs with high probability [21]. Appendix 3 details this approach.

3.1 Analysis of Natural languages

Classification methods in Natural Languages also consider the analysis of clusters. The IRaMuTeQ [20] method enables hierarchical classifications based on PCA (Principle Components Analysis). The Word2vec method [14] associates with the words, vectors of v small dimension. Other embeddings [10, 19] are also possible. Extensions to sentences are considered in Word2Sense [18].

The $LDA(k)$ (Latent Dirichlet Allocation) [4] is a probabilistic method to analyse the matrix A where the lines are the documents and the columns are the principal words, and the value $A(i,j)$ is the number of occurrences of the word j in the document i. It constructs k classes among n documents. LDA, Dynamic Topic Models [5] and IRaMuTeQ techniques require to access the entire data.

Attention mechanisms [11, 22] provide, for a word w_i in a sentence, the distribution of the other most correlated words. The correlation of a word w_j with w_i is approximately the value $v(w_i).v(w_j)$. We can then compute $\{v(w_i).v(w_j) : j \neq i\}$ for a fixed w_i and normalize the values to obtain a distribution. It is mostly used in Transformers for machine translation. We show that the weighted Reservoir can reconstruct the most significant attention distributions.

3.2 Sampling

A Reservoir sampling [23] keeps k edges from m streaming edges such that each edge has the probability k/m to be chosen in the Reservoir. It is an Erdös-Renyi model [9], written $G(n,p)$ with $p = k/m$. In the classical Erdös-Renyi model, we start with the complete graph, whereas we start with a social graph G_n.

Social graphs with m edges and n nodes follow a power law degree distribution, i.e. a Zipfian distribution where $Prob[d = j] = c/j^2$. The maximum degree is $\sqrt{c.n}$ and m is $O(cn\ln(n))$. The configuration model [6] studies random graphs with fixed degree distributions, such as the Zipfian law.

We therefore combine the two models: we start with a social graph G with a power law degree distribution and then sample it uniformly: it is the configuration model for the Zipfian law followed by the Erdös-Renyi model.

For classes of random graphs, a fundamental question is the existence of giant components[1] and phase transitions. In the Erdös-Renyi model $G(n,p)$, a giant component occurs if $p > 1/n$ i.e. when each edge of a clique is selected with probability p. If we generalize to a γ-cluster S, a giant component occurs if $p > 1/\gamma.|S|$. If the size of a cluster S is larger than $\delta.\sqrt{n}$ and the Reservoir size $k > \frac{c.\sqrt{n}.\log n}{\gamma.\delta}$, then:

$$\frac{k}{m} \geq \frac{c.\sqrt{n}.\log n}{\gamma.\delta.c.n\ln(n)} = \frac{1}{\gamma.\delta.\sqrt{n}} \geq \frac{1}{\gamma.|S|}$$

In this case we observe a giant component C in the Reservoir. Studies in [15] show that if we take a random graph with a power law degree distribution, there is no giant component with high probability. It is basis of the approach of [21] to detect clusters on a stream of edges without storing the entire graph but only only $k > \frac{c.\sqrt{n}.\log n}{\gamma.\delta}$ edges. They propose a detection algorithm which detects γ-clusters of size larger than $\delta.\sqrt{n}$ with high probability as giant components of the Reservoir. This approach can be generalized to dynamic windows as presented in [2, 8].

In the case of edges e with weights w_e, the *Weighted Reservoir sampling* keeps k edges with probability $\left(\frac{k.w_e}{\sum w_e}\right)$. In this case, there is no theoretical analysis for the existence of giant components, just experimental results.

3.3 Estimation of the clusters

Let 2-core(C) be the graph obtained from the connected component C by removing the nodes of degree 1 repeatedly[2]. Figure 3 gives the 2-core(C) of the giant component of the

[1] A giant component is a connected component larger than a constant fraction of n, the number of nodes.
[2] The k-core of a graph G is obtained by removing repeatedly all the notes of degree less than $k-1$

graph of Figure 7. It can be shown that 2-core(C) is a good approximation of a γ-cluster of size $\delta.\sqrt{n}$.

We store the 2-core(C) for each large C, as the witness of the clusters. Define

$$C = \bigcup_i 2 - core(C_i)$$

as the union of all the giant components C_i for a given stream.

3.4 Hierarchical decompositions

We explore the 2-core(C) starting with the nodes of maximum degree and iterate until we explored the whole connected component. With the graph of Figure 3, we obtain the tree decomposition in Figure 4. We start with the node 1 of maximum degree and explore all the nodes at distance 1. At the next stage the node 6 is of maximum degree and we obtain the node 3. The tree decomposition is of depth 2.

In Figure 5, each node is a tag and the text of the tweet is associated to each edge. We only show the text of edge e_1.

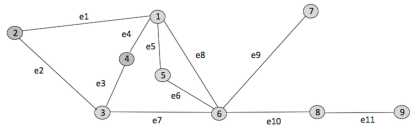

Fig. 2. The large connected component (C) in the Reservoir

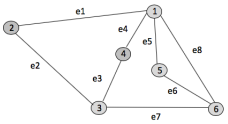

Fig. 3. The 2-core(C)

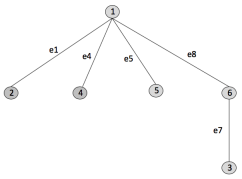

Fig. 4. Tree decomposition

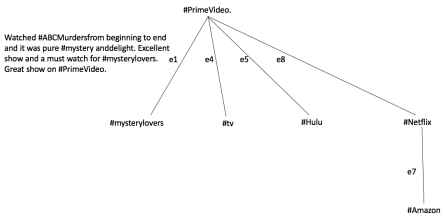

Fig. 5. Hiearchical tag tree

3.5 Dynamic clusters

At some discrete times $t_1, t_2, ...$, we store the large connected components of the Reservoirs R_t for dynamic sliding windows. A sliding window has two parameters: its time length τ and its time shift λ which divides τ, for example $\tau = 30$ mins and $\lambda = 10$ mins. There could be none. We use a NoSQL database, with 4 (Key, Values) tables where the key is always a tag ($@x$ or $\#y$) and the Values store the clusters nodes. Notice that a stream is identified by a tag (or a set of tags) and a cluster is also identified by a tag, its node of highest degree.

- *Stream(tag, list(cluster, timestamp)* is the table which provides the most recent clusters of a stream,
- *Cluster(tag, list(stream, timestamp, list(high-degree nodes), list(edges, tweets))))* is the table which provides the list of high-degree nodes and the list of nodes with their degree, in a given cluster,
- *Nodes(tag, list(stream, cluster, timestamp),tweet)* is the table which provides for each node the list of streams, clusters and timestamps where the node appears,

3.6 Stability

We can experimentally verify the existence of the giant components by performing independent experiments and measuring the stability. Consider two experiments on the same stream (or document D) with two *independent* Reservoirs. Let V_1 (resp. V_2) be the set of vertices of the giant component C_1 (resp. V_2). Let the *stability* of the stream D and Reservoir size K, be the random variable $\rho(D, k)$ defined as:

$$\rho(D, k) = \frac{|V_1 \cap V_2|}{|V_1|}$$

The random variable depends on two experiments and we can also take its expectation $\mathbb{E}(\rho(D, k))$ for n experiments. For Twitter streams, the uniform sampling provides a good stability, as suggested by the theoretical analysis of giant components. On the contrary, for texts the stability of the uniform sampling is close to 0.

4 Classification

We first introduce natural distances between giant components which can be generalized to sets of giant components. We can then apply the classical k-means algorithm.

4.1 Distances between giant components

Consider two giant components $C_1 = (V_1, E_1)$ at time t_1 and $C_2 = (V_2, E_2)$ at time $t_2 \geq t_1$ as labelled graphs. There several possible distances:

- The Jaccard distance $\mathsf{dist}_J(C_1, C_2) = 1 - J(V_1, V_2)$ where $J(V_1, V_2)$[3] is the Jaccard similarity between the domains V_1 and V_2.
- The Edit distance takes the edges into account. An *edition* is the deletion or insertion of an edge with labels and $\mathsf{dist}_E(C_1, C_2)$ is the minimum number of editions to transform C_1 into C_2, divided by $|E_1| + |E_2|$.
- The amortized Edit distance takes into account the time difference $|t_2 - t_1|$ and $\mathsf{dist}_A(C_1, C_2) = \mathsf{dist}_E(C_1, C_2) + \alpha \cdot |t_2 - t_1|$ for some constant α.
- The Matching distance: we take an α proportion of nodes v of highest degree of C, e.g. $\alpha = 10\%$ and we match them to the nodes $v' = \pi(v)$ of C' that maximize the product of the Word2vec vectors $u(v)$. The matching weight of $M(C_1, C_2, \alpha)$ is:

$$M(C_1, C_2, \alpha) = Max_\pi \ \ u(v).u(v')$$

  We take the Maximum over all possible matchings π. Notice that $W(C_1, C_2, \alpha)$ is asymmetric. We can make the matching weight symmetric , written $M_s(C_1, C_2, \alpha)$, by taking similarly the matching weight from C_2 to C_1 and finally the normalized sum of the two:

$$M_s(C_1, C_2, \alpha) = M(C_1, C_2, \alpha) + M(C_2, C_1, \alpha)$$

  Finally, the Matching distance $\mathsf{dist}_E(C_1, C_2) = 1 - M_s(C_1, C_2, \alpha)$.

Each sliding window may have several giant components of different sizes. In this case, we may write C as a distribution over the components with a weight proportional to their sizes. For example if C has two components C_1 and C_2, we write:

$$C = \frac{|C_1|}{|C_1| + |C_2|} \cdot C_1 + \frac{|C_2|}{|C_1| + |C_2|} \cdot C_2$$

We can then generalize the distances between components to distances between distributions of components. In the sequel, we only consider the largest giant component with the Jaccard distance.

4.2 k-classes

A sequence of sliding windows generates a sequence of giant components $C_1, ...C_n$ with all the distances between pairs. We can then group them in k-classes, with the classical k-means algorithm. Each class i has a representative C_i. For a new giant component C, we just check the Min_i $\mathsf{dist}(C_i, C)$ to classify C.

Let G_1 be the giant component of the stream on the tag #Amazon and G_2 be the giant component of the stream on the tag #CNN. Consider a new stream on the tag #Netflix.

[3] The Jaccard similarity or Index between two sets A and B is $J(A, B) = |A \cap B|/|A \cup B|$.

How do we classify it? We first compute the giant component G of the new stream and compute $J(V, V_1)$ and $J(V, V_2)$. As $J(V, V_1) \geq J(V, V_2)$, we classify the new stream as G_1, i.e. close to the #Amazon stream.

A tag common to V and V_1 is #Primevideo. A typical tweet on an edge on the node #Primevideo is:

@thomasjacksonjr: Watched #ABCMurders from beginning to end and it was pure #mystery and delight. Excellent show and a must watch for #mysterylovers. Great show on #PrimeVideo.

4.3 Comparaison with the classical topicalization

In a standard Word2Vec approach, let A be the correlation matrix where $A(i, j)$ is the number of occurrences of the pair (i, j) in the same sentence of a document. Suppose we sample this matrix proportionally to its weight, i.e. $Prob[s = (i, j)] = A(i, j)/B$ where B is the sum of the weight of A and s is a sample. We will have mostly frequent pairs (i, j) and a small proportion of unfrequent pairs. The Reservoir sampling realizes precisely this sampling, without storing the entire matrix A. The first giant component of the Reservoir is similar to the principal component analysis on the largest eigenvalue. Hence the method achieves a similar goal, but much more efficiently, as we proceed online.

Given a stream, we may observe several giant components in the Reservoir. Each component corresponds to a topic, like in LDA, and is described by a graph structure. The main difference is that there is no learning phase, as we proceed online.

we show the comparison with lda in the appendix B.

If we take the stable Weighted Reservoir, which part of these topics do we cover as vertices? This is given in Figure 6. As k increases, we cover a larger part of the topics, to reach 100% for $k = 500$.

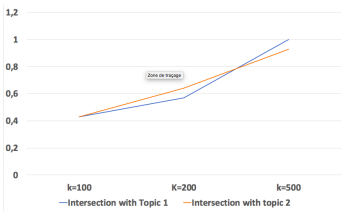

Fig. 6. Fraction of the LDA Topics as vertices of the giant component of the Weighted Reservoir

5 Measures of sentiments

The classical Word2Vec analysis takes texts L as inputs and construct vectors v of small dimension such that for two words w_i, w_j of the texts, the relative frequency of w_i, w_j in a sentence is proportional to the scalar $v^t(w_i).v(w_j)$. We take Twitter messages [24] as a benchmark and observe streams of Tweets, which are transformed in streams of edges of a Twitter graph. We sample edges uniformy in a Reservoir of size k, or with a weighted distribution which depends on the vectors v. The Reservoir is random graph which contains giant components. The edges of the nodes of high degree of the giant components define the tweets t_i of High Impact.

For the text analysis we take the tweets t_i and compute the weight $\rho(t_i, L)$ for a reference L, defined as follows:

$$\rho(t_i, L) = \sum_{(w_j, w_k) \in t_i} v^t(w_j).v(w_k)$$

We interpret $(w_j, w_k) \in t_i$, as the pair of words w_j, w_k appears in the same sentence of the tweet t_i. For a natural language such as English, L is the basic text reference, for example Wikipedia, and $v's$ are the associated vectors. We then construct a list L_o of *offensive* texts, which we want to detect and construct the associated vectors v_o. If only a few examples are available, we only modify the vectors $v's$ associated with the offensive words, and use the same vector $v's$ for the standard words.

These vectors can be very different from the $v's$. We then compare $\rho(t_i, L)$ and $\rho(t_i, L_o)$. A tweet t_i is *abusive* if $\rho(t_i, L_o) > c \cdot \rho(t_i, L)$ for some constant c.This approach generalizes to different natural languages L_i and various offensive texts $L_{o,i}$.

5.1 The detection of offensive High-Impact tweets

A *High-Impact* tweet is associated with an edge connected to a node of high degree. In Figure 7, the node 1 is of high degree and the edges e_1, e_4, e_5, e_8 correspond to High-Impact tweets.

Consider the Twitter stream associated with the tag *#Metoo*. A High Impact tweet is t_i: *@robinmonotti2: Forced or coerced vaccination is medical rape: a crime. We need a medical #MeToo movement now.* There are 7 words (@robinmonotti2, Forced, coerced, vaccination, medical, rape, crime) in the first sentence hence $42 = \binom{7}{2}$ pairs. In the second sentence, there are 5 words hence 10 pairs. In this case, $\rho(t_i, L_o) = 0.18$ and $\rho(t_i, L = 0.78)$, using the publicly available datasets that cover different hate speech-related categories[4]. We conclude that this tweet is not offensive.

5.2 Standard texts

We can apply our method to a standard text corpus. For each sentence, we first apply the *lemmatization* and the *Entity recognition* [16] steps and delete the stop words and the least frequent words. We generate either the bigrams (contiguous pairs of words) [17] or all the possible pairs of a given sentence, as potential edges. For the uniform sampling the weights of the edges are constant (for example equal to 1). For the weighted sampling, the weight of an edge is the Word2vec similarity of two words. In both cases, we process the text as a stream *without storing the entire text.*

6 Experiments

We propose a tool with two versions: the first version analyses Twitter streams on specific tags and the second version analyses texts[5]. We set the parameters of the windows (length τ and step λ) and the size K of the Reservoir and proceed online.

[4] https://github.com/alassou/t/blob/master/labeled_data.csv
[5] https://github.com/alassou/t/blob/master/wr.ipynb

6.1 Twitter streams

The Twitter version[6] is useful to detect trends and correlate different streams. A stream, determined by some keywords, generates edges such as $(u, v, text)$[7] where u is the author and v one of the selected tags in the *text*, the original tweet.

With a keyword such as *CNN*, we obtain: 3.10^3 edges and 10^3 sentences per minute. For a window of length $\tau = 10$ minutes, we have approximately $m = 3.10^4$ and $n = 8000$, as $m = O(n.\log n)$. The size $K = 400$ of the Reservoir is of the order $O(\sqrt{n}.\log n) \simeq 90.4 = 360$. The stability for the *uniform sampling* of the twitter stream is 0.9. Figure 7 gives the 2-core(C) of a giant component.

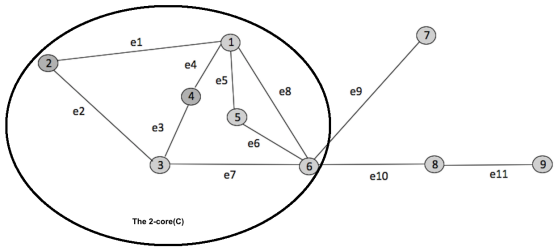

Fig. 7. A large connected component C in the Reservoir and its 2-core(C) (in the circle)

We captured 4 twitter streams on the tags #CNN, #FoxNews, #Bitcoin, and #Ripple during 24 hours with a window size of $\tau = 1h$ and a time interval $\lambda = 30$mins. Figure 8 indicates the number of edges in a window, approximately $m = 30.10^3$ per stream, for each stream, hence 10^6 edges in 24 hours. For the #Bitcoin and $k = 400$, the size of the giant component is approximately 100.

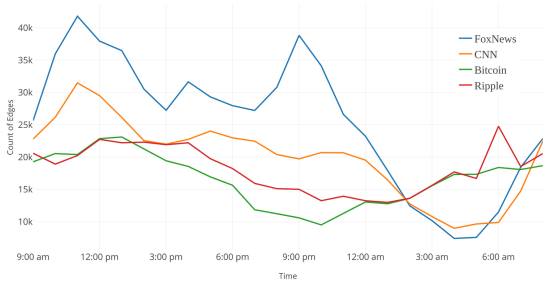

Fig. 8. Number of edges/per hour for 4 streams during 24h

[6] https://github.com/alassou/t/blob/master/topic.ipynb

[7] (@*thomasjacksonjr*, #*PrimeVideo*, "Watched #*ABCMurders* from beginning to end and it was pure #mystery and delight. Excellent show and a must watch for #*mysterylovers*. Great show on #*PrimeVideo*.")

6.2 Classical texts

We evaluate our methods on the NIPS dataset, (Googleplaystore_user_reviews) dataset and the Stanford Natural Language Inference (SNLI) corpus [7] which consists in 570k human-written English pairs: sentence and annotations.

We apply the uniform sampling and the weighted sampling where the weight is the absolute value of the Word2Vec similarity between two words. The stability of the two methods as a function of K is given in Figure 9.

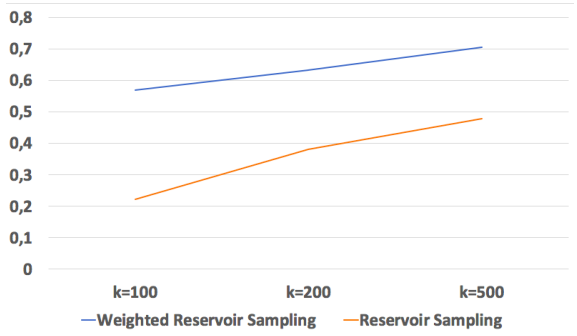

Fig. 9. Stability of the uniform and weighted sampling

The weighted sampling gives much better result as the uniform sampling stability is less then 0.5.

LDA [4] is a 2-phase method. First we select m topics from a base of documents. We then associate with each new document the distribution θ on the topics and for each topic the distribution of the words. Quality metrics [25] have been introduced to compare different methods. In our case, each experiment provides a set of giant components and each giant component is interpreted as a topic. The giant component is not a distribution but a graph with its own structure. We can only compare the support of the LDA distributions with the set of vertices of the graph. As we repeat different experiments, the expectation $\mathbb{E}(\rho(D,k))$ of the stability is very similar to the *Topic stability* introduced in [25]. The streaming approach is a 1-phase process for the uniform sampling with no classification in advance into m topics. It uses the words similarity for the weighted sampling. The main advantage is that we do not store the texts.

6.2.1 Central sentences and Attention mechanisms Each giant component can be analysed, starting from the node of maximal degree, the *center*, and their adjacent edges, the *central edges*. For the 2-core of Figure 7, we start with the center node 1 of maximum degree and its adjacent central edges e_1, e_4, e_5, e_8. Each edge is of the form $(u, v, text)$ and we can analyse the attention distribution[8] [22] of the words u and v in each "text" sentence.

The e_8 edge is *(skate, jump, "A boy is jumping on skateboard in the middle of a red bridge.")* and the e_4 edge is *(skate, sidewalk, "The boy skates down the sidewalk.")*. The attention analysis of the first sentence for the words *skate* and *jump* is given in the Figure 10. A giant component provides a *central node* and *central sentences:* the 4 sentences

[8] For a word u, its *attention in a sentence* is the distribution over the other words with a weight proportional to its Word2Vec similarity.

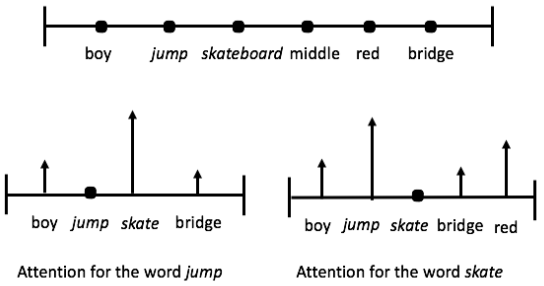

Fig. 10. The analysis of the sentence: *A boy is jumping on skateboard in the middle of a red bridge.*

associated with the edges of the central node *skate*, and then recursively along the tree decomposition of the component. At the next stage, the node 6 would be explored with three new edges.

If we classify the giant components into k-classes, viewed as topics, each component C would have a natural distribution over the topics as in [4].

7 Conclusion

We propose a tool which reads a stream of tweets or some standard text and propose a hierarchical topic representation. We read a stream of tweets, given some keywords such as "#CNN", approximately 10^3 tweets per minute. The tweets generate edges of a graph which we sample with a dynamic Reservoir sampling. We then observe the giant components of the Reservoir and each component is a topic, a list of tags which can also be represented as a hierarchical tree. For a standard text, we use a Weighted Reservoir sampling where the weight is the similarity between two words and construct the topics as giant components of the Reservoir.

References

1. Aggarwal, C.C.: An introduction to cluster analysis. In: Data Clustering: Algorithms and Applications, pp. 1–28. CRC Press (2013)
2. Babcock, B., Datar, M., Motwani, R.: Sampling from a moving window over streaming data. In: Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 633–634 (2002)
3. Bahmani, B., Kumar, R., Vassilvitskii, S.: Densest subgraph in streaming and mapreduce. Proc. VLDB Endow. **5**(5), 454–465 (Jan 2012)
4. Blei, D.M., Ng, A.Y., Jordan, M.I.: Lattent dirichlet allocation. Journal of Machine Learning Research **3:993-1022** (2003)
5. Blei, D., Lafferty, J.: Dynamic topic models. vol. 2006, pp. 113–120 (2006). https://doi.org/10.1145/1143844.1143859
6. Bollobas, B.: Random Graphs. Cambridge University Press (2001)
7. Bowman, S., Angeli, G., Potts, C., Manning, C.: A large annotated corpus for learning natural language inference (08 2015). https://doi.org/10.18653/v1/D15-1075
8. Braverman, V., Ostrovsky, R., Zaniolo, C.: Optimal sampling from sliding windows. In: Proceedings of the Twenty-eighth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. pp. 147–156 (2009)
9. Erdös, P., Renyi, A.: On the evolution of random graphs. In: Publication of the mathematical institute of the Hungarian Academy of Sciences. pp. 17–61 (1960)
10. Li, X.L., Eisner, J.: Specializing word embeddings (for parsing) by information bottleneck. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. pp. 2744–2754 (2019)

11. Lin, Z., Feng, M., dos Santos, C.N., Yu, M., Xiang, B., Zhou, B., Bengio, Y.: A structured self-attentive sentence embedding. CoRR **abs/1703.03130** (2017)
12. Mathieu, C., de Rougemont, M.: Large very dense subgraphs in a stream of edges. CoRR **abs/2010.07794** (2020), `https://arxiv.org/abs/2010.07794`
13. McGregor, A., Tench, D., Vorotnikova, S., Vu, H.T.: Densest subgraph in dynamic graph streams. CoRR **abs/1506.04417** (2015)
14. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. CoRR **abs/1301.3781** (2013)
15. Molloy, M., Reed, B.: The size of the giant component of a random graph with a given degree sequence. Comb. Probab. Comput. **7**(3), 295–305 (1998)
16. Nadeau, D., Sekine, S.: A survey of named entity recognition and classification. Lingvisticae Investigationes **30**(1), 3–26 (Jan 2007)
17. Nawab, R.M.A., Stevenson, M., Clough, P.: Comparing Medline citations using modified N-grams. Journal of the American Medical Informatics Association **21**, 105–110 (2013)
18. Panigrahi, A., Simhadri, H.V., Bhattacharyya, C.: Word2Sense: Sparse interpretable word embeddings. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. pp. 5692–5705. Association for Computational Linguistics (2019)
19. Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. In: Proc. of NAACL (2018)
20. Ratinaud: Iramuteq: Interface de r pour les analyses multidimensionnelles de textes et de questionnaires [computer software]. http://www.iramuteq.org (2009)
21. de Rougemont, M., Vimont, G.: The content correlation of streaming edges. In: IEEE International Conference on Big Data. pp. 1101–1106 (2018)
22. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. CoRR **abs/1706.03762** (2017)
23. Vitter, J.S.: Random sampling with a reservoir. ACM Trans. Math. Softw. **11**(1), 37–57 (1985)
24. Warner, W., Hirschberg, J.: Detecting hate speech on the world wide web. In: Proceedings of the Second Workshop on Language in Social Media. p. 19–26. LSM '12, Association for Computational Linguistics, USA (2012)
25. Xing, L., Paul, M.J., Carenini, G.: Evaluating topic quality with posterior variability. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP-IJCNLP). pp. 3471–3477. Association for Computational Linguistics (2019)

Appendix

A Weighted Reservoir sampling

We read a stream of edges $e_1, e_2, ..., e_m, ...$ where each edge e_i has a weight w_i and keep K edges. We keep the K first edges in the Reservoir R. For each new edge e_i, where $i > K$, we decide to select e_i with probability:

$$\frac{K \cdot w_i}{\sum_{j \leq i} w_j}$$

If we select e_i we insert it in the Reservoir in a random position j, replacing the current element: we select $1 \leq j \leq K$ with probability $1/K$.

B Comparaison with lda

To compare our method with LDA, we set $m = 2$ topics. Topic1 was described by ['man', 'woman', 'wear', 'boy', 'walk', 'girl', 'dog', 'stand', 'street', 'play', 'child', 'dress', 'hold', 'water'], whereas Topic2 was described by ['man', 'sit', 'people', 'woman', 'look', 'stand', 'group', 'shirt', 'wear', 'table', 'hold', 'player', 'work', 'play', 'ball']. The distributions are given in Figure 12.

Fig. 11. Giant components from a Reservoir

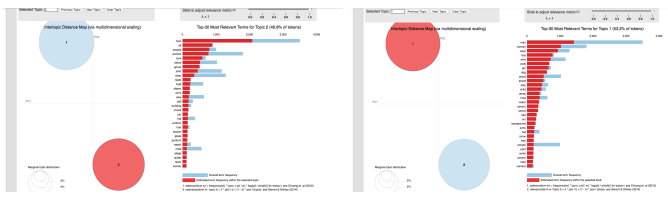

Fig. 12. The distributions of the 2 topics