

Predicting Critical Events in Dynamic Social Network

Nedha Abdulla. P

Department of Computer Science and Engineering
AWH Engineering College
Kozhikode, India

Sona. P

Department of Computer Science and Engineering
AWH Engineering College
Kozhikode, India

Abstract— A social network is usually conceived as a graph in which individuals in the network are represented by the nodes and the nodes are connected to each other by links which depict the relations among the individuals. The term “community” for any group of nodes that are densely connected among themselves and sparsely connected to others. As time evolves, communities in a social network may undergo various changes (split, expand, shrink, stable, merge) known as critical events. Prediction of critical events is an important and difficult issue in the study of social networks. This paper proposes a sliding window analysis, an autoregressive model and survival analysis techniques. The autoregressive model is here to simulate the evolution of the community structure, and the survival analysis techniques allow the prediction of future changes the community may undergo. In our approach Critical events are treated based on a weighting scheme.

Keywords— *Dynamic Network; Community Evolution; critical events; Predict*

1. INTRODUCTION

A social network is a social structure of people related to each other through a common relationship or interest. Usually, a social network is conceived as a graph in which individuals in the network are represented by the nodes. The relations among the individuals are represented by links. Social network analysis is the interactions between people and groups of people, as well as the associated resources for understanding their behavior. Tracking community structures over time and predicting their future changes has important applications in various domains such as criminology, public health, education.

In a dynamic and evolving nature of online social networks with time, as most often i) new members join the network, ii) existing members leave the network, and iii) members establish/break ties and/or change intensity/weight of interactions with other members. The term “community” for any group of nodes that are densely connected among themselves and sparsely connected to others. A community structure can be drastically affected by changes in nodes and variations in their links, such as appearing and disappearing over time. Hence, from one time point t_i to another time point t_j , $t_j > t_i$, a community can split into several other communities, expand into a larger community, shrink to smaller community or remain constant in the same way, several communities can merge into one community. We call these (split, expand, shrink, stable, merge) critical events which communities may undergo over time.

For learning the evolution of communities over time and predicting the critical events the communities may undergo. this paper proposes a sliding window analysis, an

autoregressive model and survival analysis techniques. The autoregressive model is here to simulate the evolution of the community structure, whereas the survival analysis techniques allow the prediction of future changes the community may undergo.

Sliding window concept is useful to track consistent evolving communities, defining the size of window is a challenge. This paper is to automatically identify the optimal window size in order to address the aforementioned drawbacks of existing approaches.

In the recent literature treat critical events with equal importance. However, in some applications, different communities may have their own life cycles. Critical events should thus be treated accordingly. In our paper critical events are treated based on a weighting scheme, in which the importance of events such as “appear” and “disappear” is not considered equal but based on the dynamics of communities.

Our work can be summarized as follows:

- 1) An approach for automatically detecting the size of the window to adopt when identifying and tracking communities over time. The size of the window here is estimated based on the numbers of nodes appearing, disappearing and remaining in the dynamic network at two consecutive, independent time-stamps
- 2) To obtain the critical events, propose autoregressive modeling and survival analysis. Autoregressive modeling is to simulate the evolution of the community structure, and the survival analysis techniques allow the prediction of future changes the community may undergo.
- 3) Critical events are treated based on a weighting scheme.

2. RELATED WORK

There has been increasing interest in studying the evolution of community structures in dynamic social networks. The important issues are how to track communities, how to discover critical events a community can undergo over time.

S. Y. Bhat [1] propose a unified framework, HOCTracker, for tracking the evolution of hierarchical and overlapping communities in online social networks. It is a density-based approach for detecting overlapping community structures, and automatically tracks evolutionary events like birth, growth, contraction, merge, split, and death of communities. HOCTracker adapts a preliminary community structure (identified through a novel density-based overlapping community detection approach) to the changes occurring in a network and processes only active nodes for the new time step. But Time complexity is a Challenge.

P. Lee [2] model social streams as dynamically evolving post networks and model events as clusters over these networks, obtained by means of a clustering approach that is robust to the large amount of noise present in social streams. Typical cluster evolution patterns include birth, death, growth, decay, merge and split. Event detection can be viewed as a subproblem of cluster evolution tracking in social streams. It does not predict the future events.

N. Du [3] define that a community is with high strength if it has relatively stronger internal interactions connecting its members than the external interactions with the members to the rest of the world. community strength analysis discovering how the strength of each detected community changes over the entire observation period. framework that provides reliable and consistent community strength scores. But the information provided by these studies is limited to only adjacent snapshots which cannot give us a whole picture of the community evolution.

E.G.Tajeuna [4] presented a new framework to track community structures in time-evolving social networks and to detect changes that may occur in communities. Then propose a new similarity measure, named mutual transition, for tracking the communities and rules for capturing significant transition events a community can undergo. This framework is not capable of predicting the future transitions a community may undergo.

Xiujuan Xu[5] describes the problems of dynamic social network using data mining theory and introduces a novel dynamic social network algorithm called iDBMM based on the improvement of dynamic behavioral mixed membership model algorithm (DBMM). iDBMM algorithm classifies the training set to obtain the basic characteristics of each role. Then it scores the test set relative to each role and distribute the role of the highest score to the corresponding node. Finally, the transition model is obtained by the statistical method. The experimental results are largely affected by selected characteristics. If the characteristic differences between the roles are too large, the conversion between roles (except for its owning conversion) tends to 0. If the characteristic differences between the roles are too small, the error will appear in the allocation of roles.

3. PROBLEM DEFINITION

Learning the evolution of communities over time is a key step towards predicting the critical events the communities may undergo. This is an important and difficult issue in the study of social networks. In the work to date, there is a lack of formal approaches for modeling and predicting critical events over time and treat the critical events with equal importance.

4. PROPOSED SYSTEM

Here various theories and techniques for (1) tracking communities, (2) estimating feature values by vector autoregression (VAR) and (3) predicting critical events by survival analysis.

4.1 Notation

Tracking communities means is to align communities at different time points in such way as to represent an evolution. For instance, a sequence $S = \{C_{t1}; C_{t2}; C_{t3}; C_{t4}\}$ is considered as an evolution of community C_{t1} if all communities $C_{ti}, C_{ti+1} \in S$ are similar in terms of nodes. Note that in most

cases the similarity is given in terms of nodes shared. An evolving community may undergo critical events during its evolution. Understanding how these occur requires an analysis of the past history of the topological features in relation to the critical events. Model-based approaches such as VAR could thus be used to generate feature values, VARs are multi-linear autoregressive models in which each vector observation is represented as a combination of previous observations considering Y_n to be a random d-dimensional vector observed at time t_n , this vector can be expressed as a linear combination of the p lag vectors $Y_{n-1} Y_{n-2} \dots Y_{n-p}$, as follows:

$$Y_n = \epsilon r_n + \sum_{i=1}^p Y_{n-i} M_i \quad (1)$$

where M_i is a d-by-d matrix representing the coefficients (weights) associated with the lag vectors Y_{n-i} and ϵr_n is the additive Gaussian noise with zero mean.

Survival analysis is a statistical method for studying the occurrence and timing of events. Its aim is to estimate, via a probability (generally called the survivor function $S(t)$), the risk of an events occurring given the past history of a set of time-varying observations. Formally, given the risk or hazard ($\lambda(t)$) of an events occurring at a specific time t , the survivor function is given as the cumulative risk over time:

$$S(t) = 1 - \exp\left(-\int_0^t \lambda(t) dt\right) \quad (2)$$

Let us take a dynamic social network from which individual interrelationships are collected at regular time-stamps for a duration going from t_1 to t_m . At any time $t_i (i=1, \dots, m)$, we use the graph structure $g_{t_i} = (V_{t_i}, E_{t_i})$ to represent the snapshot of the social network, where V_{t_i} stands for the set of nodes and E_{t_i} the set of edges. We then use the series $G = \{(V_{t_i}, E_{t_i}) | 1 \leq i \leq m\} = \{g_{t_i}\}_{1 \leq i \leq m}$ to denote the dynamic social network over the whole period $[t_1, t_2]$. We use $\mathcal{D} = \{1, 2, \dots, m-1\}$ to denote the set of interval durations. For a fixed duration $s \in \mathcal{D}$ we use the notation $W^{s \rightarrow}$ to mean a window W of size s that slides from left to right by a step of one time-stamp. It is worth noting that $W^{s \rightarrow}$ can also be represented as the set $\{\{t_1 \dots t_s\}, \{t_2 \dots t_{s+1}\}, \dots, \{t_{m-s+1} \dots t_m\}\} = \{W_1, W_2, \dots, W_{m-s+1}\}$ where each W_j is an instance of $W^{s \rightarrow}$. Hence, the graph instance corresponding to the window instance W_j can be expressed as follows:

$$g_{W_j} = \cup_{i=j}^{j+s-1} g_{t_i} \quad (3)$$

For each graph g_{W_j} we define a partition $\{C_{W_j}^1, C_{W_j}^2, \dots, C_{W_j}^{q_j}\}$ representing the communities detected at window-stamp W_j each detected community $C_{W_j}^l$ has $E_{W_j}^l$ and $V_{W_j}^l$ as its sets of edges and vertices, respectively. S_c to denote the sequence of communities that reflects the evolution of a community C from window-stamp W_j to W_{m-s+1} .

4.2) Determining window size.

Given a sliding window $W^{s \rightarrow}$ moving through the interval $[t_1, t_m]$ for any window-stamp W_j the set of nodes observed at this instance are given by $S_j = \cup_{i=j}^{j+s-1} \{V_{t_i}\}$. At the next window stamp W_{j+1} we have the set of nodes given by $S_{j+1} = \cup_{i=j+1}^{j+s} \{V_{t_i}\}$. At each step in which a window of size s moves from step j to step $j+1$, we then calculate the numbers of nodes that remain (N_r^s), disappear (N_d^s), and appear (N_a^s) in the graph as follows:

$$N_r^s(W_j, W_{j+1}) = |S_j \cap S_{j+1}| \quad (4)$$

$$N_d^s(W_j, W_{j+1}) = |S_j - (S_j \cap S_{j+1})| \\ = |S_j| - N_r^s(W_j, W_{j+1}) \quad (5)$$

$$N_d^s(W_j, W_{j+1}) = |S_{j+1} - (S_j \cap S_{j+1})| \\ = |S_{j+1}| - N_r^s(W_j, W_{j+1}) \quad (6)$$

Note that the larger the value of N_r^s , the more the network tends to be static, which may make it impossible to capture changes such as merge, split, shrink and expand that evolving communities may undergo. In the same way, the smaller the value of N_r^s , the more the network changes over time, which may result in several cases where communities are evolving in a non-consecutive way. Note that the number of nodes remaining in the graph over time, N_r^s increases according to the size of the window. we first calculate the fluctuation fls of the graph given size s of the sliding window, as follows:

$$fl_s = (W_j, W_{j+1}) = \frac{N_d^s(W_j, W_{j+1}) \times N_d^s(W_j, W_{j+1})}{N_r^s(W_j, W_{j+1})^2} \\ = 1 - \frac{|S_j| + |S_{j+1}|}{N_r^s(W_j, W_{j+1})} + \frac{|S_j| \cdot |S_{j+1}|}{N_r^s(W_j, W_{j+1})^2} \quad (7)$$

Note that if the network is static, i.e., when $N_r^s = |S_j| = |S_{j+1}|$, the fluctuation is 0. When the graph is highly changing, $N_r^s = |S_j \cap S_{j+1}| = N \ll |S_j|, |S_{j+1}|$ where N is the minimum number of nodes remaining in the network. Hence the fluctuation is $\frac{|S_j| \cdot |S_{j+1}|}{N}$. Hence, the fluctuation is always bounded in $[0, \frac{|S_j| \cdot |S_{j+1}|}{N}]$

In this work to study the various changes the evolving communities are undergoing, we selected the minimum window size \hat{s} such that the fluctuation fl_s is bounded within $[0, 1]$ with deviation lower than a small value ϵ .

$$\hat{s} = \min_{s \in \mathcal{D}} \{ fl_s \leq 1, \sigma^2(\mathcal{FL}) < \epsilon \} \quad (8)$$

\mathcal{FL} is the set of fluctuations given a sliding window $W^{S \rightarrow}$ and $\sigma(\mathcal{FL})$ its standard deviation. Algorithm 1 to estimate the minimal window size.

4.3) Modeling critical events.

Let $\mathcal{Cr}_e = \{ \text{Split, Merge, Shrink, Expand, Stable} \}$ be the set of critical events an evolving community may undergo. We assume that these events are mutually independent, which means the probability that a community may pass through one event is not affected by another event. In other words, the probability that a community may undergo an event can be evaluated independently of other events. Hence, for a critical event $e \in \mathcal{Cr}_e$ in observing a community evolving over time, we will either see this event occurring or we will not. two possible responses will then be recorded when observing evolving communities: either the event e occurs (codified as 1) or it does not (codified as 0). We generalize this process by modeling the instantaneous risk that a community may undergo an event e.

4.3.1) Hazard function.

Given an event e and a sliding window $W^{S \rightarrow}$, for each window stamp $W \in W^{S \rightarrow}$, we calculate the number of times $N_e(W)$ the event e has occurred, which corresponds to the number of evolving communities that passed through the critical event e during the time transition from one window-stamp to the next. number of events (N_e) is affected at each window-stamp by a harmful function M_e .

The counting process can thus be decomposed as follows:

$$N_e(W) = \Lambda_e(W) + M_e(W) \quad (9)$$

Where $\Lambda_e(W)$ is a non-decreasing predictable process, called the cumulative intensity process, and $M_e(W)$ is a mean zero martingale. Considering $\Lambda_e(W)$ to be continuous, there exists a predictable non-negative intensity process $\lambda_e(W)$ such that:

$$\Lambda_e(W) = \int_{w_1}^W \lambda_e(w) dw \quad (10)$$

Given an evolving community S_c at each window stamp, this community may or may not be observed. Hence the intensity of community with regard to the event e at any window-stamp W is

$$\lambda_e(W|S_c) = Y_e(W|S_c) h_e(W|S_c) \quad (11)$$

Where $Y_e(W|S_c)$ takes the value 1 if S_c has an instance at window-stamp W and 0 otherwise $h_e(W|S_c)$ is the intensity or the hazard rate for evolving community S_c undergoing event e at window-stamp W.

Algorithm 1: Determining window size.

```

1: Input: G // The graph at different timestamp
2: Output: Size // The window's size
3: for  $s \in \mathcal{D}$  do
4: size  $\leftarrow s$ 
5:  $\mathcal{FL} \leftarrow \{ \}$  //Initializ the set of fluctuations
6: for  $W_j, W_{j+1} \in W^{S \rightarrow}$  do
7: Resize the graphs  $gw_j gw_{j+1}$ 
8: Calculate the fluctuation using (7) //equation 7
9:  $\mathcal{FL} \leftarrow \mathcal{FL} \cup \{ fl_s \}$ 
10: end for
11: Dev  $\leftarrow \sigma^2(\mathcal{FL})$  // Calculate the variation
12: if Dev <  $10^{-2}$  and  $\forall fl_s \in \mathcal{FL}, 0 \leq fl_s \leq 1$ 
13: break //stop the algorithm
14: end if
15: end for
16: Return Size

```

4.3.2) Probability of an event occurring

The intensity process defined in (11) gives the risk that a community will pass through an event at a given time point. From this, we can then calculate, at any time point, the probability that a community will undergo an event by computing the cumulative probability $S_c(W_b|S_c)$.

$$S_c(W_b|S_c) = 1 - \exp \left\{ - \int_{w_1}^{w_b} Y_e(\omega|S_c) h_e(\omega|S_c) d\omega \right\} \quad (12)$$

Consider ϵ a new d-dimensional space. The features extracted from a given community $C_{w_b} \in S_c$ by the vector $X_{w_b} = (x_{1,b}, x_{2,b}, \dots, x_{d,b})$. To calculate the risk at a more distant unobservable time we need to predict the feature values at that unobservable time. For this, we assume that the compositional data vector X_{w_b} can be written as a linear combination of the p lag compositional vectors.

$$X_{w_b} = \epsilon r_b + \sum_{a=1}^p X_{w_{b-a}} \Omega_a \quad (13)$$

4.3.3) Estimation of the Cox parameters

Suppose that we observe communities from W_1 to some W_b which is not the last window-stamp. Clearly from W_1 to W_b we do not have a total view of all communities, which implies that the notion of likelihood cannot be explicit. Due to this constraint, we instead define the partial likelihood of the parameters $\bar{\Gamma}_w^e = (\gamma_{w_1}^e \dots \gamma_{w_b}^e)$ as

$$\text{Log}_p(\bar{\Gamma}_w^e) = \sum_{s_c} \sum_{i=1}^b \delta w_i \{X_{w_i}^* \Gamma_{w_i}^e - \text{Log}[\sum_{s_c} Y_e(W_i | S_c) \exp(X_{w_i}^* \Gamma_{w_i}^e)]\} \quad (14)$$

Where δw_i is a binary indicator which takes value 1 when community S_c has an instance at window-stamp W_i and 0 otherwise.

Having obtained the partial log-likelihood, we can approximate the parameter $\Gamma_{w_i}^e$ by solving the following recursive equation:

$$(\bar{\Gamma}_w^e)^{(it+1)} = (\bar{\Gamma}_w^e)^{(it)} - \left\{ \text{Log}_p'' \left((\bar{\Gamma}_w^e)^{(it)} \right) \right\} \text{Log}_p' \left((\bar{\Gamma}_w^e)^{(it)} \right) \quad (15)$$

where (it) denotes the current iteration step, $\text{Log}_p''(\cdot)$ corresponds to the second derivative of the partial log-likelihood, and $\text{Log}_p'(\cdot)$ to the first derivative of the partial loglikelihood.

4.4) Predicting an event

Once the parameters of our model have been estimated and the hazard function identified, we can calculate the probability that an evolving community will undergo a critical event at any time (observable or not) by applying (12). However, at each more distant unobservable time, all the parameters need to be re-estimated in order to calculate the new probability value. At any unobservable time, we thus predict that a given community will undergo an event e if its probability (12) of occurrence has the highest value compared to the probability of occurrence of the other events. Hence, assuming that we have observed the evolution of communities from t_1 to t_{b+s-1} . which corresponds to the set of observable window-stamps $O_w = \{W_1 \dots W_b\}$, at later unobservable times corresponding to the set of unobservable window-stamps $\hat{O}_w = \{W_{b+1}, W_{b+2}, \dots, W_{b+k} \dots\}$, we can run Algorithm 2 to predict critical events at more distant times.

Algorithm 2: Predicting critical events

```

1: Input:  $E_c, Cr_e, O_w, \hat{O}_w$  // Sets of evolving communities,
critical events, observable and non-observable window-
stamps.
2: Output:  $Pr_c$  //Set of predicted events
3: Initialization
  •  $M \leftarrow O_w$ , //First period used to estimate the
parameters
  •  $Pr_c \leftarrow \{\}$ 
  •  $P_{ar} \leftarrow \{\}$ 
4: Estimating parameters
  • (a) Calculate the VAR parameters
  • (b) Estimate the cox parameters at the period M
for each events using (15) //equation 15
  •  $P_{ar} \leftarrow P_{ar} \cup \{(a), (b)\}$ 
5: Updating & Predicting
  • Updating
For each evolving community  $S_{c_{w_j}} \in E_c$ , at each
window stamp  $W \in \hat{O}_w$ 
  >  $M \leftarrow M \cup \{M\}$  //Update period
  > Calculate the community's features using(13) at
W //Update features values at
next window-stamp,
  > Run step 4 to update parameters
  • Predicting
For each evolving community  $S_{c_{w_j}} \in E_c$ , at each
window stamp  $W \in \hat{O}_w$  and for each event  $e \in Cr_e$ 
  > calculate the probability  $S_e$  using(12)
  >  $Pr_c \leftarrow Pr_c \cup \max \{S_e, e \in Cr_c\}$  // select
the event having highest  $S_e$ .

```

4.5) Assigning priority to Critical events.

It is a method of assigning weights, which applies hierarchy structure of analytic hierarchy process and pairwise comparison. This method has advantages that the number of comparisons can be reduced and also consistency is automatically maintained via determination of priorities first on multiple entities and subsequent comparisons between entities with adjoined priorities.

To determine priorities of multiple attributes we use the following steps. The first step to determine priority is to create a hierarchy using attributes and entities. Then setting priorities of entities within each group. After giving priorities to entities in the same group, a priority between the entities with the same priority from the different groups is determined. Thus, A, D, and G (are entities) with the highest priority in groups I, II, and III, respectively, are compared and priorities are given between them. The same practice is repeated for the entities with the second and third priorities from each group, respectively. Finally setting a priority between the entities with adjoined priorities.

Once the priority of the entire entities is determined, the weight for each attribute is assigned. If weights are assigned while this priority is kept unchanged, the consistency is consequently maintained. Thus, comparisons were made between entities of adjoined priorities while the priority was maintained. In this pairwise comparison, the entity with a higher priority is given a high score and that with a lower priority in turn is given a lower relative score.

Algorithm 3: Assigning priority to Critical events.

- 1: Determination of Priorities
 - Create hierarchical structuring
 - setting priorities of entities within each group
 - setting a priority between the entities with the same priority from the different groups
 - setting a priority between the entities with adjoined priorities
- 2: Assigning weights for each attribute.

5. EXPERIMENT SETUP AND RESULT

To evaluate our proposed approach, we first determined the appropriate window size to use. After obtaining the appropriate window size for each of our networks, we detected the communities and tracked them over time in terms of nodes and the extracted features. Finally, to predict the critical events, we first detected these critical events and modeled them.

In our experiment the code takes the `**edge list**` of the graph in a csv file. Every row indicates an edge between two nodes separated by a comma. The first row is a header. Nodes should be indexed starting with 0. Sample graphs for 'Facebook Politicians' and 'Facebook TV Shows' are included in the 'input/' directory.

```
Input and output options
--edge-path STR Edge list csv. Default is 'input/tvshow_edges.csv'.
--features-path STR Membership json. Default is 'output/tvshow_cluster_memberships.json'.
--resolution FLOAT Validation set size. Default is 1.0.
```

Figure 1: input and output options

```
Anacoda Prompt (Anaconda3)
(base) C:\Users\LENOVO\Downloads\EgoSplitting-master
(base) C:\Users\LENOVO\Downloads\EgoSplitting-master\python src/main.py
Edge path | ./input/politician_edges.csv
Output path | ./output/politician_cluster_memberships.json
Resolution | 1
Creating egonets.
Clustering the persona graph.
Edge path | ./input/politician_edges.csv
Output path | ./output/politician_cluster_memberships.json
Resolution | 1
Creating egonets.
Clustering the persona graph.
```

Figure 2: With default resolution 1.0

```
Anacoda Prompt (Anaconda3)
(base) C:\Users\LENOVO\Downloads\EgoSplitting-master\python src/main.py --resolution 2.5
Edge path | ./input/politician_edges.csv
Output path | ./output/politician_cluster_memberships.json
Resolution | 2.500
Creating egonets.
Clustering the persona graph.
Edge path | ./input/politician_edges.csv
Output path | ./output/politician_cluster_memberships.json
Resolution | 2.500
Creating egonets.
Clustering the persona graph.
```

Figure 3: Training a model with higher resolution 2.5

```
Anacoda Prompt (Anaconda3)
(base) C:\Users\LENOVO\Downloads\EgoSplitting-master\python src/main.py --resolution 0.5
Edge path | ./input/politician_edges.csv
Output path | ./output/politician_cluster_memberships.json
Resolution | 0.500
Creating egonets.
Clustering the persona graph.
Edge path | ./input/politician_edges.csv
Output path | ./output/politician_cluster_memberships.json
Resolution | 0.500
Creating egonets.
Clustering the persona graph.
```

Figure 4: Training a model with a lower resolution 0.5

```
Anacoda Prompt (Anaconda3)
(base) C:\Users\LENOVO\Downloads\EgoSplitting-master\python src/main.py --edge-path input/tvshow_edges.csv --output-path output/tvshow_cluster_memberships.json
Edge path | input/tvshow_edges.csv
Output path | output/tvshow_cluster_memberships.json
Resolution | 1
Creating egonets.
Clustering the persona graph.
Edge path | input/tvshow_edges.csv
Output path | output/tvshow_cluster_memberships.json
Resolution | 1
Creating egonets.
Clustering the persona graph.
```

Figure 5: Training a model on the Facebook TV shows dataset

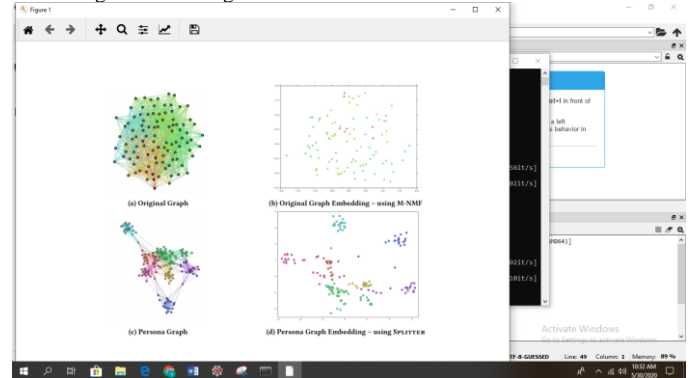


Figure 6: And finally the plot

```
politician_cluster_memberships - filtered
File Edit Format View Help
[*] 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1010, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095, 1096, 1097, 1098, 1099, 1100, 1101, 1102, 1103, 1104, 1105, 1106, 1107, 1108, 1109, 1110, 1111, 1112, 1113, 1114, 1115, 1116, 1117, 1118, 1119, 1120, 1121, 1122, 1123, 1124, 1125, 1126, 1127, 1128, 1129, 1130, 1131, 1132, 1133, 1134, 1135, 1136, 1137, 1138, 1139, 1140, 1141, 1142, 1143, 1144, 1145, 1146, 1147, 1148, 1149, 1150, 1151, 1152, 1153, 1154, 1155, 1156, 1157, 1158, 1159, 1160, 1161, 1162, 1163, 1164, 1165, 1166, 1167, 1168, 1169, 1170, 1171, 1172, 1173, 1174, 1175, 1176, 1177, 1178, 1179, 1180, 1181, 1182, 1183, 1184, 1185, 1186, 1187, 1188, 1189, 1190, 1191, 1192, 1193, 1194, 1195, 1196, 1197, 1198, 1199, 1200, 1201, 1202, 1203, 1204, 1205, 1206, 1207, 1208, 1209, 1210, 1211, 1212, 1213, 1214, 1215, 1216, 1217, 1218, 1219, 1220, 1221, 1222, 1223, 1224, 1225, 1226, 1227, 1228, 1229, 1230, 1231, 1232, 1233, 1234, 1235, 1236, 1237, 1238, 1239, 1240, 1241, 1242, 1243, 1244, 1245, 1246, 1247, 1248, 1249, 1250, 1251, 1252, 1253, 1254, 1255, 1256, 1257, 1258, 1259, 1260, 1261, 1262, 1263, 1264, 1265, 1266, 1267, 1268, 1269, 1270, 1271, 1272, 1273, 1274, 1275, 1276, 1277, 1278, 1279, 1280, 1281, 1282, 1283, 1284, 1285, 1286, 1287, 1288, 1289, 1290, 1291, 1292, 1293, 1294, 1295, 1296, 1297, 1298, 1299, 1300, 1301, 1302, 1303, 1304, 1305, 1306, 1307, 1308, 1309, 1310, 1311, 1312, 1313, 1314, 1315, 1316, 1317, 1318, 1319, 1320, 1321, 1322, 1323, 1324, 1325, 1326, 1327, 1328, 1329, 1330, 1331, 1332, 1333, 1334, 1335, 1336, 1337, 1338, 1339, 1340, 1341, 1342, 1343, 1344, 1345, 1346, 1347, 1348, 1349, 1350, 1351, 1352, 1353, 1354, 1355, 1356, 1357, 1358, 1359, 1360, 1361, 1362, 1363, 1364, 1365, 1366, 1367, 1368, 1369, 1370, 1371, 1372, 1373, 1374, 1375, 1376, 1377, 1378, 1379, 1380, 1381, 1382, 1383, 1384, 1385, 1386, 1387, 1388, 1389, 1390, 1391, 1392, 1393, 1394, 1395, 1396, 1397, 1398, 1399, 1400, 1401, 1402, 1403, 1404, 1405, 1406, 1407, 1408, 1409, 1410, 1411, 1412, 1413, 1414, 1415, 1416, 1417, 1418, 1419, 1420, 1421, 1422, 1423, 1424, 1425, 1426, 1427, 1428, 1429, 1430, 1431, 1432, 1433, 1434, 1435, 1436, 1437, 1438, 1439, 1440, 1441, 1442, 1443, 1444, 1445, 1446, 1447, 1448, 1449, 1450, 1451, 1452, 1453, 1454, 1455, 1456, 1457, 1458, 1459, 1460, 1461, 1462, 1463, 1464, 1465, 1466, 1467, 1468, 1469, 1470, 1471, 1472, 1473, 1474, 1475, 1476, 1477, 1478, 1479, 1480, 1481, 1482, 1483, 1484, 1485, 1486, 1487, 1488, 1489, 1490, 1491, 1492, 1493, 1494, 1495, 1496, 1497, 1498, 1499, 1500, 1501, 1502, 1503, 1504, 1505, 1506, 1507, 1508, 1509, 1510, 1511, 1512, 1513, 1514, 1515, 1516, 1517, 1518, 1519, 1520, 1521, 1522, 1523, 1524, 1525, 1526, 1527, 1528, 1529, 1530, 1531, 1532, 1533, 1534, 1535, 1536, 1537, 1538, 1539, 1540, 1541, 1542, 1543, 1544, 1545, 1546, 1547, 1548, 1549, 1550, 1551, 1552, 1553, 1554, 1555, 1556, 1557, 1558, 1559, 1560, 1561, 1562, 1563, 1564, 1565, 1566, 1567, 1568, 1569, 1570, 1571, 1572, 1573, 1574, 1575, 1576, 1577, 1578, 1579, 1580, 1581, 1582, 1583, 1584, 1585, 1586, 1587, 1588, 1589, 1590, 1591, 1592, 1593, 1594, 1595, 1596, 1597, 1598, 1599, 1600, 1601, 1602, 1603, 1604, 1605, 1606, 1607, 1608, 1609, 1610, 1611, 1612, 1613, 1614, 1615, 1616, 1617, 1618, 1619, 1620, 1621, 1622, 1623, 1624, 1625, 1626, 1627, 1628, 1629, 1630, 1631, 1632, 1633, 1634, 1635, 1636, 1637, 1638, 1639, 1640, 1641, 1642, 1643, 1644, 1645, 1646, 1647, 1648, 1649, 1650, 1651, 1652, 1653, 1654, 1655, 1656, 1657, 1658, 1659, 1660, 1661, 1662, 1663, 1664, 1665, 1666, 1667, 1668, 1669, 1670, 1671, 1672, 1673, 1674, 1675, 1676, 1677, 1678, 1679, 1680, 1681, 1682, 1683, 1684, 1685, 1686, 1687, 1688, 1689, 1690, 1691, 1692, 1693, 1694, 1695, 1696, 1697, 1698, 1699, 1700, 1701, 1702, 1703, 1704, 1705, 1706, 1707, 1708, 1709, 1710, 1711, 1712, 1713, 1714, 1715, 1716, 1717, 1718, 1719, 1720, 1721, 1722, 1723, 1724, 1725, 1726, 1727, 1728, 1729, 1730, 1731, 1732, 1733, 1734, 1735, 1736, 1737, 1738, 1739, 1740, 1741, 1742, 1743, 1744, 1745, 1746, 1747, 1748, 1749, 1750, 1751, 1752, 1753, 1754, 1755, 1756, 1757, 1758, 1759, 1760, 1761, 1762, 1763, 1764, 1765, 1766, 1767, 1768, 1769, 1770, 1771, 1772, 1773, 1774, 1775, 1776, 1777, 1778, 1779, 1780, 1781, 1782, 1783, 1784, 1785, 1786, 1787, 1788, 1789, 1790, 1791, 1792, 1793, 1794, 1795, 1796, 1797, 1798, 1799, 1800, 1801, 1802, 1803, 1804, 1805, 1806, 1807, 1808, 1809, 1810, 1811, 1812, 1813, 1814, 1815, 1816, 1817, 1818, 1819, 1820, 1821, 1822, 1823, 1824, 1825, 1826, 1827, 1828, 1829, 1830, 1831, 1832, 1833, 1834, 1835, 1836, 1837, 1838, 1839, 1840, 1841, 1842, 1843, 1844, 1845, 1846, 1847, 1848, 1849, 1850, 1851, 1852, 1853, 1854, 1855, 1856, 1857, 1858, 1859, 1860, 1861, 1862, 1863, 1864, 1865, 1866, 1867, 1868, 1869, 1870, 1871, 1872, 1873, 1874, 1875, 1876, 1877, 1878, 1879, 1880, 1881, 1882, 1883, 1884, 1885, 1886, 1887, 1888, 1889, 1890, 1891, 1892, 1893, 1894, 1895, 1896, 1897, 1898, 1899, 1900, 1901, 1902, 1903, 1904, 1905, 1906, 1907, 1908, 1909, 1910, 1911, 1912, 1913, 1914, 1915, 1916, 1917, 1918, 1919, 1920, 1921, 1922, 1923, 1924, 1925, 1926, 1927, 1928, 19
```

6.CONCLUSION

Learning the evolution of communities over time is a key step towards predicting the critical events the communities may undergo. This is an important and difficult issue in the study of social networks. In the work to date, there is a lack of formal approaches for modeling and predicting critical events over time. In this paper we propose a model which predicts the critical events the communities may undergo. The sliding window analysis to tracking communities over time. Then autoregressive modeling and survival analysis predict not only the next event an evolving community may undergo, but future events more distant in time. The current literature treats critical events with equal importance. The advantage of our paper is we use a weighting scheme in which the importance of events such as “appear” and “disappear” is not considered equal but based on the dynamics of communities.

REFERENCES

- [1] S. Y. Bhat and M. Abulaish, “Hoctracker: Tracking the evolution of hierarchical and overlapping communities in dynamic social networks,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 4, pp. 1019–1013, 2015
- [2] P. Lee, L. V. Lakshmanan, and E. E. Milios, “Incremental cluster evolution tracking from highly dynamic network data,” in *Proceedings of the IEEE 30th International Conference on Data Engineering (ICDE)*, 2014, pp. 3–14
- [3] N. Du, X. Jia, J. Gao, V. Gopalakrishnan, and A. Zhang, “Tracking temporal community strength in dynamic networks,” *IEEE Transactions on Knowledge & Data Engineering*, no. 1, pp. 1–1, 2015.
- [4] E.G.Tajeuna, M.Bouguessa, and S.Wang, “Tracking the evolution of community structures in time-evolving social networks,” in *Proceedings of the IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2015, pp. 1–10.
- [5] X.Xu, WeiWang, YuLiu, HongYu, XiaoweiZhao, “iDBMM: a Novel Algorithm to Model Dynamic Behavior in Large Evolving Graphs” in *2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing*.