An Ensemble Approach for Event Detection and Characterization in Dynamic Graphs

Shebuti Rayana
Department of Computer Science
Stony Brook University
Stony Brook, NY 11794-4400, USA
srayana@cs.stonybrook.edu

Leman Akoglu
Department of Computer Science
Stony Brook University
Stony Brook, NY 11794-4400, USA
leman@cs.stonybrook.edu

ABSTRACT

Event detection in datasets represented by dynamic graphs is an important task for its applications in a variety of domains, such as cyber security, online and telecommunications, fault and fraud detection, etc. Despite recent advances in this area, there does not exist a single winning algorithm known to work well across different datasets. In fact, designing a single method that is effective on a wide range of datasets is a challenging task. In this work, we propose an *ensemble* approach for event detection and characterization of dynamic graphs. Our ensemble leverages three different base detection techniques, the results of which are systematically combined to get a final outcome. What is more, we characterize the events; by identifying the specific entities, i.e. nodes and edges, that are most responsible for the detected changes. Our ensemble employs a robust rank aggregation strategy to order both the time points as well as the entities by the magnitude of their anomalousness, which as a result yields a superior ranking compared to the base techniques, thanks to its voting mechanism. Experiments performed on both simulated (network traffic flow data with ground truth) and real data (New York Times news corpus) show that our proposed ensemble successfully identifies the important change points in which a given dynamic graph goes through notable state changes, and reveals the key entities that instantiate these changes.

1. INTRODUCTION

Anomaly detection in datasets represented by dynamic graphs has received much attention in recent years because of its wide range of applications in intrusion detection in cyber networks [28], fraud detection (insurance fraud, credit card fraud, auction fraud etc.) [2, 6], fault detection in medical claims, engineering systems [9], sensor networks and many more domains. In time series data, events which deviate from the normal behavior are anomalous. In dynamic graph analysis, the whole graph is observed in a sequence of discrete time ticks. Each of these time ticks represents a partial connectivity of the whole graph. To detect the anomalies one has to find the change points in the time series (known as event detection) and also the entities (nodes/edges) responsible for those changes (known as characterization).

ODD^2 '14, August 24–27 2014, New York, NY, USA.
Copyright 2014 ACM 978-1-4503-2998-9/14/08 ...$15.00.
http://dx.doi.org/10.1145/2656269.2656274.

Despite the availability of enormous amount of data from different domains, spotting the events of interest is challenging as they are quite rare. For example, the frequency of cyber attacks is very low compared to the whole network flow volume. The number of fraudulent transactions are rare compared to normal transactions in a financial organization. Although these events of interest occur infrequently their importance is very high compared to other events.

A lot of research has been done on event detection and characterization in different research communities. Both supervised and unsupervised learning techniques are used. The limitations of supervised methods are that they require labeled data which is not always available for real-world data, and cannot detect novel events that have never been observed previously. On the other hand, unsupervised methods do not require labeled data, they detect suspicious events (change points) or entities if the behavior deviates from the normal behavior in time [17]. The success of these algorithms depends on the metric upon which the change points are detected and also the characteristics of the data being used. For example, some approaches use a distance metric between consecutive graph pairs in a time series to find events [4, 29]. In event detection and characterization it is also important to find the anomalousness of rare events/entities and rank them accordingly. This ranking is often desirable over the traditional techniques which gives binary decisions (anomaly vs non-anomaly). There exist several outlier ranking approaches for graph data [14, 27, 15], however most are for static settings. Very few ranking approaches are available for dynamic graph based anomaly detection [13, 30]. Despite the fact that there exist many algorithms on event detection in dynamic graphs, there is no single winning algorithm known to work well across different datasets. In fact, designing a single approach robust to a wide range of datasets is a challenging task.

In this paper, we propose a unified framework for event detection and characterization in dynamic graphs by designing an *ensemble* approach that systematically combines results from different event detection algorithms. Our ensemble utilizes three detectors, and can accommodate other algorithms that provide detection and characterization. It is well known that ensemble approaches are effective in improving overall performance over the individual base techniques [1]. In particular, the motivation of using a set of algorithms is twofold.

First, we aim to find consensus on the detected anomalies and second, we aim to identify those anomalies which are detected by one algorithm but not by the others. To the best of our knowledge, this is the first work in ensemble design for dynamic graph based event detection and characterization.

In our first algorithm we propose a new event detection technique which flags the change points in a time-varying graph at which many nodes deviate from their "normal" behavior. We use an "eigen-behavior" based technique which spots collective behavioral changes of nodes and edges to find suspicious events. At those events the nodes and edges which show the highest changes are also marked as anomalies. In detecting collective behavior changes, either many of the nodes/edges need to go through changes or a few nodes/edges need to change a lot. The second technique of the ensemble involves probabilistic time series anomaly detection. Here, we take a statistical approach to fit the time series of structural features of nodes and edges to several parametric distributions and select the best fit. The points are then flagged as anomalous based on their likelihoods. Finally, we employ a third algorithm called *SPIRIT* [23] which can find (hidden) trends in time series data, and dynamically detects change points by tracking the changes in the trends to spot potential anomalies. We remark that all three algorithms have two key properties: (1) event detection, and (2) characterization (finding the culprits).

In summary, our main contributions are as follows:

- **Ensemble Event Detection:** We propose an ensemble approach to detect change points in time series graph data. Our ensemble effectively merges results from different techniques to provide a robust outcome.

- **Consensus Rank Aggregation:** Our method uses both (i) rank-based and (ii) score-based aggregation approaches to build multiple consensus rankings. The results from different consensus approaches are then merged to obtain a final ranking that is better than most of the individual base algorithms.

- **Characterization:** Our approach could also attribute the changes to specific nodes and edges in the graph and hence characterize the detected changes.

Different from most earlier works on anomaly detection, we experiment with both simulated (network traffic flow data with ground truth) as well as real (New York Times news corpus data without ground truth) datasets to spot events and pinpoint anomalous agents. The network flow data has been carefully simulated in a realistic manner at NGAS R&T Space Park (`www.northropgrumman.com`) which mimics operations and anomalies that correspond to real-world events. We construct dynamic graphs based on the communications (edges) among different hosts (nodes) in the network. Our proposed approach is able to successfully unearth these events and the individual agents that initiated the events with high accuracy. Quantitative evaluation based on ground truth shows that our final ensemble yields better ranking of the events than most of the individual base algorithms. We also use published articles (Jan 2000 - July 2007) of New York Times (NYT) [26] where we construct dynamic graphs based on the named entities (nodes) being co-mentioned (edges) in the articles. Our experiments successfully reveal several big events during the time period of the NYT data, such as presidential elections of 2001, 9/11 terrorist attacks in World Trade Center, 2003 Columbia space shuttle disaster, etc., in addition to the key entities associated with these events.

The rest of the paper is organized as follows. Section 2 first discusses some important related works. Section 3 describes the work-flow of our ensemble and then its individual components. Section 4 gives our experimental setup and Section 5 presents the results. Finally, we provide a summary and discuss future directions for research.

2. RELATED WORK

Ensembles for unsupervised outlier detection is an emerging topic that has been neglected for a long time compared to ensembles for classification and clustering problems [7, 31, 11]. This is because in unsupervised settings no ground truth is available to evaluate the merit of the ensemble over its components. Moreover, unlike in clustering which falls under unsupervised learning, there exists no objective or fitness functions for outlier mining. Nevertheless, there have been several recent works on building outlier ensembles. In a recent position paper, Aggarwal [1] provided a categorization of the existing outlier ensemble approaches based on corresponding algorithmic strategies. According to Aggarwal, there have been traces of the very idea of combining results from different models in many earlier works but none of them are explicitly named as ensemble approaches.

In particular, ensemble approach is effectively used in high-dimensional outlier detection [1] where multiple subspaces of data are explored to detect outliers. The feature bagging approach [20] is the earliest work formalizing an outlier ensemble in high dimensional feature space, as outlier behavior of data points are often described by a subset of the dimensions. This approach uses the same base algorithm (LOF [3]) on different feature subsets and provides a rank based merging to create the final consensus. Feature bagging is better calibrated by [10, 18] which convert the outlier scores to probability estimates and use a score based merging. Our approach in contrast uses both rank based and score based aggregation to build the final ensemble.

In addition to using the variants of the same algorithm as ensemble components, it is possible to use different algorithms to build a heterogeneous ensemble. In [21] Nguyen et al. use both LOF [3] and LOCI [22] (density based outlier detectors) as components of their ensemble. While their approach is similar to ours, we build a heterogeneous ensemble for dynamic graphs in contrast to static clouds of points and capture the time evolving behavior of the data. Other existing outlier ensembles are also designed for clouds of multi-dimensional data points. To the best of our knowledge, ours is the first ensemble approach for event detection in time-evolving graph data.

3. PROPOSED METHOD

3.1 Ensemble Approach

We design an ensemble framework for event detection and characterization for dynamic graphs, as they appear in numerous scenarios including computer networks, trading networks, transaction networks, phone call and email communications. We propose to use three different event detection techniques to find change points in time series of graphs. Each of these algorithms operate in two phases (i) event de-

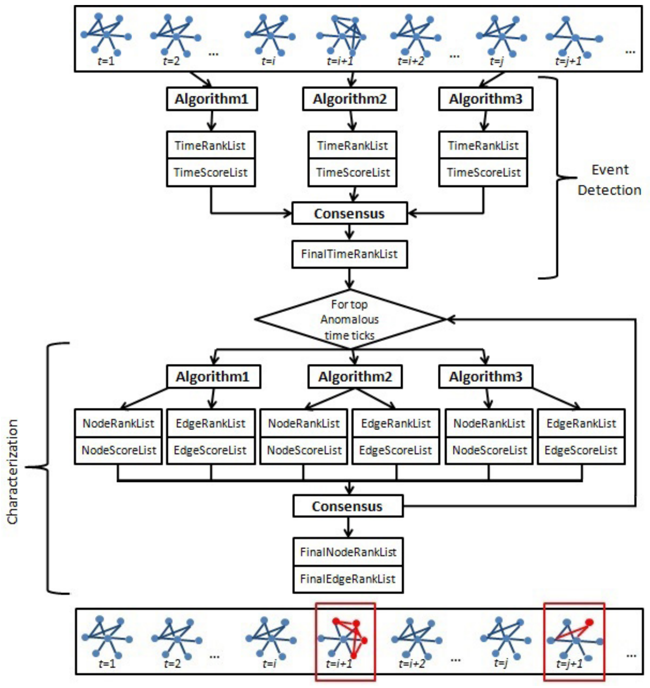

Figure 1: Proposed ensemble approach work-flow.

tection phase and (ii) characterization phase. In this section we give an overview of the work-flow of our ensemble approach, and defer the details of these individual techniques to Section 3.2. A flow chart of the framework is depicted in Figure 1, which is described next.

3.1.1 Event Detection

We use three event detection algorithms (abstracted as $Algorithm1$, $Algorithm2$, $Algorithm3$) to find anomalous time ticks (events) that show significant change points in the dynamic graph constructed from input time series data. Each algorithm has a specific measure to score the individual time ticks, depicting the amount of behavioral change of the graph. Different algorithms employ different measures. In general, the higher its score, the more anomalous a time tick is. Each algorithm provides a ranklist and a scorelist of time ticks in event detection phase. As such, we obtain three ranklists ranked from most to least anomalous and three scorelists for the time ticks as shown in Figure 1.

3.1.2 Characterization

For each time tick found to be anomalous for an algorithm we also identify the nodes and edges which are responsible for it. Again, each algorithm has a means to score individual nodes and edges which represents their amount of change in behavior. As such, for each anomalous time tick we create sorted ranked lists of nodes and similarly of edges as shown in Figure 1. Therefore, the characterization phase of each algorithm provides a ranklist and a scorelist of nodes/edges for each anomalous time tick. Note that some time ticks may be detected as anomalous by only a subset of the algorithms, in which case we create only as many lists.

3.1.3 Consensus Rank Aggregation

Our goal is to combine the results, i.e., ranked lists, of the ensemble algorithms to build a final rank ordering of (i) time ticks, and (ii) nodes/edges at top anomalous time ticks. In event detection phase, each of the base algorithm gives an anomalousness score to each time tick and then sort the list of those time ticks based on these scores to obtain the ranklist. Again in characterization phase, the anomalousness scores of nodes/edges are sorted to obtain the corresponding ranklists for top anomalous time ticks. To build the ensemble, we merge the results from different base algorithms to come up with the final result. We use both (i) rank based merging and (ii) score based merging to build the consensus.

Rank Based Merging: Different algorithms provide different scoring techniques of time ticks/nodes/edges according to their anomalousness which are not comparable. We use only the rankings of the time ticks to merge the ranklists obtained from the base algorithms in rank based merging. Here we use two techniques:

- **Inverse Rank Merging:** Each time tick/node/edge has a rank associated with it in a ranklist. We use inverse of these ranks $(1/R)$ (here, R is the rank of a time tick/node/edge in a ranklist) to calculate a score associated with individual time tick/node/edge. So, the top element in the ranklist has rank 1 and score 1, next element has rank 2 and score 0.5 and so on. We calculate the final score of individual time tick/node/edge by taking the average of these inverse rank scores from all the base algorithms. Finally, according to this final averaged rank scores we sort the time ticks/nodes/edges to obtain the final merged ranklist.

- **Kemeny Young:** *Kemeny Young* [16] method is a voting system which uses preferential ballot and pairwise comparison count to identify the most popular choice. We consider our algorithms as voters and the time ticks/nodes/edges as the candidates they vote for. The time tick/node/edge which comes at the top in a ranklist from a base algorithm is the most preferred candidate for that algorithm. The next one has less preference than the first one and so on. We calculate the scores in two steps, (i) creating a matrix that counts pair-wise voter preferences and (ii) calculating a score for each ranking position which is the sum of the pair-wise counts that apply to that ranking. We sort the time ticks/nodes/edges according to these scores to obtain the final merged ranklist.

Score based merging: As scores from different algorithms are not comparable we convert the scores to a well-calibrated probability estimate to make them comparable. In score based merging we use two techniques to convert output scores from the algorithms to probability estimates:

- **Unification:** The unification method [18] of anomalousness scores has three steps (i) Regularization, (ii) Normalization, and (iii) Gaussian Scaling:

  i) **Regularization:** We regularize the anomalousness scores from different base algorithms by transforming the interval of the scores from [base,∞) to [0,∞) in a way which does not change the rank order. If the range of score already is in the interval [0,∞) then we skip this step.

ii) **Normalization:** We use a linear transformation to transform interval of the scores to [0,1].

iii) **Scaling:** We use *Gaussian scaling* to convert the normalized scores to probability estimates which represent the probability of anomalousness of the time ticks/nodes/edges.

- **Mixture Modeling:** In the mixture modeling approach [10] we model the score distributions of the time ticks/nodes/edges as a mixture of *Exponential* and *Gaussian* probability distribution. The typical anomaly score distribution of the anomalous and normal classes show that the normal class follows an Exponential distribution and anomalous class follows a Gaussian distribution [10]. This suggests that a mixture model consisting of an Exponential and a Gaussian component may fit well to the anomaly score distributions. We use an *expectation maximization* (EM) algorithm to minimize the negative log likelihood function of the mixture model to estimate the parameters. We calculate the final posterior probability with *Bayes' rule* which represents the probability of anomalousness of the time ticks/nodes/edges.

After converting the scores from the base algorithms to probability estimates we use two techniques to merge them, (i) taking average of the probability scores and (i) taking maximum of the probability scores. Then we sort these scores to obtain the final merged ranklist.

These orderings from different consensus rank merging techniques essentially capture the agreement among the individual base algorithms and also includes the points where they disagree. Finally, we merge the ranklists from different consensus rank merging techniques using inverse rank merging approach to get the final ranklist of time ticks/nodes/edges. Here, we use inverse rank merging for its stable performance among all consensus approaches. The final ensemble scores ($1/R$) associated with the final ranklist are fed to mixture modeling [10] to find a cut-off for anomaly detection. Section 5 contains our experimental results.

3.1.4 Feature Extraction

For event detection and specifically for characterization, we capture the behavior of the dynamic graph through the behavior of its entities, i.e. nodes and edges. As such, given the graph sequence $G_1, \ldots, G_t, \ldots$, we extract several graph-centric features for every node and edge for each G_i. This way, each node/edge is represented by a time series of its feature values. We experimented with several features. In particular, for nodes we extracted (1) *weighted degree*, (2) *degree*, and (3) number of *local triangles* associated with each node. For edges we extracted (1) *weight*, (2) number of *common neighbors*, and (3) number of *total neighbors* of the two end points of each edge.

Having outlined the general work-flow of our approach, we next describe the individual algorithms of the ensemble.

3.2 Ensemble Components

3.2.1 Eigen Behavior based Event Detection(EBED)

In this algorithm we use the time series features of nodes and edges of our input graph to detect the change points through eigen-behavior analysis.

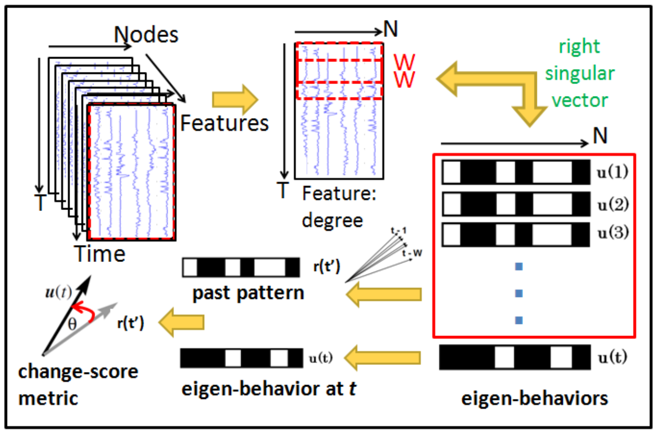

Figure 2: EBED change-point detection work-flow.

Figure 2 illustrates our proposed method, where T denotes the number of time ticks, N denotes the number of graph entities (nodes and edges), and F denotes the number of graph-centric features extracted. We take a $T \times N$ matrix for a selected feature F_i for nodes and edges (e.g., degree), define a window size W over the time series values of all entities, and compute the largest right singular vector of W. We treat this vector as the "eigen-behavior" $u(t)$ of the system during time window t, as the right singular vector of W is the same as the principal eigenvector of $W^T W$ which is essentially the time series similarity matrix of the graph entities. This vector is positive due to a famous theorem by Perron-Frobenius [24, 8], and lends itself to interpreting its entries as the "activity" or "behavior" of each graph entity. To capture the behavior of the graph over time, we slide the window down one time tick and recompute its eigen-behavior. We keep doing this until we reach the end of the time series.

Given new data i.e., a new graph at time t, we calculate the eigen-behavior of the new W. Using the eigen-behavior vectors $u(t')$ computed at previous time windows $t' < t$, we compute a typical eigen-behavior vector $r(t')$ by taking their arithmetic average. We compare the eigen-behavior $u(t)$ with the typical eigen-behavior $r(t')$ to quantify their similarity. As the anomalousness measure, we use what we call the Z *score* which is $Z = 1 - u^T r$. If $u(t)$ is the same as $r(t')$ then their dot product is 1, i.e., $Z = 0$ and if $u(t)$ is perpendicular to $r(t')$ then $Z = 1$. Thus, $Z \in [0, 1]$ and a significantly high value of Z indicates a change point.

For each detected anomalous time tick $\bar{t}$, we also pinpoint the specific nodes and edges which are responsible for the change. To do so, we compute the percentage of relative change $(u_i(\bar{t}) - r_i(\bar{t}'))$ for all nodes/edges. The higher the relative change, the more anomalous the node/edge.

3.2.2 Probabilistic Time Series Anomaly Detection (PTSAD)

To observe the lower-granularity behavior of nodes and edges we design a second algorithm which uses a probabilistic approach to detect anomalies in time series data. Here, we use several parametric distributions to fit individual time series of nodes and edges. We start with a well-known parametric distribution, *Poisson*, which is often used for fitting count data. While simple Poisson is not sufficient for sparse series with many zeros, as often observed in the real-world. In fact most real-world count data is frequently characterized by overdispersion and excess number of zeros. Hence, our second choice is a *Zero-Inflated Poisson* (ZIP) [19] count model, as it provides a way of modeling excess zeros.

We further look for simpler models which fit the data with many zeros and employ the *hurdle models*. Rather than using a single complex distribution, hurdle models [25, 12] assume that the data is generated by two simple, separate processes; (i) the hurdle and (ii) the count components. The former process determines whether there exists activity or not at a certain time tick and in case of activity the count process determines the actual positive counts. One simple way to model the hurdle process is to assume an independent *Bernoulli*. In reality, there may be dependencies, where each activity influences the probability of subsequent activities. Thus, we also consider modeling the hurdle process with first order *Markov* models. For the count component, we use the *Zero-Truncated Poisson* (ZTP) [5] distribution.

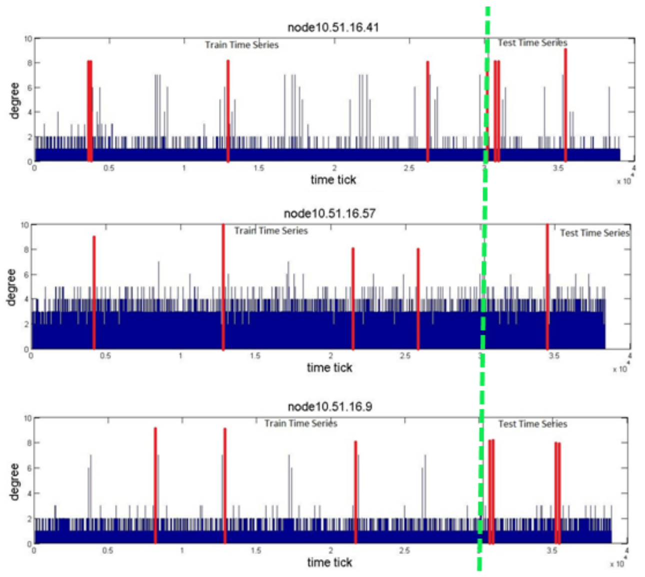

Figure 3: PTSAD approach for event detection in time series of three example nodes (rows) in network traffic flow data.

All in all, we fit four different (parametric) distributions to a training portion of each node/edge time series: Poisson, ZIP, Bernoulli+ZTP and Markov+ZTP. In order to select the best model for each time series, we employ a model selection procedure where we use several criteria including Akaike Information Criterion (AIC), log-likelihood of training data, Vuong's likelihood ratio test [32], and finally the log gain on test data. We further check the goodness of fit for those models with bootstrapping. Note that the best-fit model for each entity may be different. We also remark that our framework is flexible to incorporate additional distributions in the future, including non-parametric ones.

After fitting all the time series training data of individual nodes and edges we perform a single-sided test to compute a *p-value* (i.e., $P(X \geq x) = 1 - cdf_H(x) + pdf_H(x)$, where H is the model that fits best for a given entity). The lower the p-value, the more anomalous the time tick is for a given entity (to flag anomalies, we use the 0.05 threshold to capture significance at the 95 percentile). As it would be unrealistic to assume that the training data is anomaly-free, we first use the p-values to spot anomalous time ticks in the training data. We then remove those anomalous points and refit our models to build the representative ones. At this point, our system is ready to assess new coming data.

In Figure 3, we show example anomalous time ticks (red bars) as detected in the time series of three randomly selected nodes from our simulated network traffic flow data. The dashed bar separates test series from the training series.

PTSAD does not yield an aggregated result/score for the individual time ticks. It rather finds the anomalous time ticks for individual nodes/edges. Thus we aggregate the p-values of all nodes/edges along all the time ticks. We do so by taking the normalized sum of the p-values and invert them ($\in [0, 1]$) (so that higher is more anomalous) and then sort the time ticks accordingly. After detecting the most anomalous aggregated time ticks, we use the normalized logarithm of p-values ($\in [0, 1]$) of nodes/edges corresponding to those time ticks for characterization (again, the higher, the more anomalous).

3.2.3 SPIRIT

We use a third algorithm called *SPIRIT* for Streaming Pattern DIscoveRy in multIple Time-Series by Papadimitriou et al. [23]. This approach can incrementally capture correlations, discover trends, and dynamically detect change points in multiple regular time series data. Here, the intuition is to discover the underlying trend of a large number of numerical streams with a few hidden variables, where the hidden variables are the *projections* of the observed streams onto the principal direction vectors (eigenvectors). These discovered trends are exploited for detecting anomalies.

In a nutshell, the algorithm starts with a specific number of hidden variables which capture the general trends of the data. Whenever the main trends change, new hidden variables are introduced or several of existing ones are discarded to capture the change. This algorithm can further quantify the change in the behavior of graph entities for characterization through their *participation weights*, which are the entries of the principal direction vectors for the entities. For further details on the specific algorithm, we refer the reader to the original paper [23].

Finally, we remark that our ensemble is flexible to incorporate other approaches for event detection in multiple time series data.

4. EXPERIMENT SETUP

4.1 Dataset Description

4.1.1 Dataset 1: Challenge Network Traffic Flow

The simulated dataset that we use is a Cyber Challenge Network traffic flow data over 217 hours (≈9 days) and captures the interactions between hosts in the network. The dataset contains the to-from information of the interactions along with the time stamps. The time-aggregated graph representing the whole network has 125 nodes and 352 undirected edges connecting them. We choose the sample rate of 10 minutes for this dataset (with 1304 time ticks) as lower sample rates result in excess number of change points (potentially many false positives) due to large fluctuations in the graph structure over small time periods, on the other hand, higher sample rates obscure the true positives (i.e., actual changes).

4.1.2 Dataset 2: NYT News Corpus

The real dataset that we use is the NYT hand-annotated (by human editors) news corpus [26] of seven and a half years of published articles(Jan 2000 - July 2007). We construct our dynamic graphs based on the named entities co-mentioned in an article. Therefore, the named entities are the nodes of the graph and if two entities are co-mentioned in an article there is an edge between them. Here the named entities are people, places, organizations, etc. This data has around 320000 entities to construct the time-aggregated graph. We analyze the data with weekly granularity.

4.2 Feature Selection

We extracted different features for nodes and edges with selected sample rates for both datasets. In particular, for nodes we extracted (1) degree, (2) weighted degree, and (3) number of local triangles. For undirected edges we extracted (1) weight, (2) number of common neighbors, and (3) number of total degree of the two end points. To avoid the skewing effect of large edge weights we selected the *degree* feature for the nodes to analyze the network for our final ensemble. We also selected the *total degree* feature for analyzing the time series of edges. For NYT data we used *weighted degree* feature for the nodes and *weights* feature for the edges to build the final ensemble. However the choice of these features can be application dependent.

4.3 Window Size Selection in EBED

For *EBED* we use a window size W of 4 for both datasets to calculate the principal eigen-vector. In social network or any human activity related network, a window size corresponding to 7 days generally represents the pattern of human behavior. As our experiment with network flow data was not portraying human activity, we experimented with different window sizes (4,5,6,7 etc.) to find the best size suitable for the network. Similarly, we experimented with different window sizes to come up with a suitable size for NYT data. In general, the larger the window gets, the more aggregated the results become.

4.4 Model Selection in PTSAD

For probabilistic time series analysis, we fit four different (parametric) distributions (Poisson, ZIP, Bernoulli+ZTP and Markov+ZTP) to a training portion of each node/edge time series. We used first 7 days of data from network flow dataset and first 100 weeks of data from NYT dataset for training and rest of the data from both datasets for testing. To pick the best fit out of the four for each time series, we employed several model selection procedures; including the straightforward log-likelihood of training data, the Akaike Information Criterion (AIC), Vuong's likelihood ratio test [32], and finally the log gain on test data. Each model selection criterion votes for a distribution (i.e., model) for the time series of each node/edge. The distribution which gets the highest vote is selected as a best fit for that particular node/edge. We briefly describe the four criteria as follows.

- *Log-likelihood* is higher for the better model. Model complexity (i.e. number of parameters) is not incorporated.
- *AIC* penalizes high model complexity. The lower the AIC, the better the model.
- *Vuong's test* finds the log-likelihood ratio of a given pair of models (say model 1 and model 2) and calculates a p-value for the sign of the ratio. If p-value is larger than 0.05, then the ratio does not give any decision (i.e., one model cannot be claimed better than the other). Otherwise, model 1 is claimed to be better than model 2 if the ratio is positive, and vice versa if the ratio is negative.
- *Log gain* is similar to Vuong's test, but instead assesses the log-likelihood ratio on the test data.

Table 1 shows the percentage pairwise agreements between the model selection criteria on the best model chosen for the nodes' time series of our network flow data. We also give the cardinality, which denotes the number of series for which both models can provide a fit (some fittings may fail due to model or data degeneracy, hence cardinality is less than the number of nodes). We notice that there exist high agreements among the various tests, and we use the majority vote to pick the final model for each node/edge.

Table 1: Agreement Among Model Selection Criteria in Challenge Network

cardinality + % of agreement	**Log likelihood**	**Vuong's Test**	**Log Gain**
AIC	card.: 121 78.51%	card.: 83 97.59%	card.: 96 82.29%
Log likelihood		card.: 83 100%	card.: 96 80.21%
Vuong's Test			card.:81 90.12%

4.5 Goodness of Fit Test

We further check the goodness of fit for the best model chosen using bootstrapping. For each node/edge, we take the best fit model and its fitted parameters P. We generate N (= 1000) simulated time series by bootstrapping from the best fit model. Next we estimate the parameters P^* from each of the N time series and form an empirical distribution of those fitted parameters. We obtain a 95% bootstrap confidence interval as the interval from the 2.5%-ile to 97.5%-ile of this bootstrap distribution. If P falls within the confidence interval, we conclude that the best fit is indeed a good fit for the nodes/edges time series.

5. EXPERIMENT RESULTS

In this section we present our experimental results of event detection and characterization in the simulated Cyber Challenge Network dataset and NYT news corpus dataset using our ensemble approach.

5.1 Dataset 1: Results

5.1.1 Event Detection

Recall from Section 3.2 the three methods used in our ensemble have different scoring techniques to rank the time ticks for change point detection:

- Z score is used for *EBED*.
- Inverse of the normalized sum of p-values of all nodes/edges is used for *PTSAD*.
- Projection is used for *SPIRIT*.

As the scoring measures are completely different, it is not possible to directly combine these scores to build a ranking of the time ticks by anomalousness. Therefore, we make individual sorted ranked lists for each algorithm, and use both rank based merging and score based merging techniques described in Section 3.1.3 to come up with the consensus ranklists. Finally, we use inverse rank based merging to obtain the final ranklist.

Figure 4 shows the top ranking time points (with red bars) from all three techniques used in our ensemble to find anomalous events (from top to bottom: *EBED*, *PTSAD*, and *SPIRIT*). We notice that time tick 376 is detected as a change point by all the three algorithms and it is the highest ranked anomalous event. *EBED* and *PTSAD* also agree on time tick 1126, at which point *SPIRIT* also has a somewhat small spike. There also exist some events that only individual methods detect. Our ensemble also includes those in the final ordering.

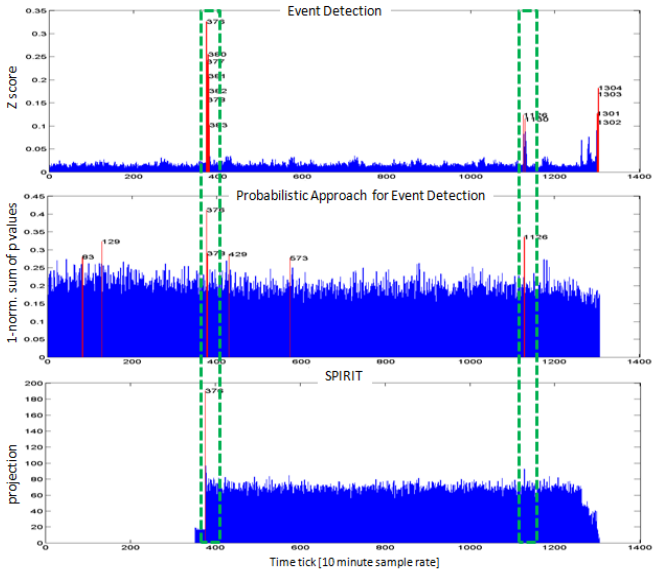

Figure 4: Change point detection in time series of challenge network for all three base algorithms in our ensemble approach. Red bars depict the top anomalous time ticks.

5.1.2 Characterization

After detecting anomalous events, we identify the nodes and edges which are responsible for those events. Again, different algorithms have different scoring techniques to attribute the change to specific nodes/edges. Following are the scores used by the algorithms in our ensemble.

- Relative change (% change of current eigen-behavior from previous one) is used for *EBED*.
- Normalized logarithm of p-value is used for *PTSAD*.
- Participation weight is used for *SPIRIT*.

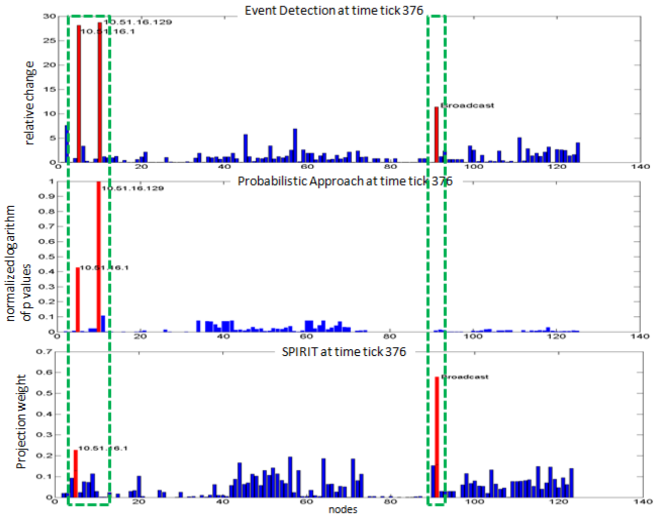

Figure 5: Anomalous nodes at time tick 376 from three algorithms. Agreements among different algorithms are marked with dashed green boxes.

Figure 5 shows the anomalous nodes for time tick 376 from all the three algorithms (from top to bottom: *EBED*, *PTSAD*, and *SPIRIT*). We realize that the results from different algorithms show considerable agreement. In particular, all the algorithms agree on IP '10.51.16.1' to be anomalous. In addition, we show the anomalous nodes for time tick 1126 in Figure 6 (as detected by *EBED* and *PTSAD*). They both agree on IP '10.50.10.14' to be anomalous, while *PTSAD* also flags several additional IPs as suspicious.

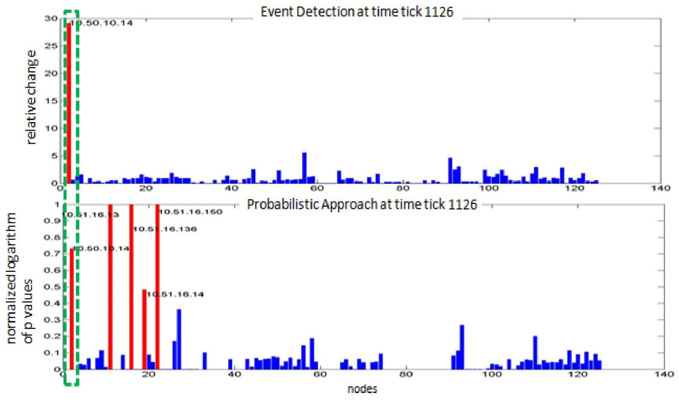

Figure 6: Anomalous nodes at time tick 1126 from *EBED* and *PTSAD*. The agreed-upon IP (node) is marked with a dashed green box.

Similarly, Figure 7 shows the anomalous edges for time tick 376 from all the algorithms. Again, we notice that the results from different algorithms have several agreements on the anomalous edges as well as individually identified ones. In particular, *EBED* and *SPIRIT* agree on edges '10.51.16.1'–'10.50.10.14' and '10.51.16.1'–'10.51.16.57'. In addition, *EBED* and *PTSAD* agree on edges '10.51.16.1'–'10.51.16.33' and '10.51.16.1'–'10.51.16.41'. Recall from Figure 5 that IP '10.51.16.1' is the top anomalous node detected

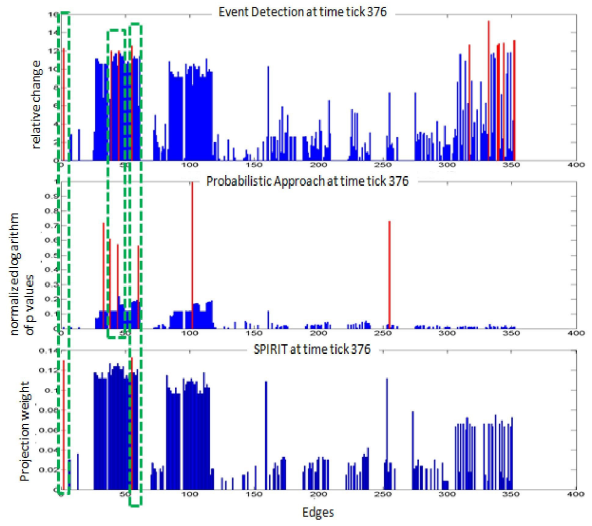

Figure 7: Anomalous edges at time tick 376 from three base algorithms. Agreements among the algorithms are marked with dashed green boxes.

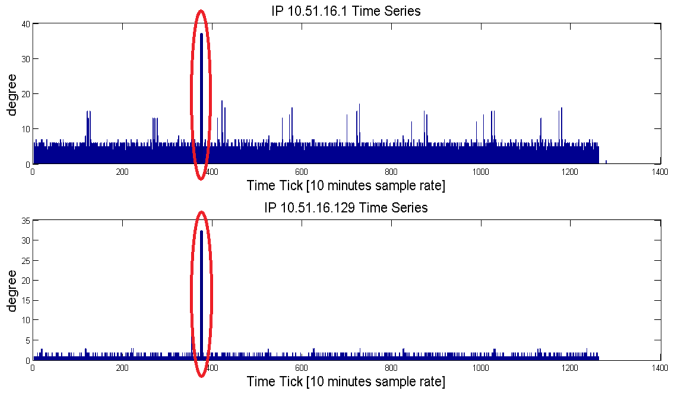

Figure 8: Time series of the top two anomalous nodes at time tick 376, when both IPs link to numerous other IPs.

by all the algorithms. The edge detection identifies the specific connections of those IPs that are most suspicious.

For further analysis we illustrate the time series of the top two node anomalies of time tick 376, i.e., IPs '10.51.16.1' and '10.51.16.129', in Figure 8. We notice that both nodes' connectivity increased significantly at the detected event. Figure 9 also depicts the graph visualization (from left to right) before, at, and after the event. From the figure we clearly see the change in the behavior of the two hosts; while being sparsely connected otherwise, they suddenly start communicating with many other hosts at time tick 376.

Essentially, the events and the specific agents identified as suspicious via our proposed approach correspond to a series of emergency event scenarios and their associated masterminds as simulated in our challenge network flow data. For data sharing agreement reasons, however, we are not allowed to elaborate on the specifics of the real-world events simulated in the data any further.

Table 2: Average Precision values for Event Detection [feature: Degree, sample rate: 10 minutes]

	Algorithms	AP
Base Algorithms	EBED	0.8333
	PTSAD	0.5722
	SPIRIT	0.7292
Consensus Rank Merging Algorithms	Inverse Rank	1.0000
	Kemeny Young	0.8095
	Unification(avg)	0.8056
	Unification(max)	0.7255
	Mixture Model(avg)	0.1684
	Mixture Model(max)	0.1684
	Final Ensemble	0.8667

Table 3: Average Precision values for Characterization [feature: Degree, sample rate: 10 minutes]

	Algorithms	Event: 376	Event: 1126
Base Algorithms	EBED	1.0000	1.0000
	PTSAD	1.0000	0.2500
	SPIRIT	0.3026	0.0213
Consensus Rank Merging Algorithms	Inverse Rank	1.0000	0.5000
	Kemeny Young	1.0000	0.2000
	Unification(avg)	1.0000	1.0000
	Unification(max)	0.8333	1.0000
	Mixture Model(avg)	1.0000	1.0000
	Mixture Model(max)	1.0000	1.0000
	Final Ensemble	1.0000	1.0000

5.1.3 *Quantitative Results*

The ground truth of our Network traffic flow data contains two major events (time tick 376 and 1126) and three associated nodes (IP '10.51.16.1', IP '10.50.10.14' and IP '10.51.16.129') responsible for those events. Table 2 and Table 3 present the *average precision* (AP) values of event detection and characterization phases, respectively, both for the individual base algorithms as well as the consensus approaches and the final ensemble (for sample rate 10 minutes, using feature *degree*). We note that the final ensemble for event detection has higher AP than all three individual ensemble components (in Table 2). Although the individual consensus approach of inverse rank based merging has higher AP than the final ensemble, the relation is reverse in characterization phase of time tick 1126 (in Table 3). Similarly, performance of mixture-modeling based consensus approach differs notably for detection versus characterization. These show that there is no single detector or consensus approach that consistently performs the best. On the other hand, the AP values for the final ensemble show that it is as good as the best individual base algorithm. Therefore, our ensemble framework proves to be an effective approach for event detection and characterization, which provides better or as good results as the best individual component.

5.2 Dataset 2: Results

5.2.1 *Event Detection*

Figure 10 shows the top ranking time points (with red bars) from all three techniques used in our ensemble with weighted degree feature to find anomalous events in NYT

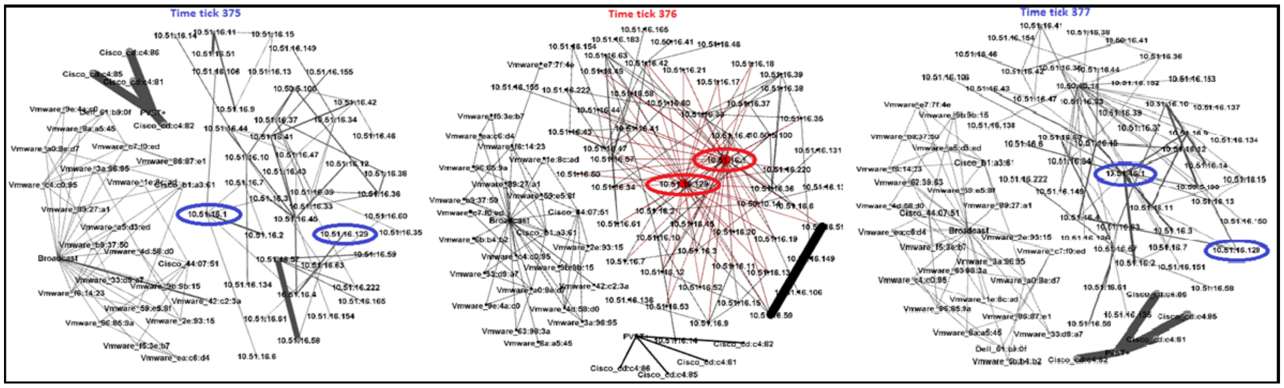

Figure 9: Cyber Challenge Network visualized (from left to right) before, at, and after the detected event at time tick 376. Notice the behavioral change of the two anomalous nodes detected in characterization step.

dataset(from top to bottom: *EBED*, *PTSAD*, and *SPIRIT*). We observe that time ticks 61, 62 are detected as change points by all the three algorithms. *EBED* and *SPIRIT* also agree on time ticks 90 and 162. The final ensemble dig out some important real-world events, such as, time ticks 61, 62 represent the events after the presidential election of USA in 2001 when George W. Bush was elected as president and was mentioned a lot in news articles, time tick 90 represents the 9/11 World Trade Center (WTC) terrorist attack in 2001 and time tick 162 represents the space shuttle Columbia (sent by NASA) disaster in 2003. There also exist some events that only individual methods detect. Our ensemble also includes those in the final ordering.

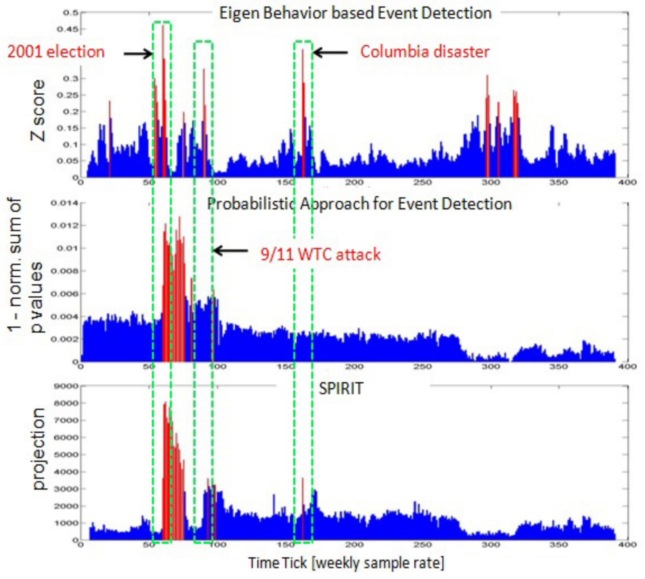

Figure 10: Change point detection in time series of NYT data for all three algorithms in our ensemble approach. Dashed green boxes depict three important real-world events ranked at the top.

5.2.2 *Characterization*

The characterization phase also finds entities associated or responsible for the detected events. For example, characterization of event 162 was able to find the seven astronauts of NASA who were killer in Columbia Space Shuttle disaster. Figure 11 shows the visualization of the change from week 161 to week 162 where suddenly the seven astronauts were highly co-mentioned with each other. We skip the detailed results of characterization phase for space limitation.

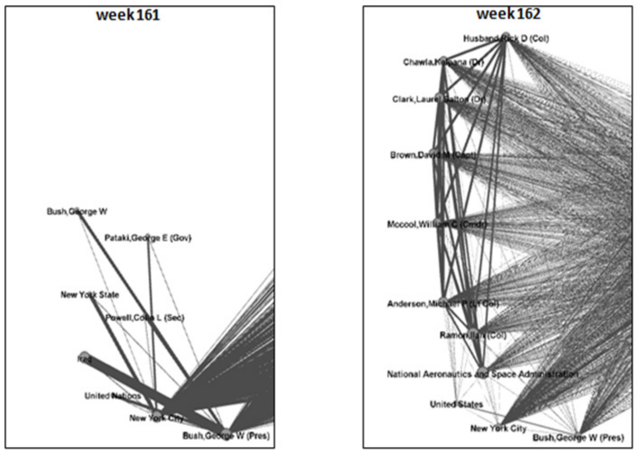

Figure 11: Characterization for 2003 Columbia Disaster. Visualization of graph structure change from time tick 161 to time tick 162 in NYT data, where a clique of NASA and seven astronauts emerges.

6. DISCUSSION

In this work, we proposed an *ensemble* approach for detecting and characterizing events in dynamically evolving graph data. This is the first work in ensemble design with dynamic graph based event detection algorithms as individual components. Our ensemble framework incorporates three different techniques; an eigen-behavior tracking based approach that we propose, a probabilistic model fitting approach, and a trend extraction based approach. Our method combines the findings in a unified manner to obtain a final consensus on the anomalousness of the time points. A key aspect of our proposed method is its focus on characterization; in addition to spotting suspicious change points, we also pinpoint the specific nodes and edges in the network that are most responsible for the changes. These are often considered as the specific entities associated with the detected events. Finally, our framework is amenable to incorporating additional techniques that exhibit event detection, ranking, and characterization properties.

We validated our proposed method on a simulated cyber network traffic dataset of hundreds of network hosts and their interactions. Our approach has successfully detected several change points in the cyber flows, as well as unearthing the hosts responsible for those changes with high accuracy. Additional experiments on a real NYT news corpus identified major events across the seven years span of the dataset and the key entities involved in those events. While our quantitative results showed that the ensemble approach outperforms the individual base algorithms and provides more accurate results, the qualitative results revealed events that agree with human interpretation.

While our experimental results have provided evidence that our proposed ensemble approach is successful for event detection and characterization with high performance in dynamic graphs both in simulated dataset with small ground truth and real dataset with large volume, further work is needed to fully analyze its performance on other large real datasets with more ground truth anomalies. Our future research will incorporate additional algorithms into the ensemble and investigate means to estimate detector accuracies and utilize weights proportional to these accuracies in combining their results. We will also extend our ensemble approach to outlier detection for high dimensional data points as well as other anomaly mining settings.

Acknowledgments

This material is based upon work supported by the Army Research Office under Contract No. W911NF-14-1-0029, an R&D grant from Northrop Grumman Aerospace Systems, an Office of Naval Research SBIR grant under Contract No. N00014-14-P-1155, and Stony Brook University Office of Vice President for Research. We thank Steve Cento and Tony Acosta for providing us with the network challenge dataset with ground truth. Any findings and conclusions expressed in this material are those of the author(s) and should not be interpreted as representing the official policies or views, either expressed or implied, of the funding parties.

7. REFERENCES

[1] C. C. Aggarwal. Outlier ensembles. In *ACM SIGKDD Explorations*, 2012.

[2] R. J. Bolton, D. J. Hand, and D. J. H. Statistical fraud detection: A review. In *Statistical Science*, 2002.

[3] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: Identifying density-based local outliers. In *ACM SIGMOD*, 2000.

[4] H. Bunke and K. Shearer. A graph distance metric based on the maximal common subgraph. In *Pattern Recognition Letters*, pages 255–259, 1998.

[5] A. Cameron and P. Trivedi. *Regression Analysis of Count Data*. Cambridge Univ. Press, 1st edition, 1998.

[6] D. H. Chau, S. Pandit, and C. Faloutsos. Detecting fraudulent personalities in networks of online auntioneers. In *PKDD*, pages 103–114, 2006.

[7] T. G. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems*, volume 1857, pages 1–15, 2000.

[8] G. Frobenius. Ueber matrizen aus nicht negativen elementen. In *Sitzungsber. Königl. Preuss. Akad. Wiss.*, 1912.

[9] R. Fujimaki, T. Yairi, and K. Machida. An approach to spacecraft anomaly detection problem using kernel feature space. In *PAKDD*, pages 401–410, 2005.

[10] J. Gao and P.-N. Tan. Converting output scores from outlier detection algorithms into probability estimates. In *ICDM*, 2006.

[11] J. Ghosh and A. Acharya. Cluster ensembles: Theory and applications. pages 551–570. 2013.

[12] N. A. Heard and D. J. Weston. Bayesian anomaly detection methods for social networks. In *The Annals of Applied Statistics*, pages 645–662, 2010.

[13] T. Ide and H. Kashima. Eigenspace-based anomaly detection in computer systems. In *ACM KDD*, 2004.

[14] P. Iglesias, E. Müller, F. Laforet, F. Keller, and K. Böhm. Statistical selection of congruent subspaces for outlier detection on attributed graphs. In *ICDM*, 2013.

[15] W. Jin, A. K. H. Tung, J. Han, and W. Wang. Ranking outliers using symmetric neighborhood relationship. In *PAKDD*, pages 577–593, 2006.

[16] J. Kemeny. Mathematics without numbers. In *Daedalus*, pages 577–591, 1959.

[17] D. Kifer, S. Ben-David, and J. Gehrke. Detecting change in data streams. In *VLDB*, 2004.

[18] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. Interpreting and unifying outlier scores. In *SDM*, 2011.

[19] D. Lambert. Zero-inflated poisson regression with an application to defects in manufacturing. In *Technometrics*, pages 1–14, 1992.

[20] A. Lazarevic and V. Kumar. Feature bagging for outlier detection. In *KDD*, pages 157–166, 2005.

[21] H. V. Nguyen, H. H. Ang, and V. GopalKrishnan. Minning outliers with ensemble of heterogeneous detectors on random subspaces. In *DASFAA*, 2010.

[22] S. Papadimitriou, H. Kitagawa, P. B. Gibbons, and C. Faloutsos. Loci: Fast outlier detection using local correlation intergal. In *ICDE*, 2003.

[23] S. Papadimitriou, J. Sun, and C. Faloutsos. Streaming pattern discovery in multiple time-series. In *VLDB*, pages 697–708, 2005.

[24] O. Perron. Zur theorie der matrices. In *Mathematische Annalen*, 1907.

[25] M. D. Porter and G. White. Self-exciting hurdle models for terrorist actovity. In *The Annals of Applied Statistics*, pages 106–124, 2012.

[26] E. Sandhaus. The new york times annotated corpus ldc2008t19. In *Linguistic Data Consortium*, 2008.

[27] T. Seidl, E. Müller, I. Assent, and U. Steinhausen. Outlier detection and ranking based on subspace clustering. In *Uncertainty Manag. in Info. Sys.*, 2009.

[28] K. Sequeira and M. Zaki. Admit: Anomaly-based data mining for intrusions. In *ACM SIGKDD*, 2002.

[29] P. Shoubridge, M. Kraetzl, W. Wallis, and H. Bunke. Detection of abnormal change in a time series of graphs. *Interconnection Networks*, pages 85–101, 2002.

[30] J. Sun, C. Faloutsos, S. Papadimitriou, and P. S. Yu. Graphscope: Parameter-free mining of large time-evolving graphs. In *KDD*, 2007.

[31] G. Valentini and F. Masulli. Ensembles of learning machines. In *WIRN*, volume 2486, pages 3–22, 2002.

[32] Q. H. Vuong. Likelihood ratio tests for model selection and non-nested hypotheses. In *Econometrica*, 1989.