Localizing Temporal Anomalies in Large Evolving Graphs

Teng Wang* Chunsheng Fang† Derek Lin† S. Felix Wu‡

Abstract

Mining for anomalies in graph structured datasets is an important and challenging problem for many applications including security, health care, and social media. In this paper, we propose a novel framework to localize temporal anomalies in large evolving graphs with reduced false alarm rate. Specifically, we first introduce a node-centric model based on Vector Autoregression to analyze node behavior history in dynamic graphs. Then we develop two community-centric models to reduce the amount of false positive results by tracking the structural change and dynamics of graph communities. We analyze the performance of our proposed anomaly localization framework on several synthetic and real-world data sets including Enron email network data, an enterprise network traffic data, and CNN public Facebook page. All experimental results show the effectiveness and consistency of our framework in localizing temporal anomalies with reduced false alarm rate.

1 Introduction

Time-evolving graphs often record dynamic transactions among a set of entities, such as enterprise web traffic, personal email communication, and online social network interactions [1]. Temporal anomalies are defined as graph-based outliers present in one or more instances of dynamic graphs. They are graph entities with unusual behavior patterns hidden inside the dynamic graph structure. Localizing temporal anomalies provides the basis to address several application-specific tasks such as insider threat detection in enterprise networks and fraud detection in financial services [2].

Although some recent research has addressed the problem of anomaly detection in dynamic graphs [3, 4], these mostly focus on event detection, i.e. detecting when a temporal anomaly occurs during a graph evolving process, and do not address the problem of localization, i.e. identifying the specific nodes that are responsible for the detected anomalous changes in graph structure. In addition, because most recent work [5, 6] leverages only node-centric models to detect temporal anomalies, many non-anomalous nodes may be included due to normal community-level changes, thus increasing the false alarm rate. For example, in an enterprise network, group membership changes are often normal behavior during cross-team collaboration. In a social network, a sudden increase in a user's communication should not be regarded as anomalous if most of his friends are actively talking about a sudden breaking news story. Motivated by the above two challenges, we propose a new framework for localizing temporal anomalies with reduced false alarm rate. The main contributions of our work are summarized as follows:

- We propose a novel method based on a Vector Autoregression (VAR) model to localize temporal anomalies in dynamic graphs. Instead of modelling the entire graph structure at each timestamp, our algorithm localizes anomalous nodes by tracking the evolution of a set of node properties and their interactions across the whole timeline.
- We design a new anomaly localization framework with reduced false alarm rate by leveraging both a node-centric model and a community-centric model. Using dynamic community paths, we can accurately recognize normal community-level transitions and exclude many false positive results.
- We evaluate the performance of our framework on several synthetic and real-world datasets covering different application areas including Enron email network data (personal communication), an enterprise network traffic data, and CNN public Facebook page (social media). All our experiments show the effectiveness and consistency of our framework in localizing temporal anomalies with reduced false alarm rate.

2 Related Work

Although there is much work on detecting temporal anomalies in dynamic graphs, these methods [7, 8] are mostly designed only for detecting whether or not a graph structure change is anomalous. Few of them focus

*Department of Computer Science, University of California, Davis. tewang@ucdavis.edu. Part of this work was done while this author was working at Pivotal Software Inc.

†Pivotal Software. cfang@pivotal.io, dereklinct@gmail.com

‡Department of Computer Science, University of California, Davis. sfwu@ucdavis.edu

on the problem of anomaly localization, i.e. identifying the specific graph entities that contribute to the detected anomalous change in graph structure. Kumar et al [6] propose a commute-time-based anomaly detection method to localize anomalous edges by tracking the changes in graph structure and edge weights. Their algorithm can only localize anomalies with respect to the graph transition between two neighboring snapshots. In dynamic graphs, many temporal anomalies are hidden inside the evolutionary path of graph structure changes, which can only be detected with a complete behavior model along the whole timeline. Rossi et al. [5] builds a dynamic behavior mix-membership model to analyze graph evolutionary patterns and localize anomaly nodes. However, their method only focuses on a node-centric model and neglects the community information of graph entities. In this paper, we propose a novel anomaly localization framework with a dynamic behavior model leveraging both node-centric properties and community-centric properties.

Community-based anomaly detection has attracted much attention in recent years. Sun et al. [9] propose a generative model to detect evolutionary communities in heterogeneous networks. However, their work assumes that the communities in adjacent timestamps should be consistent, which ignores those dynamic events where the community size changes significantly along the evolutionary path. Chen et al. [8] develop a parameter-free algorithm to uncover six types of community-based anomalies (grown, shrunken, merged, split, born, and vanished) in evolutionary networks. Gupta et al. [10] introduce the definition of evolutionary community outliers and propose an optimization framework to minimize community matching error across snapshots. However, these methods only focus on using community properties to identify group-level anomalies in dynamic graphs, and neglect their relationship with node-level anomalies. In this paper, we leverage the community properties of graph entities in two directions. First, we develop a community-centric model to identify group-level anomalies among dynamic graph patterns. Second, we combine the node-centric model and community-centric model into an integrated algorithm to reduce the false alarm rate of our anomaly localization framework.

3 Anomaly Localization Framework

Given the graph representation of network data, our goal is to model the dynamic change of graph structure and localize temporal anomalies among the nodes. Figure 1 shows the general idea of our proposed framework with three graph models. *History-based Node Model* is a node-centric model. It uses centrality features to describe the behavior history of a single node in the evolving graph. Both the *Community-based Node Model* and *Evolutionary Community Path Model* are community-centric models. The former emphasizes the community information of a single node at a fixed time slot, and the latter tracks the dynamics of graph communities along the whole timeline.

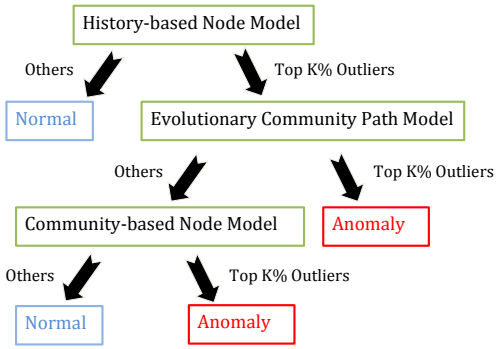


Figure 1: Our Framework for Anomaly Localization

In this section, we first describe the node-centric model based on Vector Autoregression. Then we introduce two community-centric models and their usage on reducing the false alarm rate of anomaly localization. Finally, we combine these three models into an integrated framework.

3.1 Node-Centric Model Given a temporal sequence of graphs $G = \{G_t, t = 1, ..., T\}$, each snapshot G_t is a weighted undirected graph which aggregates all the network connections within the interval $[t-1, t]$. In a graph snapshot, $G_t = (V, E_t)$, $V = \{v_1, ..., v_n\}$ is a fixed node set and E_t is a set of weighted edges during the time slot $[t-1, t]$.

Feature Matrix. We describe the node's observed behavior with a set of representative graph features. In general, we represent each graph snapshot G_t as a feature matrix:

$$F_t = \begin{bmatrix} g_{11,t} & g_{12,t} & \cdots & g_{1n,t} \\ g_{21,t} & g_{22,t} & \cdots & g_{2n,t} \\ & \cdots & & \\ g_{m1,t} & g_{m2,t} & \cdots & g_{mn,t} \end{bmatrix}$$

where $g_{ij,t}$ denotes the ith feature value of the jth node at time t. m is the number of selected features, and n is the number of nodes in this snapshot. Based on previous work [5, 11], we use six graph features to describe the clustering and centrality properties of each node: total degree, clustering coefficient, closeness, eigenvector, betweenness, and PageRank. Therefore, the value of m is 6 in this paper.

History-based Node Model. In order to represent

the behavior history of each node in the dynamic graphs, we use the Vector Autoregression (VAR) model [12] in time series analysis. Because the information of the ith node is stored in the ith column of the feature matrix F_t, we describe the evolution of the ith node's features as follows:

(3.1)
$$F_t^{(i)} = c + A_1 F_{t-1}^{(i)} + A_2 F_{t-2}^{(i)} + ... + A_j F_{t-j}^{(i)} + ... + A_p F_{t-p}^{(i)} + e_t$$

where c is a $m * 1$ vector of constants and e_t is a $m * 1$ vector of error terms. p is the lag order that can be determined automatically by the VAR model. A_j is a $m * m$ matrix describing the node feature transition between time t and $t - j$:

$$A_j = \begin{bmatrix} a_{1,1}^j & a_{1,2}^j & ... & a_{1,m}^j \\ a_{2,1}^j & a_{2,2}^j & ... & a_{2,m}^p \\ ... & & & \\ a_{m,1}^j & a_{m,2}^j & ... & a_{m,m}^j \end{bmatrix}$$

In A_j, we can find that our model not only describes the relationship between the same features at different timestamps, e.g. $a_{1,1}^j$, but also includes the relationship across different features, e.g. $a_{1,2}^j$, which makes our model more powerful than the ordinary ARMA model on tracking multiple graph features in time-evolving graphs.

In the last step, we define the outlier score of each node in the history-based node model as the difference between the predicted feature vector and observed feature vector. To be specific, we use Equation 3.1 to predict the graph feature vector of the ith node at $t + 1$ as $\hat{F}_{t+1}^{(i)}$. Then we compare the predicted $\hat{F}_{t+1}^{(i)}$ to the observed $F_{t+1}^{(i)}$ using the Euclidean norm and get the *History-based Outlier Score* as $O_{hb,(t+1)}^i = ||\hat{F}_{t+1}^{(i)} - F_{t+1}^{(i)}||$.

3.2 Community-Centric Models In order to localize temporal anomalies in dynamic graphs, we not only consider the behavior history of a single node, but refer to its community information as well. In this paper, we propose two community-centric models: *Community-based Node Model* and *Evolutionary Community Path Model*. The former emphasizes the community information of a single node at a fixed time slot, and the latter tracks the structural change and dynamics of graph communities along the whole timeline.

3.2.1 Evolutionary Community Path Model Similar with the history-based node model, we define a temporal sequence of graphs $G = \{G_t, t = 1, ..., T\}$, where the snapshot $G_t = \{V, E_t\}$ is a weighted undirected graph which aggregates all the network connections within the interval $[t - 1, t]$.

Community Detection. Our objective is to do a graph partition within a graph snapshot G_t so that edges between groups have a very low weight and edges within one group have a high weight. In this paper, we use a fast greedy algorithm [13] to detect non-overlapping communities based on the greedy optimization of modularity. Although this fast greedy algorithm can only give us a locally optimal partition of the snapshot, it is very efficient and runs in essentially linear time on some real-world networks. Considering the huge amount of node interaction data and the requirement of real-time analysis, this fast greedy algorithm is the most suitable algorithm for our framework.

Evolutionary Community Path. At this step, we connect communities detected at adjacent time stamps together if the number of their common nodes is above a certain threshold. Suppose we find a set of communities $\mathbb{C}_t = \{C_{t,1}, C_{t,2}, ..., C_{t,x}\}$ in graph snapshot G_t and their predecessors $\mathbb{C}_{t-1} = \{C_{(t-1),1}, C_{(t-1),2}, ..., C_{(t-1),x'}\}$ in graph snapshot G_{t-1}. To match adjacent subgroups together between $\mathbb{C}_t$ and its predecessors $\mathbb{C}_{t-1}$, the most widely-adopted method is to use Jaccard coefficient [14]. However, this classic definition can only be used for identifying state transition of two communities of similar size. During the evolution of graph communities, we still need to consider other dynamic events where community size changes significantly, such as forming, dissolving, expanding, contracting, splitting, and merging. Although the communities involved in those events should be regarded as well-connected in our framework, the Jaccard coefficient may often report them to be low-similarity communities. In order to deal with this problem, we propose another definition to evaluate community similarity in all possible events:

(3.2)
$$sim(C_{t,a}, C_{(t-1),x}) = \frac{|C_{t,a} \cap C_{(t-1),x}|}{min(|C_{t,a}|, |C_{(t-1),x}|)}$$

With this new definition, communities will be matched if the similarity value exceeds the threshold $\theta \in [0, 1]$. When the size of two adjacent communities does not vary much, this definition keeps the property of the Jaccard coefficient; When dynamic evolution events occur where the community size changes significantly, this definition can match the communities together because of the high proportion of common nodes in the smaller community.

Furthermore, we also add the concept of *gap interval* [15] into our model when building the evolutionary community paths. For a community $C_{t,a}$, we not only consider the community set $\mathbb{C}_{t-1}$ detected at the prior

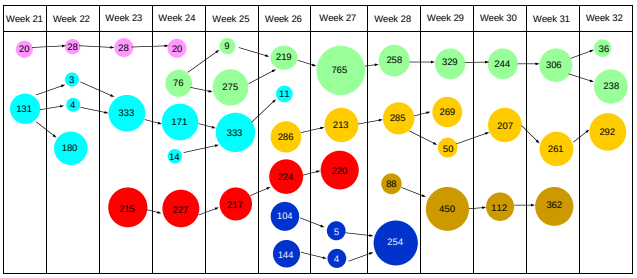


Figure 2: Evolutionary Community Paths in Enron Email Dataset

time stamp $t-1$, but also include all the community sets $\{\mathbb{C}_{t-1}, \mathbb{C}_{t-2}, \mathbb{C}_{t-g}\}$ detected within the last g steps into consideration. In some dynamic networks, node behavior is highly dynamic. Allowing a reasonable number of gap intervals helps us to detect more evolutionary community paths. The parameter selection of the similarity threshold θ and *gap interval* will be discussed in the experiment section (Section 4).

After connecting those detected communities at different timestamps together, we get a new graph showing the evolution of communities along the whole timeline. In this graph, each node represents a detected community. As the last step, we find all the connected components in this graph and define them as the evolutionary community paths in this network. Figure 2 shows some examples of the evolutionary community paths detected in Enron email dataset from Jun. 1, 2001 to Aug. 1, 2001. The number inside each node represents the size of the detected communities, i.e. the number of people in those communities. Each connected component in this graph represents a detected evolutionary community path, which is shown in different colors.

Community Path Outliers. Given an evolutionary community path in the dynamic network, we want to check if it includes anomalous activity at a certain stamp. In this step, we leverage the definition of the *Feature Matrix* in the node-centric model and extend it to the community level. Suppose there are w nodes included in the community path y at time stamp t, we define the ith feature value of path y at time t as $g_{iy,t} = \frac{1}{w}\sum_{j=1}^{w} g_{ij,t}$, where $g_{ij,t}$ denotes the ith feature value of node j at time t. In summary, for each community path y at time t, we have a vector $[g_{1y,t}, g_{2y,t}, ..., g_{iy,t}, ..., g_{my,t}]^T$ indicating its m feature values. Then, we use the VAR model to learn how the activities in community path y changes over time.

(3.3)
$$F_t^{(y)} = c + B_1 F_{t-1}^{(y)} + B_2 F_{t-2}^{(y)} + ... B_j F_{t-j}^{(y)} + ... + B_p F_{t-p}^{(y)} + e_t$$

where B_j is a $m * m$ matrix describing the community path transition between time t and $t-j$. As the last step, we define the outlier score of each community path at time stamp t as the difference between the predicted feature vector and the observed feature vector. Similar with the method in the history-based node model, we use Equation 3.3 to predict the feature vector of community path y at $t+1$ as $\hat{F}_{t+1}^{(y)}$ and calculate the *Community Path Ourlier Score* as $O_{cp,(t+1)}^{y} = ||\hat{F}_{t+1}^{(y)} - F_{t+1}^{(y)}||$.

3.2.2 Community-based Node Model Different from the evolutionary community path model that leverages the behavior history of communities in the dynamic network, the community-based node model is a static graph model which compares the node with its involved community at a fixed time stamp.

Following the definitions in the evolutionary community path model, at each time stamp t, we do a graph partition on snapshot G_t and get a set of communities $\mathbb{C}_t = \{C_{t,1}, C_{t,2}, ..., C_{t,x}\}$. For each detected community r at time t, we summarize its feature vector as $F_t^{(r)} = [g_{1r,t}, g_{2r,t}, ..., g_{ir,t}, ..., g_{mr,t}]^T$, where $g_{ir,t} = \frac{1}{w}\sum_{j=1}^{w} g_{ij,t}$ indicating the average feature of its w nodes. Suppose node z is one of the nodes in community r at time t and its feature vector is $F_t^{(z)} = [g_{1z,t}, g_{2z,t}, ..., g_{mz,t}]^T$. We define the outlier score of each node in the community-based node model as the difference between its feature vector and the average feature vector of its community. At last, we get the *Community-based Outlier* as $O_{cb,t}^{z} = ||F_t^{(r)} - F_t^{(z)}||$.

3.3 Integrated Framework Previously, we defined three kinds of outlier scores in different graph models: *History-based Outlier Score* (O_{hb}), *Community Path Outlier Score* (O_{cp}) and *Community-based Outlier Score* (O_{cb}). Now, we combine these three definitions together into an integrated framework.

In a temporal sequence of graphs $G = \{G_t, t = 1, ..., T\}$, suppose node x is a member of community C at time t and community C is included in the evolutionary path y.[1] We judge node x in the following four conditions:

1) If $O_{hb,t}^{(x)} <$ top $K\%$, node x is considered **normal** at time t. In other words, if the behavior of node x follows its history pattern, we will consider it as a normal node at time t.

[1] If node x is not a member of any community at time t, we will consider it as a normal node at time t.

2) If $O^{(x)}_{hb,t} \geq$ top $K\%$ and $O^{(y)}_{cp,t} \geq$ top $K\%$, node x is considered **anomalous** at time t. In other words, if the behavior of node x does not follow its history pattern and the community path it belongs to shows anomalous activities, we consider it anomalous at time t.

3) If $O^{(x)}_{hb,t} \geq$ top $K\%$ and $O^{(y)}_{cp,t} <$ top $K\%$ and $O^{(x)}_{cb,t} <$ top $K\%$, node x is considered **normal** at time t. In this situation, although node x does not follow its own history pattern, its behavior is actually following the pattern of its community at time t. Therefore, we consider node x as a normal node at time t.

4) If $O^{(x)}_{hb,t} \geq$ top $K\%$ and $O^{(y)}_{cp,t} <$ top $K\%$ and $O^{(x)}_{cb,t} \geq$ top $K\%$, node x is considered **anomalous** at time t. In this situation, the behavior of node x follows neither the pattern of its own history nor the pattern of the community it belongs to. Therefore, we consider node x anomalous at time t.

In our proposed framework, we can reduce the false alarm rate by the condition 3). When we find that the target node does not follow its own history behavior pattern ($O^{(x)}_{hb,t} \geq$ top $K\%$), we continue to examine its current community information and the related community path. If the current state of its involved community path is normal ($O^{(y)}_{cp,t} <$ top $K\%$) and the behavior of node x follows its community pattern ($O^{(x)}_{cb,t} <$ top $K\%$), we will consider node x as a normal node at time t. A real world example of this situation can be found in the email connections of an employee who is just starting a cross-team collaboration. Although the email network of this person may show a large change, he is following the normal community behavior pattern, which makes his behavior less anomalous.

4 Experimental Analysis

Because real world data sets can only provide anecdotal evidence of anomalies, we first create a synthetic data set to quantitatively analyze the effectiveness of our proposed anomaly localization framework. Then we run our framework on three real world data sets. Some proven real world examples are also used to evaluate the performance of our framework in reducing false alarm rates.

4.1 Synthetic Data To create a dynamic graph example, we first generate a random graph with 1000 nodes and 4000 edges. We also construct an adjacency matrix S for this graph where $S[i,j]$ represents the weight of the edge between node i and node j. Additionally, we build a random matrix $R \in \mathbb{R}^{1000*1000}$, where each entry $R(i,j)$ in matrix R is given by:

(4.4)
$$R(i,j) = \begin{cases} 0, & \text{with probability } p = 0.999 \\ 2 * u(i,j), & \text{with probability } p = 0.001 \end{cases}$$

where $u(i,j)$ is a random number generated uniformly between -1 and 1. Then we create a graph sequence $G = \{G_t, t = 1, 2, ..., 25\}$ where $G_i = G_{i-1} + (R_i + R'_i)/2$, $G_0 = S$ and R_i represents the random matrix generated at time i separately.

Pattern Injection. Suppose $\{C_{20,1}, C_{20,2}, ..., C_{20,x}\}$ are the detected x communities in graph G_{20}. We inject three types of patterns into the graph snapshot G_{20}:

- *Normal Pattern*: Select a tuple of random nodes $\{(v_i, v_j) \,|\, v_i \in C_{20,a}, v_j \in C_{20,b}, a \neq b\}$ and swap their positions in graph G_{20}.
  In this situation, although the changed node does not follow its own history behavior pattern at time $t = 20$, its current community path is normal and its behavior matches the community pattern at time $t = 20$. Therefore, this injection pattern is normal, which is related to condition 3) in section 3.3.

- *Anomalous Pattern I*: Select a tuple of random nodes $\{(v_i, v_j) \,|\, v_i \in C_{20,a}, v_j \in C_{20,b}, a \neq b\}$ and swap their positions in graph G_{20}. Then, create a star-like connection pattern within G_{20}. Taking node v_i as an example, increase its edge weights to all other 15 nodes in G_{20} by 1.
  In this situation, the behavior pattern of the changed node does not follow its community pattern at time $t = 20$. In addition, because the star-like pattern is applied to the whole graph G_{20}, its influence on the community path is very small. Therefore, this anomaly is related to condition 4) in section 3.3.

- *Anomalous Pattern II*: Select a tuple of random nodes $\{(v_i, v_j) \,|\, v_i \in C_{20,a}, v_j \in C_{20,b}, a \neq b\}$ and swap their positions in graph G_{20}. Then, create a star-like connection pattern within a node's own community. Taking node v_i as an example, increase its edge weights to all other 15 nodes in $C_{20,a}$ by 1.
  In this situation, because the star-like pattern is applied within the community of the changed node, the community path it belongs to is heavily influenced. Therefore, this anomaly is related to condition 2) in section 3.3.

We inject three types of patterns at time $t = 20$ and repeat each pattern injection five times. Lastly, we get

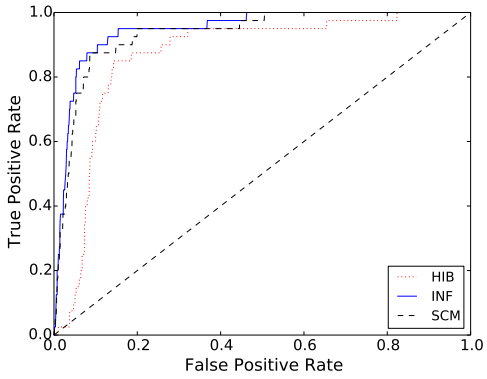
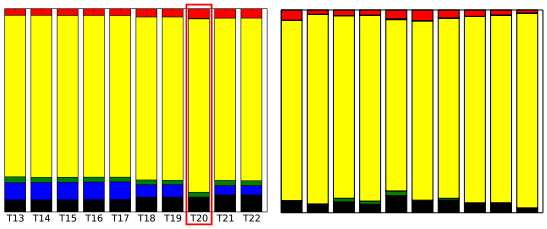


Figure 3: ROC curves comparing different anomaly localization models

20 anomalous nodes, 10 changed normal nodes, and 970 unchanged nodes in graph G_{20}.

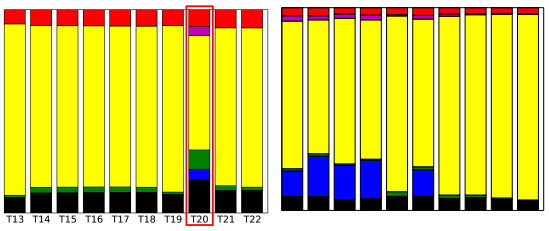
Parameter Selection. In our proposed anomaly localization framework, we need to give two parameters to build the evolutionary community path: similarity threshold $\theta \in [0, 1]$ and allowed gap interval g. In our experiments on a synthetic data set, we set the threshold value to be 0.45, which can provide a reasonable compromise between community matching accuracy and identifying the optimal number of community paths. Also, because the graph sequence we create is relatively stable at each time stamp, we do not need to add gap intervals to build more community paths. Therefore, we set gap interval $g = 0$. In addition, we set the maximum value of the lag order in our time series model as 3 and use the VAR model to determine its optimal value automatically.

Result Evaluation. We run our anomaly localization framework on synthetic data and evaluate its effectiveness by contrasting the detected anomalous nodes against the ground truth set of anomalous nodes we injected in G_{20}. In addition, we compare the performance of our integrated framework (INF) against the history-based node model (HIB) and a static-community algorithm (SCM). HIB only uses the history-based node model to localize anomaly without taking any community information into account. And SCM leverages both the history-based node model and the community-based node model, but the community information it uses is from a static snapshot without dynamic properties of the evolutionary community path. The ROC curves for these three algorithms are shown in Figure 3. The area under the ROC curves for INF, HIB and SCM are respectively given by 0.95, 0.87 and 0.93. We find that our integrated framework performs the best among these three algorithms. Because HIB neglects the community information of a graph node, it may mistakenly consider our injected *Normal Pattern* as an anomaly, which will increase the false positive rate of its localization result. For SCM, because it does not consider dynamic properties in the evolutionary community path, it may fail to detect our injected *Anomalous Pattern II*, which will increase the false negative rate of its localization result. We also show some detailed graph feature analysis of our injected patterns in Figure 4 and Figure 5. The y-axis of these figures represents the graph feature distribution at each point. Each distinct color represents one graph feature (Black: degree, Blue: clustering coefficient, Green: betweenness, Yellow: closeness, Magenta: eigenvector, Red: PageRank) and the length represents its feature value, which is normalized by the sum of the total features. In Figure 4.a and Figure 5.a, we find graph features changing at time $t = 20$ in both node A and node B. However, in Figure 4.b and Figure 5.b, node A matches the features of its community members at time $t = 20$, while node B shows obvious different graph features from its community members.



(a) Node A from $T13$ to $T22$ (b) Community members of node A at $T20$

Figure 4: An Example Node with Normal Pattern



(a) Node B from $T13$ to $T22$ (b) Community members of node A at $T20$

Figure 5: An Example Node with Anomalous Pattern I

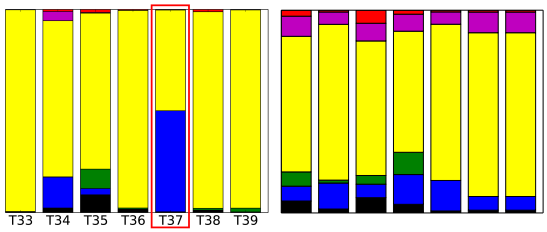
4.2 Enron Email Network Data The Enron email network data is a graph structured data set based on emails exchanged among employees of the Enron Corporation from 1998 to 2002. The graph data set includes all the email records of the year 2001 with a total of 54,214 nodes and approximately 1 million edges. We ag-

gregate the data on a weekly basis, leading to 53 weekly graph snapshots, and we use our proposed framework to localize anomalous nodes. For parameter selection, we use the same values from our experiments on synthetic data: similarity threshold $\theta = 0.45$, gap interval $g = 0$. Furthermore, because we cannot analyze node behavior without enough user communication records, we only focus on the top 1000 nodes that have the most email communication.

The Enron scandal was revealed on October 16, 2001 when Enron reported a 638-million-dollar third quarter loss. Before the scandal broke, there were numerous internal email records and public announcements that showed unusual behavior within Enron Corporation [16]. In this experiment, we use our proposed framework to localize Enron employees with unusual email communication patterns.

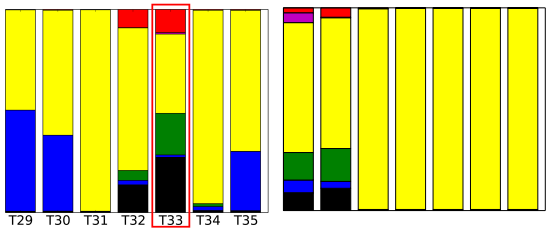
First, we localize Rosalie Fleming, the assistant to Kenneth Lay, as an anomaly during the time Kenneth Lay returned as chief executive between Aug. 2001 and Sept. 2001. Next, we detect an anomalous communication pattern for Janette Elbertson, the assistant of Mark Haedicke (the Managing Director of Enron Corp.), in the middle of August 2001. With further investigation, we find that during this time, someone sent an anonymous warning stating that *questionable accounting practices would lead the company to implode in a wave of accounting scandals*. This warning attracted much attention from Enron executives. Figure 6 and Figure 7 show the graph feature analysis of Rosalie Fleming and Janette Elbertson respectively. In Figure 6, we can easily see anomalous graph feature changes for Rosalie in week 37 (Sept. 17). In Figure 7, anomalies are localized for week 33 (Aug. 20) in the behavior history of Janette.

Additionally, our framework successfully reduces false alarms using the community-centric models. For example, Danny McCarty, the Chief Commercial Officer of Enron's Pipeline Group, shows an unusual inactive email communication pattern at the end of June. 2001. However, by analyzing his dynamic community information, we find that his previous collaborators, Stanley Horton and Drew Fossum, show a similar behavior pattern at that time. Stanley Horton is the chairman and chief executive officer of Enron Transportation Services Company and Drew Fossum is the Diretor of Enron Louisiana Transportation Company. Both are in the same division as Danny McCarty, and they collaborate extensively on energy projects [8]. Figure 8 shows the behavior analysis of these three people from May, 2001 to July 2001. We find that they all have similar graph feature changes in week 24 (June 18). Since the yellow bar represents closeness centrality, these feature changes indicate that they all become less active in their email communication at that time. It is very likely due to a normal event in their division, such as an off-site meeting. In summary, our framework not only localizes related people of the Enron scandal with high accuracy, but also successfully reduces false alarm cases with dynamic community information.



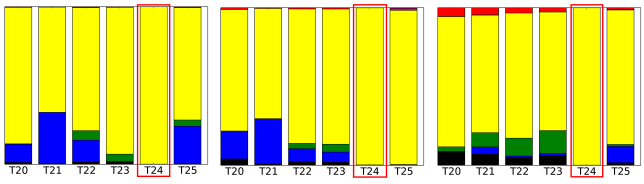
(a) Rosalie Fleming from $T33$ to $T39$ (b) Community members of Rosalie Fleming at $T37$

Figure 6: Graph Feature Analysis of Rosalie Fleming



(a) Janette Elbertson from $T29$ to $T35$ (b) Community members of Janette Elbertson at $T33$

Figure 7: Graph Feature Analysis of Janette Elbertson



(a) Danny McCarty (b) Drew Fossum (c) Stanley Horton

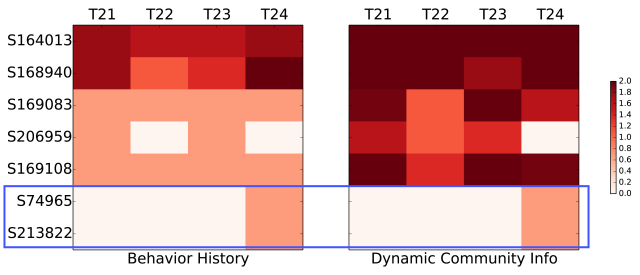
Figure 8: Reduced False Alarms in Enron Email

4.3 Enterprise Network Traffic Data The enterprise network traffic data consists of user-to-server access traffic records from Jan. 1, 2014 to Jul. 13, 2014. This dataset is made available by Pivotal Software Inc. It is a bipartite graph with 148,765 source IPs (users), 135,457 destination IPs (servers), and 34,142,946 user-to-server connections. In the data preprocessing step, we first aggregate the graph data on a daily basis to get

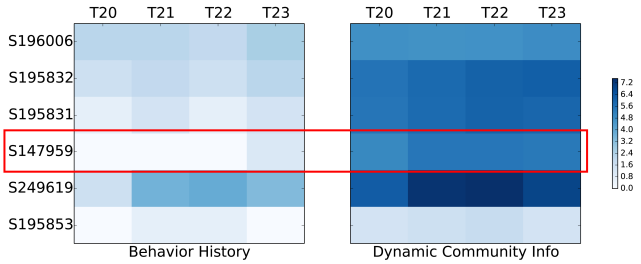
a bipartite graph snapshot $G_{B,t}$. Then we do a standard one-mode projection of $G_{B,t}$, in the form of an undirected graph $G_{P,t} = (V, E_P)$, where source v_i and v_j are connected if and only if they share at least one common destination [17]. Finally, we get a temporal sequence of graphs $G = \{G_t, t = 1, ..., 192\}$, where G_t is a source-based view of the original bipartite graph at time t. Then we run our proposed framework on the temporal graph sequence G to localize anomalous users in the enterprise network. We use the same parameter values from our experiments on synthetic data: similarity threshold $\theta = 0.45$, gap interval $g = 0$.

Figure 9a shows an example of the detected anomalous users V_1 in the enterprise network traffic data. In each of the two figures, the x-axis represents the timeline and the y-axis represents the servers that V_1 has connections with. The block (T_i, S_i) in the figure denotes the number of connections from V_1 to server S_i at time T_i. The darker the block color is, the more connections it includes. Among the two figures, the user history pattern summarizes the previous connections from V_1 to the listed servers, and the community pattern summarizes the previous connections from all the users in the same community as V_1 to the listed servers. We find that, for both patterns, there are unusual new connections to servers $S74965$ and $S213822$ at time $T24$ in V_1's behavior. However, in its community, no member other than V_1 connects to these two servers. It is highly possible that the detected user V_1 is downloading information that he should not access. Furthermore, we show an example of reduced false alarms in Figure 9b. We find that although there is an unusual increase in connections from V_2 to server $S147959$ at time $T23$, it is normal for the community members of V_2 to have many connections to server $S147959$. Therefore, user V_2 is considered normal at time $T23$ in our framework. This is an example of network communication for an employee who has just joined a new team project.

4.4 Facebook Data The data we use is crawled from Facebook public pages by our SINCERE system[2]. SINCERE is a diversified search engine based on user social informatics and stores user interaction data of more than 1,800 Facebook public pages. We use CNN's public Facebook page, one of the largest newsgroups on Facebook, as our data set and collect all its information, including the content of comments, *user-like* information, and their time stamps. In total, we collect data from CNN's public Facebook page from Apr. 1, 2011 to Apr. 1, 2012, including 852,237 comments and 988,235 likes. Then we divide the data into a temporal sequence



(a) Detected Anomaly: User V_1



(b) Reduced False Alarms: User V_2

Figure 9: Example Users in the Enterprise Network

of graph snapshots by day $G = \{G_t, t = 1, ..., N\}$, where G_t aggregates all the *user-like* information [18] within the interval $[t-1, t]$. In the *user-like* graph $G_t = (V_t, E_t)$, nodes V_t stand for the users who liked a comment or whose comments were liked by others during the time slot $[t-1, t]$. Edges E_t stand for the like connections between those users. In summary, we get 366 *user-like* graph snapshots with a total of 173,481 nodes and 988,235 connections. Then we run our proposed framework on the temporal graph sequence G to localize anomalous users in the Facebook data. For the similarity threshold, we use the same value from our experiments on synthetic data, $\theta = 0.45$. Considering the dynamics and periodicity of online user behavior, we set the gap interval as one week, i.e. $g = 7$.

We discover several interesting anomalies in the Facebook data set. For instance, we detect a Facebook user *dee.v.bee.12* (UserID=805330523) who shows anomalous behavior on the CNN Facebook page. This user's information is then retrieved from Facebook. The example messages we show below are those cached in our SINCERE system. *dee.v.bee.12* is one of the top anomalies around Jan. 20, 2012. During that time, he posted many spam messages supporting Ron Paul, a former American politician, on CNN's public page[3]:

RON PAUL! RON PAUL! RON PAUL! RON PAUL! RON PAUL! RON PAUL! RON PAUL!

[2] http://sincere.se/

[3] https://www.facebook.com/cnn/posts/10150553600681509

RON PAUL! RON PAUL! RON PAUL! RON...

In addition, we find this user spreads the same message repeatedly in four different posts around Feb. 02, 2012, when he is also ranked as one of the top anomalies in our framework. The message he sent repeatedly is as follows:

> *there is vote fraud in nevada. we demand a re vote, on paper ballots counted in public and no votes shall be taken out of public sight! YOU ARE STEALING THE PEOPLES VOICE!*

Our framework also successfully reduces false alarms by leveraging the community information of online user behavior. For example, we find the Facebook user Timothy (UserID: 100000023776257) showing unusually increased communication on Jan. 20, 2012 based on his own history behavior pattern. However, in his dynamic community pattern, we find that he is having a discussion with a group of active users about Occupy Wall Street in a CNN post[4] during that time. In summary, our framework successfully localizes malicious online users by digging into their history behavior and social patterns, which offers a new solution to detect adaptive online attackers who attempt to build legitimate social links instead of fake accounts to avoid detection.

5 Conclusions

In this paper, we propose a new framework to localize temporal anomalies in large evolving graphs with reduced false alarm rate. Our framework consists of three graph models with different emphasis on dynamic graph structure. The *History-based Node Model* is a node centric model that describes the behavior history of a single node in dynamic graphs. Both the *Community-based Node Model* and the *Evolutionary Community Path Model* are community-centric models. The former emphasizes the community information of a single node at a fixed time slot, and the latter tracks the dynamics of graph communities along the whole timeline. Our experimental results on synthetic and real world data sets show our proposed framework effectively and consistently localizes temporal anomalies in large evolving graphs with reduced false alarm rate.

Acknowledgments

We would like to thank Pivotal Data Science Team for the valuable discussions. We also thank the Cyber Security Research Alliance of United State Army Research Laboratory for partially supporting this research.

[4]https://www.facebook.com/cnn/posts/285797141473607

References

[1] C. Fang, M. Kohram, X. Meng, and A. Ralescu, "Graph embedding framework for link prediction and vertex behavior modeling in temporal social networks," in *SNA-KDD*, 2011.

[2] C. Phua, V. Lee, K. Smith, and R. Gayler, "A comprehensive survey of data mining-based fraud detection research," *CoRR*, 2010.

[3] L. Akoglu and C. Faloutsos, "Event detection in time series of mobile communication graphs," in *Army Science Conference*, 2010, pp. 77–79.

[4] N. R. Suri, M. N. Murty, and G. Athithan, "Characterizing temporal anomalies in evolving networks," in *Advances in Knowledge Discovery and Data Mining*. Springer, 2014, pp. 422–433.

[5] R. A. Rossi, B. Gallagher, J. Neville, and K. Henderson, "Modeling dynamic behavior in large evolving graphs," in *WSDM*, 2013.

[6] K. Sricharan and K. Das, "Localizing anomalous changes in time-evolving graphs," in *SIGMOD*, 2014.

[7] T. Idé and H. Kashima, "Eigenspace-based anomaly detection in computer systems," in *KDD*, 2004.

[8] Z. Chen, W. Hendrix, and N. F. Samatova, "Community-based anomaly detection in evolutionary networks," *Journal of Intelligent Information Systems*, 2012.

[9] Y. Sun, J. Tang, J. Han, M. Gupta, and B. Zhao, "Community evolution detection in dynamic heterogeneous information networks," in *KDD-MLG*, 2010.

[10] M. Gupta, J. Gao, Y. Sun, and J. Han, "Integrating community matching and outlier detection for mining evolutionary community outliers," in *KDD*, 2012.

[11] K. Henderson, B. Gallagher, L. Li, L. Akoglu, T. Eliassi-Rad, H. Tong, and C. Faloutsos, "It's who you know: graph mining using recursive structural features," in *KDD*, 2011.

[12] S. Johansen, "Likelihood-based inference in cointegrated vector autoregressive models," *OUP Catalogue*, 1995.

[13] A. Clauset, M. E. Newman, and C. Moore, "Finding community structure in very large networks," *Physical review E*, 2004.

[14] P. Jaccard, "The distribution of the flora in the alpine zone. 1," *New phytologist*, pp. 37–50, 1912.

[15] D. Greene, D. Doyle, and P. Cunningham, "Tracking the evolution of communities in dynamic social networks," in *ASONAM*, 2010.

[16] "Enron: The smartest guys in the room." http://www.pbs.org/independentlens/enron/timeline2001.html, Dec. 2011.

[17] Q. Ding, N. Katenka, P. Barford, E. Kolaczyk, and M. Crovella, "Intrusion as (anti) social communication: characterization and detection," in *KDD*, 2012.

[18] T. Wang, K. C. Wang, F. Erlandsson, S. F. Wu, and R. Faris, "The influence of feedback with different opinions on continued user participation in online newsgroups," in *ASONAM*, 2013.