Knowledge Discovery from Temporal Social Networks

Shazia Tabassum, Fabíola S. F. Pereira, and João Gama

***Abstract*—Extracting knowledge from network data is a complex task. It requires the use of appropriate tools and techniques, especially in scenarios that take into account the volume and evolving aspects of the network. There is a vast literature on how to collect, process, and model social media data in the form of networks, as well as key metrics of centrality. However, there is still much to be discussed in relation to the analysis of the underlying huge networks. In this article the goal is to discuss briefly different techniques in the process of gaining knowledge from networked data, especially considering time perspective. Firstly, we presented some techniques for online sampling of temporally ordered massive networked data which can be efficiently plugged in for a further network mining task. Next are discussed approximate mechanisms for high speed network change detection using centrality measures. Hand in hand we also presented, the ways of processing temporal network streams, applications with real data and illustrations using network visualizations. In the end are discussed concepts related to temporal ordering of links and paths in temporal networks.**

***Index Terms*—Temporal Networks, Sampling Evolving Networks, Change Detection, Streaming Network Analysis.**

I. Introduction

Handling and processing, high-velocity networked data generating from real-world applications is a current exigency. Dynamic and evolutionary learning with space and time efficient techniques is an imperative solution for these flooding data networks. We are here focusing this issue in the arena of dynamic sampling and change detection from temporally evolving large networks.

Dynamic Sampling is an exemplary way to deal with the issues relating to massive evolving data, like answering approximate queries, running simulations, understanding and modeling true network structure, inadequate data, detecting events/changes in the network, etc. Apart from other applications one of its major appeals lies in estimating the true network properties [1] that cannot be handled in entirety. Though sampling population is a statistically established area, it is not much explored in the current scenario of real-time dynamic networked data. A comprehensive survey on sampling network streams can be found in [2]. However it does not focus on multi or weighted graphs. We presented in the section III some dynamic sampling mechanisms.

Another problem discussed here is Dynamic Change Detection. Given a temporal network discussed above, how can we detect structural changes across different time steps with an online approach, under the one-pass constraint of data? According to Gama et al. [3], change detection refers to techniques and mechanisms for explicit drift detection characterized by the identification of change points or small time intervals during which changes occur. In evolving networks context, these changes can be detected observing the whole network, for instance communities [4] and motifs [5] evolution; or the changes can be analyzed in a node-centric way, where nodes centrality and roles are observed during network evolution [6]. In section IV, we discussed these techniques.

Shazia Tabassum and João Gama are with INESC TEC, University of Porto, Portugal; and Fabíola S. F. Pereira is with Federal University of Uberlandia, Brazil. Corresponding Author e-mail: (jgama@fep.up.pt).

II. Literature Review

There are a number of research works on sampling static networks but here we will only discuss dynamic sampling algorithms on temporal/evolving network streams coherent to the scope of this article. As most of the real world networks follow power-law distributions in their degree, clustering coefficient etc. From this kind of networks that posses low number of nodes with high degree and high number of nodes with low degree, we are likely to get samples with most or all of nodes from this lighter long tail. Since traditional stratified sampling does not hold with the constraint of one pass, fast and space efficiency, we need to find some strategies that overcome these biases in an efficient way.

In [2] Ahmed et al. presented a simple edge stream sampling which uses a similar approach as reservoir sampling [7] (refer section III-B2). In this case, a new edge enters into the reservoir if its hash value is within top-m minimum hash values, where m is the size of reservoir. However, the method did not ensue as an efficient representative sample. Additionally, Ahmed et al. [2] also proposed a *Partially-induced edge sampling algorithm* called (PIES). This algorithm works by storing nodes and also edges probabilistically in their reservoirs while deleting the one already present at random as in the reservoir sampling [7]. It maintains a fixed size reservoir of nodes while the reservoir size for edges varied based on nodes. CPIES, an update over PIES is given by Zhang et al. [8]. They modified the decremental module of PIES, by deleting the nodes from the reservoir with minimal degree to produce a better cluster preserving structure. PIES had a selection bias to high degree nodes which enhances in CPIES as it tends to delete low degree nodes. Papagelis et al. [9] proposed sampling algorithms that given a user in a social network quickly obtains a near-uniform random sample of nodes in its neighborhood using random walks.

A. Change Detection in Dynamic Networks

Processing graphs as streams is an incoming problem. The work [5] is one of the most complete when considering data mining in evolving graph streams. The focus is on mining closed graphs, not on change detection though. In [10] a framework for processing graphs as streams is proposed for the link prediction task. This framework considers the cumulative grown of the graph, not addressing the space saving issue [11].

The most studied events in dynamic networks are anomalies and bursts [4]. Anomaly detection refers to the discovery of rare occurrences in data sets [12]. The most representative work in anomaly detection for dynamic graphs is [13]. It addresses the problem considering a time sequence of graphs (graph sequences). The focus is on faults occurring in the application layer of web-based systems. First, they extract activity vectors from the principal eigen vector of dependency matrix. Next, via singular value decomposition, it is possible to find a typical activity pattern (in $t-1$) and the current activity vector (t). In the end, the angular variable between the vectors defines the anomaly metric. The network processing is through snapshots, not in a streaming fashion. Akoglu and Faloutsos [14] used the Eigen Behavior based Event Detection (EBED) method to detect events in SMS interactions a *who-texts-whom network*. They are able to detect events in a global perspective of the network.

Eberle et al. [15] proposed to discover anomalous sub graphs in graph streams using a change detection metric. The authors compute graph properties GP as the graph evolves and then compare, using average and standard deviation, if there is an abrupt change in these GP. If yes, the change has been detected. The algorithm processes incoming edges in batches using sliding window strategy.

III. Sampling Evolving Multi-Graphs

Evolving Network Stream: In a streaming scenario, a temporally evolving network is usually considered as a stream of edges$\{e_1, e_2, e_3, e_4...\} \in E$ generating from a graph stream G. Every edge $e = (u, v, t)$ is composed of a pair of vertices's from V and a time-stamp t, which indicates the time of occurrence of e. We assume that the edges are streaming in the order of time-stamps. E and V can have a temporally changing cardinality.

Multi-graph Stream In a multi-graph stream an edge e can recur randomly in G at various time-stamps t. Examples include phone calls, tweets, co-authorship etc.

Graph Size Graph size in evolving networks is usually the number of edges $|E|$ at any time t or a time interval τ.

A. Methods of Sampling Streaming Graphs

1) Node Based Methods: Node based methods in general, sample a set of nodes from the original graph. The resultant samples contain a set of vertices from the graph stream and showing no connections between them. Acquiring the corresponding edges between them increases the time complexity. Some sampling methods ([16],[8]) store the set of nodes and also the set of edges (which also contain nodes) for ease of computation.

2) Edge Based Methods: These samples are generated by selecting a subset of edges from the original graph. The resultant graph is a subgraph of original graph with nodes and edges. Edges can be labelled, weighted or attributed.

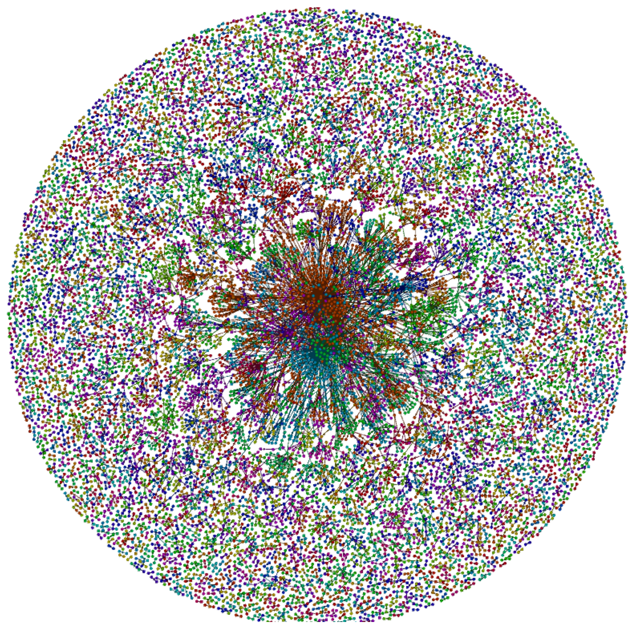


Fig. 1: Pictorial representation of 10^4 top K edges sample at the end of 31 days stream of telecom phone calls (edges) using Space Saving (colors represent communities).

B. Algorithms for Sampling

1) Space Saving Algorithm (SS): What if we need a sample of most active connections in the network at every time stamp t, without having enough space to store all the connections in the network? In such cases the Space Saving Algorithm proposed by [17] is an appropriate choice. Space saving algorithm is the most approximate and efficient algorithm for finding top frequent elements from a data stream. The algorithm maintains partial interest of information as it monitors only a subset of elements from the stream. Considering the edge stream [18], [19] it maintains counters for every element in the sample and increments its count when the edge re-occurs in the stream. If a new edge is encountered in the stream it is replaced with an edge of the least counter value and its count is incremented. Consequently it gives the top K frequent edges at any t from a multi-graph stream. Note that this algorithm keeps track of the top frequent edges but not how much frequent they are. The samples posses a good community structure [18]. Top K, i.e the sample size in this case should be given manually, which do not grow with the growth of the network. Figure 1 shows a sample from a stream of anonimised phone calls provided by a service provider with around 280 to 10 calls per second at mid-day and mid-night and on an average 12M calls per day made by 4M subscribers.

2) Reservoir Sampling (RS): This is a well known algorithm of Reservoir Sampling [7]. This algorithm works by maintaining a reservoir of edges with a predefined sample of size K. In the edge streaming scenario, firstly the reservoir is filled with the initial edges from the stream. Every edge coming after that is computed for the probability K/i of being inserted. Where i is the length of the stream exhausted till then. If the probability of the contending edge in the stream is greater than the probability of an edge in the reservoir which is $1/i$, then uniformly at random an edge is picked from the reservoir. The picked edge is replaced with the edge in the stream. In case the probability is less, the streaming edge is discarded. As i increases, the probability of i^{th} element getting inserted into the reservoir decreases. Therefore, it leads to samples with very old elements from the stream. [18] and [20] show that these samples posses very weak community structure and high bias to low degree nodes.

3) Biased Random Sampling (BRS): This algorithm is proposed as a simple variant of RS in [18]. It is actually an unbiased random sampling technique but considering RS as a standard, this is its biased version. Unlike the above algorithm where the probability of streaming edges diminishes as the stream progresses, this algorithm ensures every edge goes into the reservoir. An edge from the reservoir is chosen for replacement at random. Therefore, the edge insertion is deterministic but deletion is probabilistic. An edge staying for a long time in the reservoir has the same probability of getting out as an edge inserted recently. Consequently, the edges in the reservoir are distributed randomly over time. It gives a better community structure and distributions close to true network than Reservoir Sampling [18] and [20].

The complexity of algorithm increases linearly with the size K of sample in the above mechanisms.

4) Biased Dynamic Sampling Using Forgetting (SBias): Recent interactions are evident to show the current status of relationships, nevertheless some old stronger relations are also substantially significant [20]. Therefore, this sampling algorithm uses a fast memory-less forgetting function with two parameters that help introduce biases on the network based on time and relationship strengths. The main idea is to exponentially forget edges based on time and weight of edges. At every time interval τ the frequency of an edge in a multi-graph is mapped to its weight. Every edge is considered a vector stream of its occurrence and non-occurrence at every τ. The forgetting function is imposed on all $|E|$ edge vector streams independently, where E is an edge set at time interval τ. An illustration is given in figure 3. A threshold θ is used to eliminate the edges from the network. The parameter values α and θ can be increased to decrease the sample size.

$$\hat{w}_\tau(e) = w_\tau(e) + (1-\alpha)\hat{w}_{\tau-1}(e) \tag{1}$$

Where $w_\tau(e)$ is the weight of an edge at τ. This algorithm provides better distributions (closest to the true network) than the above algorithms (figure 8) [20]. This illustration is given using the CollegeMsg Networked data set comprised of private messages sent on an online social network at the University of California, Irvine [21], obtained from SNAP data sets[2]. One of the important property of this algorithm is it does not maintain a fixed size sample, which gets decreasing with the increasing size of true network (most of them obeying a power law increase over time). Though the sample size varies it is bounded by the variance of network size per time step.

[2]Data available at https://snap.stanford.edu/data/CollegeMsg.html

C. Sampling Ego-Networks with forgetting factor

As ego-networks also densify [22] over time by making infeasible to store all the information of them in the memory. An example graph of an ego-network (2-levels) from a phone calls network is shown in figure 2. The function (1) and forgetting mechanism here is the same as in the above SBias model but applies for a single ego/personal network with any number of levels/radius from the ego. The main variation is that in the above algorithm we don't need to remove the edges adjacent to the deleted edge. In this case we remove the edges adjacent to a deleted edge which otherwise do not have a connection to the ego [23]. This method has an advantage as an input for detecting changes in personal networks, predicting links and detecting frauds etc.

IV. Change Detection in Evolving Networks

We present here the change detection techniques from a node-centric perspective in a network stream processing environment. We call this task as *Node Centrality Change Detection*. The techniques here presented were proposed in [6] and focus on tracking nodes properties instead of global graph properties ([15], [14]).

In the approaches presented in this section, we consider an edge stream S which is a continuous and unbounded flow of objects $E_1, E_2, E_3.....$, where each edge E_i is defined by (v, u, t) which represents a connection between vertices/nodes u and v at time t. The vertices $\{u, v, ...\} \in V$ and get added to or deleted from V at anytime t.

A. Processing a Streaming Network

For every incoming edge (v, u, t) from a stream S, the centrality scores $C^m(v)$ and $C^m(u)$ for nodes u and v are updated in the order of t, for m being a node centrality metric (in case of degree centrality, the degree of nodes u and v is updated for every incoming edge $\{u,v\}$ at t where as in the case of betweenness and closeness, the centrality of nodes are updated only after every T). Another variable T is a discrete time-step/time-interval with granularity defined by the user. After every T the centrality scores are reset and starts accumulating again. Therefore, we keep track of centrality score of nodes per day. Consequently we store a set of nodes (with changing cardinality) and a streaming vector of its associated centralities per time step T. As a result, we have an independent non stationary stream of centrality scores $\{C^m_{T_1}, C^m_{T_2}, C^m_{T_3}........\}$ for every node v in S after every time step T. To get a normalized version of scores, after every timestep T the centrality of a node is divided by the number of nodes in graph at T. Therefore we have normalized centrality scores in the vector stream. Further we employed aggregating mechanisms to the above streams of centrality scores per node.

B. Aggregating Mechanisms

For notational simplicity in the below equations we use C_T for $C^m_T(v)$ as all notations for the techniques below are considered for a stream of centrality scores per node per centrality metric.

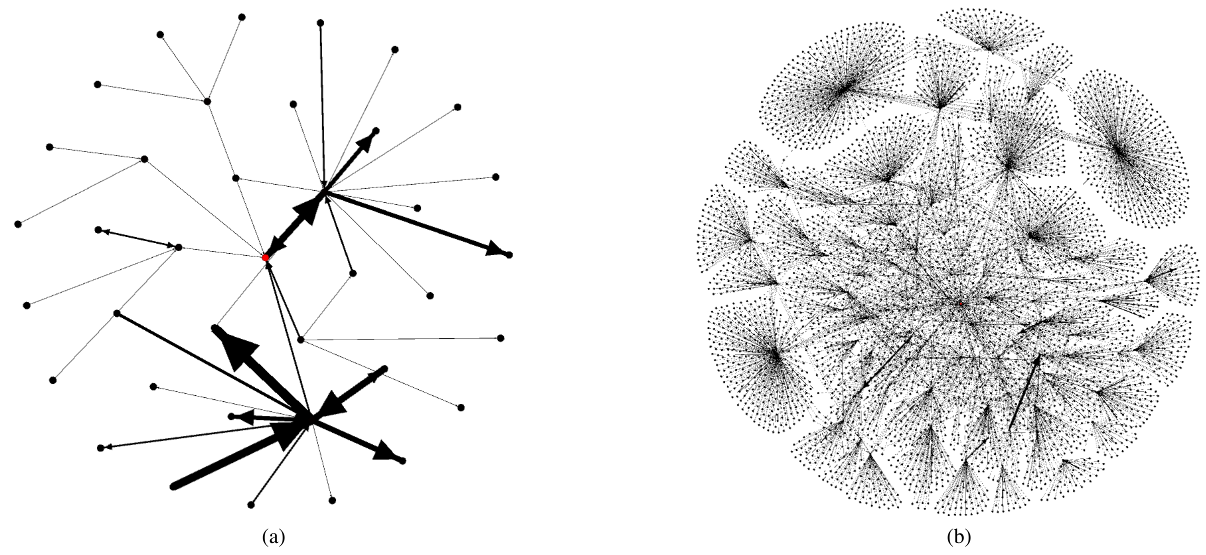


Fig. 2: Ego-network of a user (red) on day 1 and day31 from the phone call network.

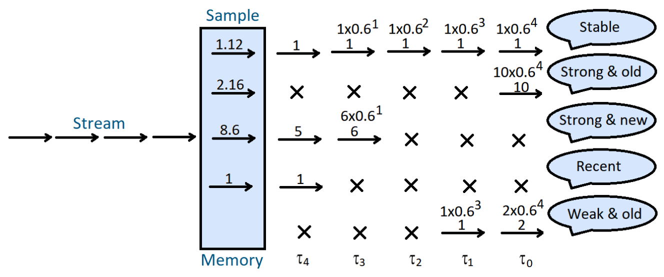


Fig. 3: Pictorial representation of SBias sample at τ_4 with $\alpha = 0.4$ and $\theta = 0.5$.

1) Moving Window Average (MWA): A window of size W_S consists of data points from the latest temporal time steps $\{T, T-1, T-2, ..., T-(W_S-1)\}$. The window keeps on sliding to always maintain the latest W_S time steps and the data points from $T - W_S$ are forgotten. Alongside, the mean of data points within the window is calculated by using simple equation (2) where C_{T-i} is the stream of centrality scores at time-step $T - i$ using measure m per node. In this approach all the data points in the window are assigned equal weights.

$$\mu_T = \frac{1}{W_S} \sum_{i=0}^{W_S-1} C_{T-i} \tag{2}$$

As the window slides the mean of data points in the window is updated, either using the above equation (2) for small window sizes and equation (3) for large window sizes.

$$\mu_T = \mu_{T-1} W_S - C_{T-W_S} + C_T \tag{3}$$

2) Weighted Moving Window Average (WMWA): Weighted moving window average follows the same window sliding strategy as in MWA and computes average over the data points in the window. The improvement over MWA is that the accumulated data points per time step T in the window are weighted linearly as given in equation (4). The oldest data points in the window attain a least weight and the latest data point acquires the highest weight linear to the least one. Weights are updated, when the window slides. Assignment of weights per data point depends on the size of window.

$$\mu_T = \sum_{i=0}^{W_S-1} \frac{C_{T-i}(W_S - i)}{W_S - i} \tag{4}$$

3) Page Hinckley Test (PH): Page Hinckley [24] is one of the memory less sequential analysis techniques typically used for change detection [25], [26], [27], [3]. We use it as a non-parametric test, as the distribution is non stationary and not known. This test considers a cumulative variable m_T, defined as the cummulated difference between the latest centrality score at T and the previous mean till the current moment, as given in the equation (5) below:

$$m_T = \sum_{i=1}^{T} |C_T - \mu_{T-1}| - \alpha \tag{5}$$

Where $\mu_T = 1/|T| \sum_{i=1}^{T} C_i$, $\mu_0 = 0$ and α = magnitude of changes that are allowed. For calculating μ_T we also need to store the number of time-steps passed.

The equation (5) given above uses fixed α value, which is not pertinent with our multiple vector streams of centralities per node, where the centrality scores of few active nodes are way higher than some least active nodes. Therefore, using same value of α over differing node centralities would not be fair enough. Hence, we use a relative α, which is relative with the differing centrality scores per node. Relative α is a point percentage of previous aggregated mean of that node, as given in equation (6).

$$m_T = \sum_{i=1}^{T} |C_T - \mu_{T-1}| - \alpha\mu_{T-1} \tag{6}$$

Further to calculate change point score we need a variable M_T which is the minimum value of m_T and is always maintained and updated for every new time step T as given in equation 7

$$M_T = min(m_T; i = 1...T) \tag{7}$$

C. Detecting Change Points

1) Change Point Scoring Function: To detect the change points and their magnitude after every time-step T in MWA and WMWA, we use a change point scoring function given in equation (8)

$$\Gamma_T = \frac{|C_T - \mu_{T-1}|}{max(C_T, \mu_{T-1})} \tag{8}$$

Where C_T is the current centrality score and μ_{T-1} is the mean of previous centrality scores in the window. The change point scoring function gives the percentage point increase or decrease of the current centrality score with the previous mean. It takes values $0 \leq \Gamma_T \leq 1$.

For a PH test, after every time-step T the change points are scored using the equation (9).

$$\Gamma_T = m_T - M_T \tag{9}$$

2) Change Point Detection: We can decide the magnitude of change allowed by the above change point scoring function. For this we use a threshold θ on Γ, to signal an alarm of change in the node. It takes values either 0 or 1. "1" indicates a node centrality change and "0" indicates no change.

$$\epsilon_T = \begin{cases} 1, & \text{if } \Gamma_T \geq \theta. \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

We also apply a relative θ for detecting change points in PH Test only, as the change point scores from windowed approaches are already normalized in equation (8). Therefore to normalize threshold over multiple streams of centrality scores in PH test we use a relative threshold θ by multiplying the threshold θ with M_T of that node at time T as in equation 11.

$$\epsilon_T = \begin{cases} 1, & \text{if } \Gamma_T \geq (\theta \times M_T). \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

While carrying out the above-mentioned mechanisms we considered the following assumptions. For window based approaches change detection starts only after the window of size W_S is filled. If there exists no edges for a node in a time step T, then the mean is calculated assuming a "0" centrality score. If a node is newly introduced (with edges) in the stream in the time interval T then the previous mean at $T-1$ is considered "0" during change point scoring. In window based approaches if a node does not appear in the stream for a W_S time steps, the node is deleted to save space.

The above described model for change detection using centrality measures is illustrated in figure 4

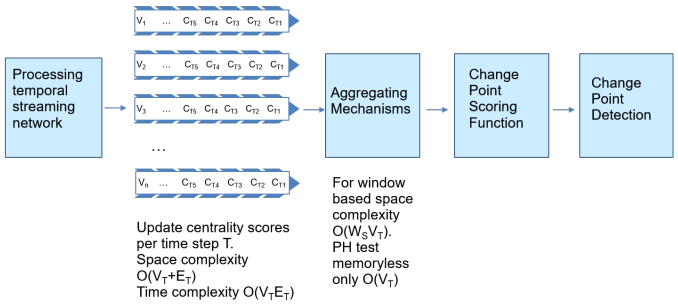

Processing temporal streaming network
V1 ... CT5 CT4 CT3 CT2 CT1
V2 ... CT5 CT4 CT3 CT2 CT1
V3 ... CT5 CT4 CT3 CT2 CT1
...
Vn ... CT5 CT4 CT3 CT2 CT1
Update centrality scores per time step T. Space complexity O(VT+ET) Time complexity O(VTET)
Aggregating Mechanisms
For window based space complexity O(WSVT). PH test memoryless only O(VT)
Change Point Scoring Function
Change Point Detection

Fig. 4: Preference change detection model.

D. Centrality changes in Twitter network

In [28], [6] we applied the node centrality change detection problem to infer user preference changes in temporal interaction networks. Figure 5 illustrates the Twitter evolving network related to Brazilian news we utilized. The network represents interactions among users through retweets.

In order to exemplify the techniques presented in this section, Figure 6 shows the events detected in Twitter network from Figure 5 based on in-degree centrality and MWA aggregating mechanism.

V. Temporal Centralities in Temporal Social Networks

The topological structure of static networks can be characterized by an abundance of measures. In essence, such measures are based on connections between neighboring nodes (such as the degree or clustering coefficient), or between larger sets of nodes (such as path lengths, network diameter and centrality measures). When the additional dimension of time is included in the network picture, many of these measures need rethinking. Our main goal here is to represent social networks, specially Twitter, founded on temporal graphs theory [29]. According to Holme and Saramaki [30], temporal networks can be divided into two classes corresponding to the types of representations: contact sequences and interval graphs. While in contact sequences, the edges are active over a set of times, in interval graphs they are active over a set of intervals. In Figure 7 we exemplify Twitter social network as an interval graph.

In temporal networks the concept of geodesic distance should take into account the temporal ordering of links [30]. A temporal path $P_{u,v}$ in a temporal graph G is a sequence $P_{u,v} =< (v_1, v_2, t_1), (v_2, v_3, t_2), ..., (v_{k-1}, v_k, t_{k-1}) >$, where $(v_i, v_{i+1}, t_{init}, t_{end}) \in E$ is the i-th temporal edge on $P_{u,v}$, $1 \leq i \leq k$, R is the retention time of nodes, i.e., the time between information arrival in the node and the instant from which it can be forwarded, T is the edge traversal time, $t_i + R + T \leq t_{i+1}$, $t_{init} \leq t_i \leq t_{end}$, $n \leq t_1$ and $t_{k-1} \leq N$, $u = v_1$ and $v = v_k$.

In a Twitter temporal graph representation, one can adopt $R = 1$ day and $T = 0$, as tweets are published instantaneously and the average interaction time for posts is one day [31]. Considering the temporal network of Figure 7 and the parameters $W = [1, 9]$, $T = 0$ and $R = 1$, we can cite some examples

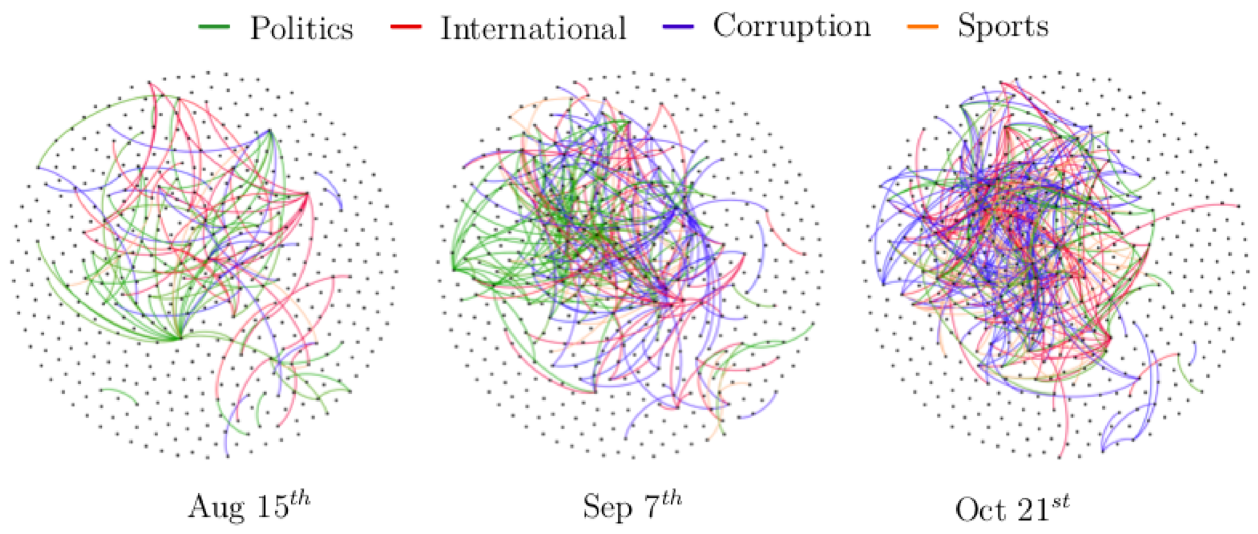


Fig. 5: Snapshots of samples of the evolving interaction network. Nodes are Twitter users. One tie from user u_1 to u_2 means that u_2 retweed at t some text originally posted by u_1. Colors represent topics that users are talking about. The samples were built by filtering nodes with degree between 50-22000 and edges representing the 4 most popular topics. Each snapshot corresponds to 1 day time-interval. This figure highlights the *edges* evolving aspect. Nodes are not evolving for better visualization.

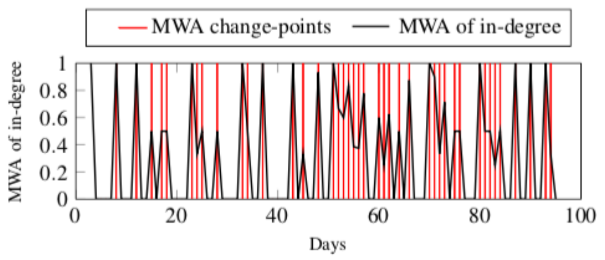


Fig. 6: Change events detected (red) in Twitter network based on in-degree centrality and MWA aggregating mechanism.

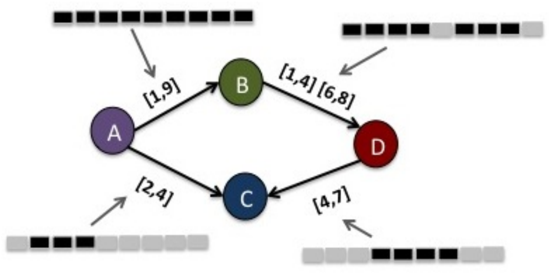


Fig. 7: Twitter as an interval graph. Nodes are Twitter users and an edge $(u, v, t_{init}, t_{end})$ indicates that v starts following u at t_{init} and unfollows u at t_{end+1}.

of temporal paths: $P_{A,D} =< (A, B, 1), (B, D, 2) >= 1$ (fastest path), $P_{A,D} =< (A, B, 2), (B, D, 6) >= 4$, $P_{A,C} =< (A, C, 2) >= 0$ (fastest path).

Centrality metrics that take into account the distance between two nodes are impacted by temporal paths definition. We focus on closeness and betweenness [32]. To compute these metrics in a temporal network scenario, we need to consider the number of fastest paths instead of the number of shortest paths as in static definition. A comprehensive study on centrality metrics for temporal networks can be found on [33].

As application example of temporal centralities in temporal networks, in [28] we discuss that tracking how follow/unfollow relationships on Twitter evolve over time can help us to understand their impact on users behaviors.

VI. Conclusion

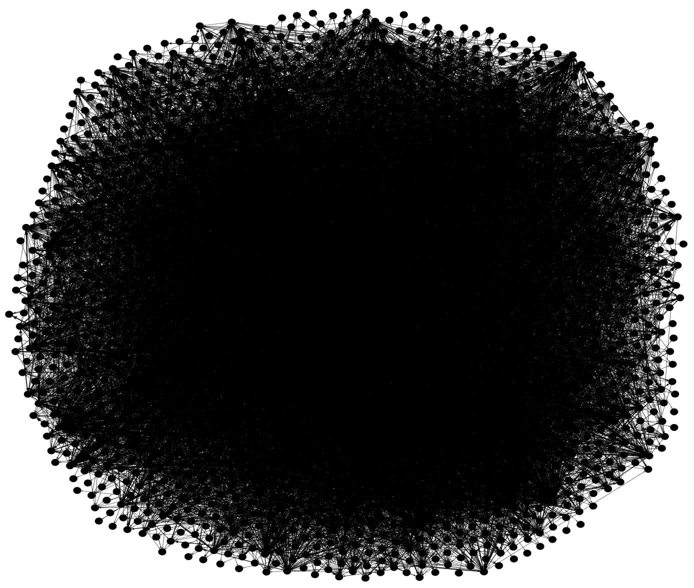
We discussed three topics in this article concerned from the size and complexity of the networks in the start to finish of knowledge discovery process. Four sampling mechanisms for fast massive scale free networks were presented, namely Space Saving for top K, Reservoir Sampling, Biased Random Sampling and Sampling using forgetting (SBias). Their properties and biases were discussed supported with some illustrations for better comprehensibility.

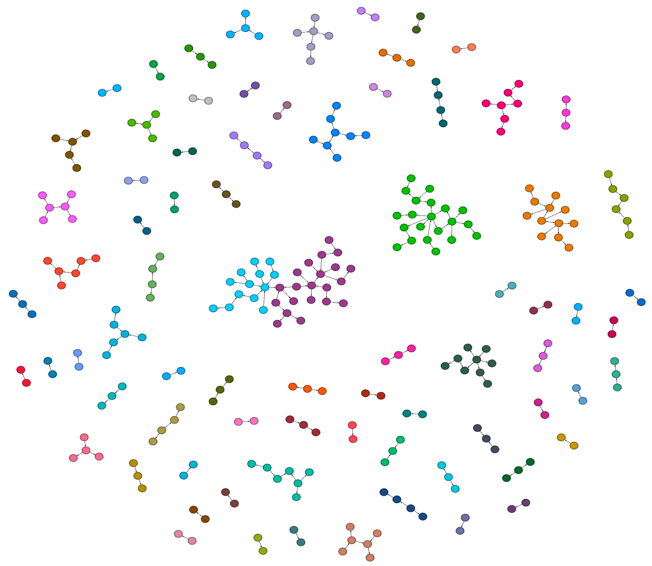
Further we demonstrated a fast approach for detecting changes in the network using centralities alternative to node features or content that is time and space expensive to be acquired. Additionally these are complimented with some aggregating and memory less tests for efficient change points detection.

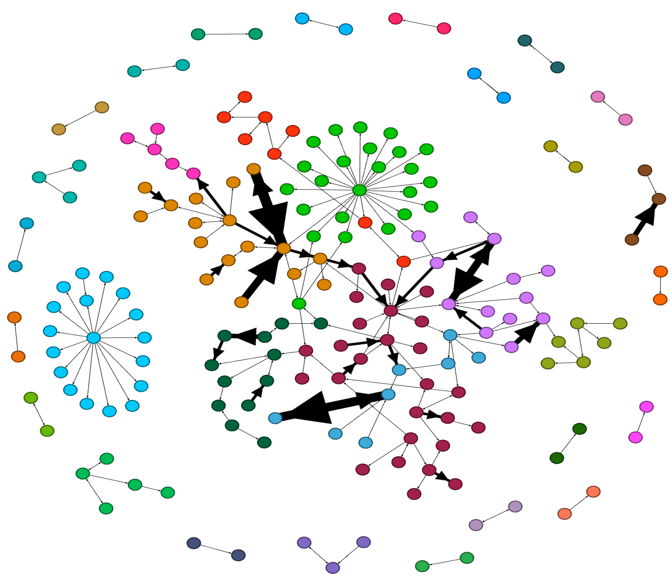
Lastly some concepts relating to information transmission and temporal paths were discussed briefly, opening new range of possibilities in these kind of networks.

Acknowledgment

The authors would like to thank the support of the project TEC4Growth RL SMILES Smart, mobile, Intelligent and Large Scale Sensing and analytics NORTE-01-0145-FEDER-000020 which is financed by the North Portugal regional operational program (NORTE 2020), under the Portugal 2020 partnership agreement, and through the European regional development fund.


(a) True Network #Nodes=1899 #Edges=20296 AvDeg=10.6 D=0.008 #Components=16


(b) RS #Nodes=296 #Edges=225 AvgDeg=0.76 Density=0.005 #Components=76


(c) BRS #Nodes=189 #Edges=225 AvgDeg=1.196 Density=0.006 #Components=25

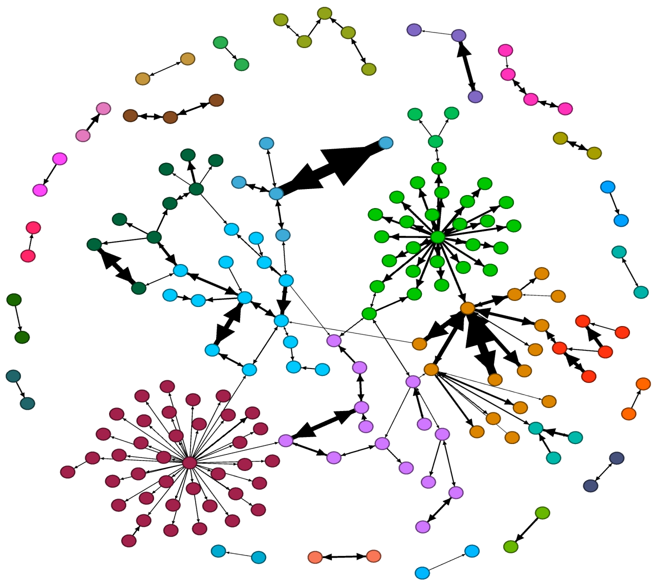

(d) SBias #Nodes=182 #Edges=225 AvgDeg=1.236 Density=0.007 #Components=21

Fig. 8: Snapshot at the end of observed stream (CollegeMsg) of true network and samples (1%).

References

[1] N. K. Ahmed, N. Duffield, T. L. Willke, and R. A. Rossi, "On sampling from massive graph streams," *Proceedings of the VLDB Endowment*, vol. 10, no. 11, pp. 1430–1441, 2017.

[2] N. K. Ahmed, J. Neville, and R. Kompella, "Network sampling: From static to streaming graphs," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 8, no. 2, p. 7, 2014.

[3] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, p. 44, 2014.

[4] M. Cordeiro and J. Gama, *Online Social Networks Event Detection: A Survey*. Cham: Springer International Publishing, 2016, pp. 1–41.

[5] A. Bifet, G. Holmes, B. Pfahringer, and R. Gavaldà, "Mining frequent closed graphs on evolving data streams," in *17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '11, 2011, pp. 591–599.

[6] F. S. F. Pereira, S. Tabassum, J. Gama, S. de Amo, and G. M. B. Oliveira, *Processing Evolving Social Networks for Change Detection Based on Centrality Measures*. Cham: Springer International Publishing, 2019, pp. 155–176.

[7] J. S. Vitter, "Random sampling with a reservoir," *ACM Transactions on Mathematical Software (TOMS)*, vol. 11, no. 1, pp. 37–57, 1985.

[8] J. Zhang, K. Zhu, Y. Pei, G. Fletcher, and M. Pechenizkiy, "Clustering-structure representative sampling from graph streams," in *International Workshop on Complex Networks and their Applications*. Springer, 2017, pp. 265–277.

[9] M. Papagelis, G. Das, and N. Koudas, "Sampling online social networks," *IEEE Transactions on knowledge and data engineering*, vol. 25, no. 3, pp. 662–676, 2013.

[10] J. Fairbanks, D. Ediger, R. McColl, D. A. Bader, and E. Gilbert, "A statistical framework for streaming graph analysis," in *IEEE/ACM Int. Conf. on Advances in Social Networks Analysis and Mining*, ser. ASONAM '13, 2013, pp. 341–347.
[11] J. Gama, *Knowledge discovery from data streams*. CRC Press, 2010.
[12] L. Akoglu, H. Tong, and D. Koutra, "Graph based anomaly detection and description: a survey," *Data Mining and Knowledge Discovery*, vol. 29, no. 3, pp. 626–688, 2015.
[13] T. Ide and H. Kashima, "Eigenspace-based anomaly detection in computer systems," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '04, 2004, pp. 440–449.
[14] L. Akoglu and C. Faloutsos, "Event detection in time series of mobile communication graphs," in *Proceedings of 27th army science conference*, ser. 18, vol. 2, no. 3, 2010.
[15] W. Eberle and L. Holder, "Identifying anomalies in graph streams using change detection," in *KDD Workshop on Mining and Learning in Graphs (MLG)*, 2016.
[16] N. K. Ahmed, J. Neville, and R. Kompella, "Space-efficient sampling from social activity streams," in *Proceedings of the 1st international workshop on big data, streams and heterogeneous source mining: algorithms, Systems, Programming Models and Applications*. ACM, 2012, pp. 53–60.
[17] A. Metwally, D. Agrawal, and A. El Abbadi, "Efficient computation of frequent and top-k elements in data streams," in *International Conference on Database Theory*. Springer, 2005, pp. 398–412.
[18] S. Tabassum and J. Gama, "Sampling massive streaming call graphs," in *ACM Symposium on Advanced Computing*, 2016, pp. 923–928.
[19] S. Tabassum, "Social network analysis of mobile streaming networks," in *Mobile Data Management (MDM), 2016 17th IEEE International Conference on*, vol. 2. IEEE, 2016, pp. 20–25.
[20] S. Tabassum and J. Gama, "Biased dynamic sampling for temporal network streams," in *Complex Networks and Their Applications VII. COMPLEX NETWORKS 2018. Studies in Computational Intelligence, vol 812*. Springer, 2018, pp. 512–523.
[21] P. Panzarasa, T. Opsahl, and K. M. Carley, "Patterns and dynamics of users' behavior and interaction: Network analysis of an online community," *Journal of the American Society for Information Science and Technology*, vol. 60, no. 5, pp. 911–932, 2009.
[22] S. Tabassum and J. Gama, "Evolution analysis of call ego-networks," in *International Conference on Discovery Science*. Springer, 2016, pp. 213–225.
[23] ——, "Sampling evolving ego-networks with forgetting factor," in *Workshop MobDM, Mobile Data Management (MDM), 2016 17th IEEE International Conference on*, 2016.
[24] E. S. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, no. 1/2, pp. 100–115, 1954.
[25] H. Mouss, D. Mouss, N. Mouss, and L. Sefouhi, "Test of page-hinkley, an approach for fault detection in an agro-alimentary production system," in *Proceedings of the Asian control conference*, vol. 2. Citeseer, 2004, pp. 815–818.
[26] J. Gama, R. Sebastião, and P. P. Rodrigues, "On evaluating stream learning algorithms," *Machine learning*, vol. 90, no. 3, pp. 317–346, 2013.
[27] R. Sebastião, M. M. Silva, R. Rabiço, J. Gama, and T. Mendonça, "Real-time algorithm for changes detection in depth of anesthesia signals," *Evolving Systems*, vol. 4, no. 1, pp. 3–12, 2013.
[28] F. S. Pereira, J. a. Gama, S. Amo, and G. M. Oliveira, "On analyzing user preference dynamics with temporal social networks," *Mach. Learn.*, vol. 107, no. 11, pp. 1745–1773, Nov. 2018.
[29] F. S. F. Pereira, S. Amo, and J. Gama, "Evolving centralities in temporal graphs: a twitter network analysis," in *Mobile Data Management (MDM), 2016 17th IEEE International Conference on*, 2016.
[30] P. Holme and J. Saramaki, "Temporal networks," *Physics Reports*, vol. 519, no. 3, pp. 97–125, 2012.
[31] H. Wu, J. Cheng, S. Huang, Y. Ke, Y. Lu, and Y. Xu, "Path problems in temporal graphs," *Proceedings of the VLDB Endowment*, vol. 7, no. 9, pp. 721–732, 2014.
[32] R. Zafarani, M. A. Abbasi, and H. Liu, *Social Media Mining: An Introduction*. New York, NY, USA: Cambridge University Press, 2014.
[33] V. Nicosia, J. Tang, C. Mascolo, M. Musolesi, G. Russo, and V. Latora, *Temporal Networks*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, ch. Graph Metrics for Temporal Networks, pp. 15–40.