Prediction of Vital Actions in Active Social Networking Sites using Artificial Intelligence

Mr. Archit Tiwari
Student, Amity University Chhattisgarh
architsgolu@gmail.com

Mr. Mohammed Bakhtawar Ahmed
Assistant Professor, Amity University Chhattisgarh
bakhtawar229@gmail.com

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 ti to another tj, tj >ti, 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 verlapping 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 dosnot 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 member stotheres to fthe 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 notcapable of predicting the future transitions a community may undergo.

XiujuanXu[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 SYSYTEM

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

4.1 Notation

Trackingcommunities means is to align communities at differenttimepointsinsuchawayastorepresentanevolution. For instance, a sequence S $=\{\boldsymbol{C_{t1}};\boldsymbol{C_{t2}};\boldsymbol{C_{t3}};\boldsymbol{C_{t4}}\}$ is considered as an evolution of community $\boldsymbol{C_{t1}}$ if all communities $\boldsymbol{C_{ti}},\boldsymbol{C_{ti+1}} \in \boldsymbol{S}$ 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 observationsconsidering $\boldsymbol{Y_n}$to be a random d-dimensional vector observed at time tn, this vector can be expressed as a linear combination of the *p* lag vectors$\boldsymbol{Y_{n-1}Y_{n-2}}\ldots\ldots\boldsymbol{Y_{n-p}}$, as follows:

$$\boldsymbol{Y_n} = \boldsymbol{\varepsilon r_n} + \sum_{i=1}^{p} \boldsymbol{Y_{n-i}}\,\boldsymbol{M_i} \qquad (1)$$

where $\boldsymbol{M_i}$ is a d-by-d matrix representing the coefficients (weights) associated with the lag vectors $\boldsymbol{Y_{n-i}}$ and$\boldsymbol{\varepsilon r_n}$ 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()), the risk of an events occurringgiventhepasthistoryofasetoftime-varyingobservations. Formally, given the risk or hazard (λ(t)) of an events occurring at a specific time t, the survivor function is given as the cumulative risk over time:

$$\mathrm{S(t)}=1-\boldsymbol{exp}\left(-\int_0^t \boldsymbol{\lambda(t)d(t)}\right) \qquad (2)$$

Let us take a dynamic social network from which individual interrelationships are collected at regular time-stamps for a duration going from $\boldsymbol{t_1}$ to $\boldsymbol{t_m}$tm. At any time $\boldsymbol{t_i}$(i=1,...,m), we use the graph structure $\boldsymbol{gt_i} = (\boldsymbol{Vt_i},\boldsymbol{Et_i})$ to represent the snapshot of the social network, where $\boldsymbol{Vt_i}$ stands for the set of nodes and $\boldsymbol{Et_i}$ the set of edges. We then use the series $\mathrm{G}=\{(\boldsymbol{Vt_i},\boldsymbol{Et_i})|\boldsymbol{1 \le i \le m}\} = (\boldsymbol{gt_i})_{\boldsymbol{1 \le i \le m}}$ to denote the dynamic social network over the whole period $[\boldsymbol{t_1},\boldsymbol{t_2}]$. We use $\boldsymbol{\mathcal{D}} = \{\boldsymbol{1},\boldsymbol{2},\ldots.\boldsymbol{m-1}\}$ to denote the set of interval durations. For a fixed duration s$\in \boldsymbol{\mathcal{D}}$ we use the notation $\boldsymbol{W^{S\rightarrow}}$ 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 $\boldsymbol{W^{S\rightarrow}}$can also be represented as the set $\{\{\boldsymbol{t_1}\ldots.\boldsymbol{t_s}\{\boldsymbol{t_2}\ldots.\boldsymbol{t_{s+1}}\}\ldots..\{\boldsymbol{t_{m-s+1}}\ldots.\boldsymbol{t_m}\}\} = \{\boldsymbol{W_1},\boldsymbol{W_2},\ldots\ldots\boldsymbol{W_{m-s+1}}\}$ where each $\boldsymbol{W_j}$is an instance of $\boldsymbol{W^{S\rightarrow}}$.Hence, the graph instance corresponding to the window instance $\boldsymbol{W_j}$ can be expressed as follows:

$$gw_t = \cup_{i=j}^{j+s-1} gt_i \quad (3)$$

For each graph gw_jwe define a partition$\left\{C_{wj}^1, C_{WJ}^2 \ldots C_{wj}^{qj}\right\}$ representing the communities detectedat window-stamp $W_j$each detected community C_{wj}^lhas E_{wj}^l and V_{wj}^las 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=$\cup_{i=j}^{j+s-1}\{Vt_i\}$.At the next window stamp W_{j+1} we have the set of nodes given by S_{j+1}=$\cup_{i=j}^{j+s}\{Vt_i\}$. 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 thefluctuation fls ofthegraphgivenasize s oftheslidingwindow, as follows:

$$fl_s = (W_j, W_{j+1})=\frac{N_a^s(W_j,W_{j+1})\times N_d^s(W_j,W_{j+1})}{N_r^s(W_j,W_{j+1})^2}$$

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

Note that if the network is static, i.e., when N_r^s=$|S_j|$=$|S_{j+1}|$,thefluctuationis 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 tostudythevariouschangestheevolvingcommunities 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.

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

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

4.3)Modeling critical events.

LetCr_e={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 ϵCr_ein 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 $\boldsymbol{W^{S\rightarrow}}$,for each window stamp W $\boldsymbol{\epsilon W^{S\rightarrow}}$,we calculate the number of times $\boldsymbol{N_e(W)}$ 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 ($\boldsymbol{N_e}$) is affected at each window-stamp by a harmful function $\boldsymbol{M_e}$.

The counting process can thus be decomposed as follows:

$$\boldsymbol{N_e(W)}=\wedge_{\boldsymbol{e}}\boldsymbol{(W)}+\boldsymbol{M_e(W)} \quad (9)$$

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

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

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

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

Where $\boldsymbol{Y_e(W|S_c)}$ takes the value 1 if $\boldsymbol{S_c}$ has an instance at window-stamp W and 0 otherwise $\boldsymbol{h_e(W|S_c)}$ is the intensity or the hazard rate for evolving community $\boldsymbol{S_c}$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∈ $\boldsymbol{\mathcal{D}}$ do
4: size← $\boldsymbol{s}$
5: $\boldsymbol{\mathcal{F}l}$ ← {} //Initializ the set of fluctuations
6: for $\boldsymbol{W_j, W_{j+1}} \in \boldsymbol{W^{S\rightarrow}}$ do
7: Resize the graphs$\boldsymbol{gw_j gw_{j+1}}$
8: Calculate the fluctuation using (7) //equation 7
9: $\boldsymbol{\mathcal{F}l \leftarrow \mathcal{F}l \cup \{fl_s\}}$
10: end for
11: Dev← $\boldsymbol{\sigma^2(\mathcal{F}l)}$// Calculate the variation
12:if Dev< $\boldsymbol{10^{-2}}$ and ∀ $\boldsymbol{fl_s} \in \boldsymbol{\mathcal{F}l}$,0≤ $\boldsymbol{fl_s \leq 1}$
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$\boldsymbol{S_c(W_b|S_c)}$.

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

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

$$\boldsymbol{Xw_b = \varepsilon r_b + \sum_{a=1}^{p} Xw_{b-a}\Omega_a} \quad (13)$$

4.3.3) Estimation of the Cox parameters

Suppose that we observe communities from $\boldsymbol{W_1}$to some $\boldsymbol{W_b}$ which is not the last window-

stamp.Clearly from $\boldsymbol{W_1}$to$\boldsymbol{W_b}$ 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 $\overline{\boldsymbol{\Gamma}}^{\boldsymbol{e}}_{\boldsymbol{w}} = (\boldsymbol{\gamma}^{\boldsymbol{e}}_{\boldsymbol{w1}} \ldots \boldsymbol{\gamma}^{\boldsymbol{e}}_{\boldsymbol{wb}})$ as

Algorithm 2: Predicting critical events

1: Input: E_c, Cr_e,O_w,$\widehat{O}_w$ // Sets of evolving communities, critical events, observable and non-observable window-stamps.
2: Output:Pr_c //Set of predicted events
3: Initialization
- M← O_w, //First period used to estimate the parameters
- Pr_c ←{}
- P_{ar} ← {}

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} ← P_{ar} ∪ {{(a), (b)}}

5:Updating & Predicting
- Updating

For each evolving community$S_{c^1_{w_j}} \in E_c$,at each window stamp W∈ $\widehat{O}_w$
- M←M ∪{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$S_{c^1_{w_j}} \in E_c$,at each window stamp W∈ $\widehat{O}_w$ and for each event e∈Cr_e
- calculate the probability S_e using(12)
- Pr_c ← Pr_c ∪ max {S_e, e|e ∈ Cr_c} // select the event having highest S_e.

$$\boldsymbol{\mathcal{L}og_p}(\overline{\boldsymbol{\Gamma}}^{\boldsymbol{e}}_{\boldsymbol{w}}) = \sum_{\boldsymbol{s_c}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{b}} \boldsymbol{\delta w_i} \{\boldsymbol{X}^{*}_{\boldsymbol{w_i}} \boldsymbol{\Gamma}^{\boldsymbol{e}}_{\boldsymbol{w_i}} -$$

$$\boldsymbol{\mathcal{L}og}[\ \textstyle\sum_{\boldsymbol{S_c}} \boldsymbol{Y_e}(\boldsymbol{W_i}\,|\boldsymbol{S_c}) \boldsymbol{exp}\ (\boldsymbol{X}^{*}_{\boldsymbol{w_i}} \boldsymbol{\Gamma}^{\boldsymbol{e}}_{\boldsymbol{w_i}}) \qquad (14)$$

Where $\boldsymbol{\delta w_i}$ is a binary indicator which takes value 1 when community $\boldsymbol{S_c}$ has an instance at window-stamp $\boldsymbol{W_i}$and 0 otherwise.

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

$$(\overline{\boldsymbol{\Gamma}}^{\boldsymbol{e}}_{\boldsymbol{W}})^{(\boldsymbol{it+1})} = (\overline{\boldsymbol{\Gamma}}^{\boldsymbol{e}}_{\boldsymbol{W}})^{(\boldsymbol{it})} - \left\{\boldsymbol{\mathcal{L}og}^{''}_{\boldsymbol{p}}\left((\overline{\boldsymbol{\Gamma}}^{\boldsymbol{e}}_{\boldsymbol{W}}{}^{(\boldsymbol{it})})\right) \boldsymbol{\mathcal{L}og}^{'}_{\boldsymbol{p}}\left((\overline{\boldsymbol{\Gamma}}^{\boldsymbol{e}}_{\boldsymbol{W}}{}^{(\boldsymbol{it})})\right)\right\} \quad (15)$$

where (it) denotes the current iteration step,$\boldsymbol{\mathcal{L}og}^{''}_{\boldsymbol{p}}(.)$ corresponds to the second derivative of the partial log-likelihood, and $\boldsymbol{\mathcal{L}og}^{'}_{\boldsymbol{p}}(.)$ 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 $\boldsymbol{t_1}$to $\boldsymbol{t_{b+s-1}}$.which corresponds to the set of observable window-stamps $\boldsymbol{O_w}$={$\boldsymbol{W_1} \ldots . \boldsymbol{W_b}$}, at later unobservable times corresponding to the set of unobservable window-stamps $\widehat{\boldsymbol{O}}_{\boldsymbol{w}}$={$\boldsymbol{W_{b+1}}, \boldsymbol{W_{b+2}} \ldots . \boldsymbol{W_{b+k}} \ldots$},

we can run Algorithm 2 to predict critical events at more distant times.

4.5) Assigning priority to Criticalevents.

It is amethodofassigningweights, whichapplieshierarchystructureofanalytichierarchyprocessandpairwisecomparison.Thismethodhas advantagesthatthenumberofcomparisonscanbereducedandalsoconsistencyisautomaticallymaintainedviad

eterminationof prioritiesfirstonmultipleentitiesandsubsequentcomparisonsbetweenentitieswithadjoinedpriorities.

To determine priorities of multipleattributes 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. Aftergivingprioritiesto entitiesinthesamegroup,aprioritybetweentheentities with the same priority from the different groups is determined. Thus, A, D, and G(are entites) 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 entitieswiththesecondandthirdprioritiesfromeachgroup, respectively.finaly setting a priority between the entities with adjoined priorities.

Oncethepriorityoftheentireentities 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

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.

thispairwisecomparison,theentitywithahigherpriorityisgiven a high scoreand that with a lower priority in turnisgivenalowerrelativescore.

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 experimentthe 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.

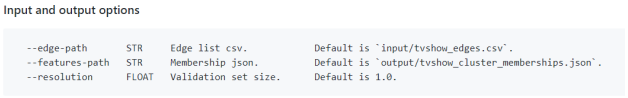

Figure 1:input and output options

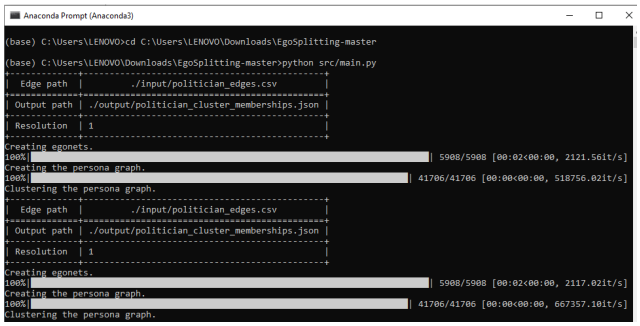

Figure 2:With default resolution 1.0

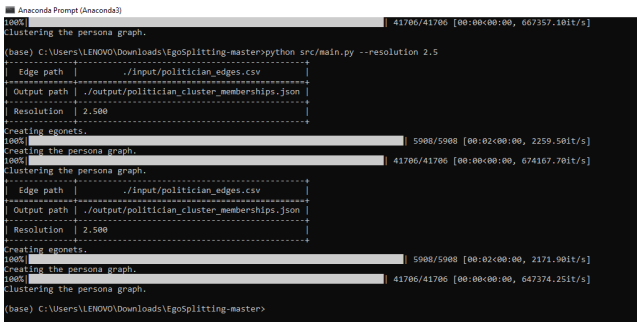

Figure 3: Training a model with higher resolution 2.5

Figure 4:Training a model with a lower resolution 0.5

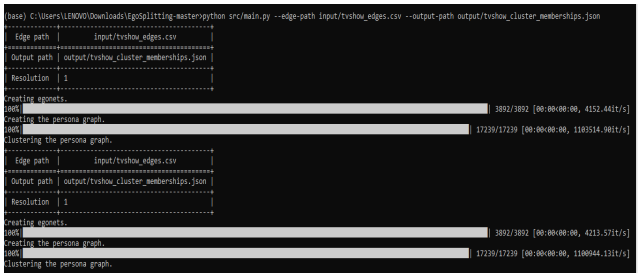

Figure 5: Training a model on the Facebook TV shows dataset

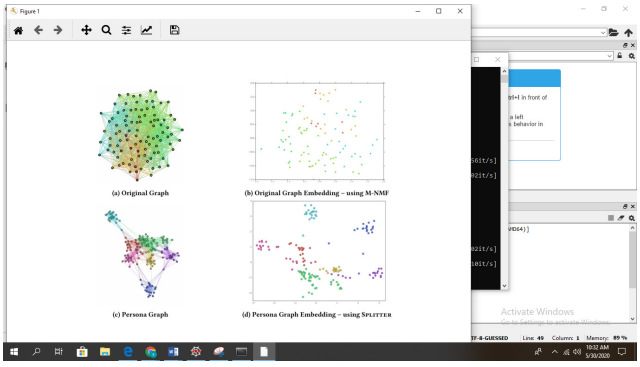

Figure 6:And finally the plot

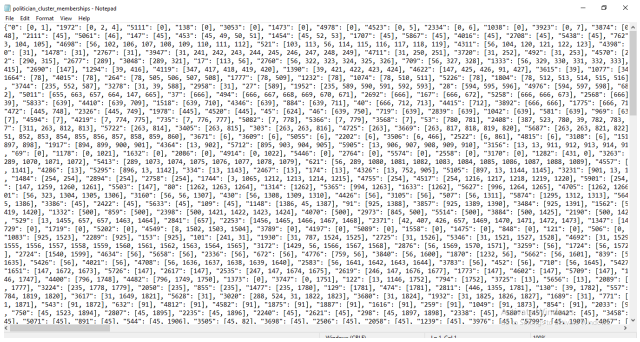

Figure 7: The output get saved as a json file as shown below

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 proposes a model which predicting 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 treat 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,andS.Wang,“Tracking the evolution of community structures in time-evolving social networks,”inProceedings of the IEE International Conference on Data Science and Advanced Analytics (DSAA), 2015, pp. 1–10.

[5]X.Xu, WeiWang, YuLiu,HongYu, XiaoweiZhao,” iDBMM: aNovel Algorithm to Model Dynamic BehaviorinLargeEvolvingGraphs”in 2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing.