Predicting Virality on Networks Using Local Graphlet Frequency Distribution

Andrew Baas[1], Frances Hung[2], Hao Sha[3], Mohammad Al Hasan[4], and George Mohler[5]
[1]Dept. of Physics, Baylor University
[2]Dept. of Mathematics, Pomona College
[3,4,5]Dept. of Computer Science, Indiana University—Purdue University, Indianapolis
[1]Andrew_Baas@baylor.edu, [2]fwhe2015@MyMail.pomona.edu
[3]haosha@iupui.edu, [4]alhasan@cs.iupui.edu, [5]gmohler@iupui.edu

***Abstract*—The task of predicting virality has far-reaching consequences, from the world of advertising to more recent attempts to reduce the spread of fake news. Previous work has shown that graphlet distribution is an effective feature for predicting virality. Here, we investigate the use of aggregated edge-centric local graphlets around source nodes as features for virality prediction. These prediction features are used to predict expected virality for both a time-independent Hawkes model and an independent cascade model of virality. In the Hawkes model, we use linear regression to predict the number of Hawkes events and node ranking, while in the independent cascade model we use logistic regression to predict whether a k-size cascade will multiply by a factor X in size. Our study indicates that local graphlet frequency distribution can effectively capture the variances of the viral processes simulated by Hawkes process and independent-cascade process. Furthermore, we identify a group of local graphlets which might be significant in the viral processes. We compare the effectiveness of our methods with eigenvector centrality-based node choice.**

***Index Terms*—Hawkes process, Cascade process, local graphlets, virality**

I. INTRODUCTION

Network virality is how quickly and widely an event travels throughout a network. Studying virality is key to understanding event spread in real-life networks. In addition to predicting the virality of events given a single node or set of nodes, it allows people to preemptively pick nodes which increase the chances of events going viral. For example, by building a model which predicts whether a cascade from a certain node will go viral, a user can run that model on all nodes to predict which ones will yield the most viral event cascades. From predicting viral posts on social media (i.e. Facebook and Twitter) to finding strategic ambassadors for advertising products [1], the ability to predict successful event spread has many potentially profitable applications. Researchers have used virality to study the spread of subjects as diverse as memes [2] and computer viruses [3].

There are many different methods currently used to predict network virality; some of the most basic use global centrality measures like degree and eigenvector centrality [4]. Degree centrality, one of the most intuitive measures of node importance, assigns more importance to nodes with higher degree. Therefore, according to this centrality, events originating at high-degree nodes should spread farther than those originating at lower-degree nodes. One of the best global centrality measures is eigenvector centrality. In this centrality, high-scoring nodes are those which are linked to other high-scoring nodes (for example, a low-degree node with high-degree neighbors may have a higher centrality score than a high-degree node with low-degree neighbors). Such global centrality measures are indeed fairly reliable and effective in choosing viral nodes. However, in real-life applications, the whole network structure may not be available or good global centrality measures may be too time-consuming to calculate.

More recent research has used network sub-structures called graphlets to predict virality. By giving a more detailed picture of structures within a network than degree, studying graphlets allows for the utilization of more graph information than more-established structural properties. Global graphlets of size 3, 4, and 5 can be counted efficiently using GUISE [5] and a recent study [6] has shown that global graphlets can be used to predict Hawkes virality. Moreover, edge-wise local graphlets up to size-5 can also be counted efficiently by using a method named E-CLoG [7]. So far, however, the application of local rather than global graphlets as predictive features has not been explored.

By using local graphlet counts to predict event spread, our methods allow users to predict overall virality given only limited scope of the graph. In this paper, we present ways to find the most potentially viral starting nodes in two different models: the Hawkes process [8], [9], [10], [11] and independent cascade [12]. These models of virality differ in their inherent predictability and way of spreading, but our results indicate that the structural information revealed by local graphlets is a good predictor for both of them. We consider two questions central to network virality application: (1) whether we can predict the size of Hawkes process and independent cascade process, and (2) whether we can choose the best nodes to start the Hawkes process and independent cascade in order to maximize their size.

Our contributions to studying virality in this paper can be summarized as follows. In our first section, we find that local graphlet counts are strong predictors for the size of Hawkes process, as well as for identifying the top nodes which produce the largest event count. Using local graphlet counts, we can train linear regression models which predict ultimate event

count better than models using first-degree and second-degree counts. In the next section, we show that graphlet counts are also good at predicting whether an independent cascade will grow 10-fold, and using this, we propose an algorithm for predicting the N-best nodes in an independent cascade model. In this N-nodes problem, local graphlets identify alternate node groupings which produce high-count cascades similar or bigger in size to groupings found via eigenvector centrality. Overall, local graphlet counts are conclusively good features for virality in both the Hawkes and independent cascade models, and have potential for solving the N-best nodes problem. Another advantage to our approach is that we are able to find specific graphlets and graphlet patterns which characterize nodes with high virality potential.

II. BACKGROUND

A. Local Graphlets

Graphlets are small subsections of a network which are classified by their specific structure. Figure 1 shows the different graphlets of size 3, 4, and 5 used by the E-CLoG algorithm [7]. In this algorithm, the program counts some graphlet types and uses combinatorial methods to deduce the other graphlet counts.

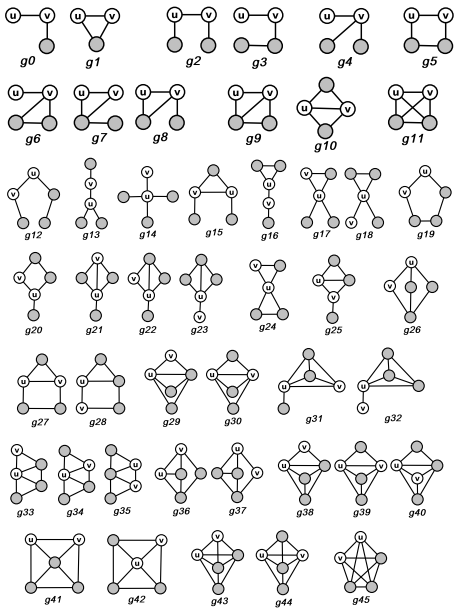


Fig. 1: 3,4,5-size local graphlets

B. Hawkes Process

Hawkes process is a special kind of point process where the conditional intensity function increases upon new events. For Hawkes process on a network G, given a sequence of events represented by (v_i, t_i) pairs where v_i is the node where an event occurs and t_i is when it occurs, the conditional intensity at node v, $\lambda_v(t)$, is of the form [8]

$$\lambda_v(t) = \mu + \sum_{\substack{t>t_i \\ v_i \in N(v)}} g(t - t_i) \tag{1}$$

where $\mu \geq 0$ is the base intensity and $N(v)$ is a set of neighbors of v. The triggering kernels $g(t - t_i)$ are summed over all events that occurred on a neighboring node v_i at a previous time t_i. Thus, the probability of a node being activated at a given time is dependent upon the probability distribution $g(t)$ and the number of neighboring nodes which have recently been activated.

We have decided to use the Hawkes process for this analysis both because it has been used in previous research connecting graphlets to virality [6] and because it has been able to be used successfully in many other areas of research. The Hawkes process has been applied in research in areas as important and diverse as the foreign exchange market [13], analysis of financial data [14], and molecular biology [15].

III. METHODS

A. Generating Simulation Graphs

As indicated in [7], degree distribution can partially explain the evolution of viral process in a network. To nullify the influence of degree distribution in our analysis, for each network in Table I we generate a collection of synthetic networks which have the same degree distribution as the original network. In particular, we adopt an edge-swapping method [16], [17] where two edges e_1 and e_2 are selected at random and their second vertices are swapped to generate two new edges e_3 and e_4. If e_3 or e_4 already exists, this proposed swap is rejected and the process is repeated with a new pair of randomly chosen edges e_1 and e_2. As a result, the degree of a vertex is invariant under edge-swapping and the overall degree distribution remains constant. A series of degree-preserving graphs can thus be generated for further analysis.

B. Modeling Virality

To model the virality of a network, we run a series of Hawkes processes [9], [10], [11] originating from every node of the network. The Hawkes process is a specific kind of self-exciting point process where discrete events occur according to a stochastic intensity $\lambda(t)$ that increases upon the arrival of a new event and decreases between two consecutive events. Our implementation of the Hawkes process starts from one active node. During each iteration, an active node has a chance to activate any of its neighbors. Any node can be activated at most once per iteration. Moreover, an active node can only attempt each neighbor once per iteration. Our Hawkes process ends when there are no more activated nodes. The virality can be measured by the total number of Hawkes events (node activations), HE, which have occurred in the lifetime of the process. Therefore, a large HE indicates high virality.

In our implementation of the Hawkes process, θ serves as a threshold, below which an inactive node would not be activated. Specifically, an active node can activate a neighbor if a randomly generated number $\in [0, 1)$ is not greater than θ. In general, θ has to be less than certain critical value θ_c for Hawkes process to converge. The critical value θ_c can be calculated by

$$\theta_c = 1/\lambda_m \tag{2}$$

where λ_m is the largest eigenvalue of the adjacency matrix A of a network G. To obtain λ_m, we adopt ARPACK [18], a software which is capable of solving a few eigenvalues for large sparse matrices efficiently. Alternatively, one can approximate λ_m by its upper bound. For instance, for graphs with power law degree distribution with exponent β, the largest eigenvalue of the adjacency matrix is almost surely approximately the square root of the maximum degree, $d_{max}^{1/2}$ if $\beta > 2.5$, and is almost surely approximately $cd_{max}^{3-\beta}$ if $2 < \beta < 2.5$, where c is a constant [19]. In order for Hawkes process to finish in finite steps, we define the θ value as the following

$$\theta = \alpha\theta_c \tag{3}$$

where $\alpha \in [0, 1]$.

In independent cascades, each edge in a network is assigned a random weight, which represents the probability of an event spreading through the edge. Additionally, each node can be activated once; each node can attempt to spread events to its neighbors only one time. In this paper, we assign random weights to the edges using a gaussian distribution ($\mu = 0.2$, $\sigma = 0.3$).

C. Counting Local Graphlets

To obtain local graphlet distribution by enumeration is expensive and not scalable for very large real-life networks. In fact, for a network of $|V|$ vertices, the brute-force complexity of counting local graphlets up to size 5 is $O(|V|^5)$. Alternatively, we adopt a hybrid algorithm E-CLoG [7] which obtains local graphlet distribution efficiently by combining enumeration and combinatorial calculation. E-CLoG counts all size 3, 4, and 5 local graphlets considering all possible edge orbits. Specifically, E-CLoG enumerates 4 out of 8 size-4 local graphlets and 14 out of 32 size-5 local graphlets, and generates the counts for the rest of the local graphlets in constant time through combinatorial calculation. Furthermore, E-CLoG can run in parallel, thus highly scalable. However, E-CLoG counts the local graphlets with respect to a given edge. To convert from edge-centric to node-centric, we sum up the local graphlet counts of all edges that connect a given node. To be specific, for vertex v_i, the node-centric count of the kth local graphlet can be calculated by

$$C(v_i)[k] = \sum_{j \in N(i)} C(e_{i,j})[k] \tag{4}$$

where $N(i)$ is a set of neighbors of v_i. For edge $e_{i,j}$ in network G, E-CLoG outputs a 42-dimensional vector $C(e_{i,j})$ whose kth element is the count of the kth local graphlet in Figure 1 (local graphlet figure). After converting the vectors from edge-centric to node-centric, we obtain a 42-dimensional vector $C(v_i)$ for node v_i, which would be the features (explanatory variables) for our regression.

D. Hawkes Process

1) Regression Model for Predicting Virality: Given a real-world network G_0, after degree-preserve-rewiring, we generate K rewired networks $G_1 \sim G_K$. For G_i, where i = 0, ..., K, we perform the following steps:

1) obtain θ_i for G_i using the method described above;
2) run Hawkes process from each node v_j in G_i, and record the Hawkes event count $HE_i(v_j)$;
3) run E-CLoG on G_i to obtain edge-centric local graphlet count $C_i(e_{j,k})$;
4) convert edge-centric local graphlet count $C_i(e_{j,k})$ to node-centric local graphlet counts $C_i(v_j)$;
5) repeat step 1 to 4 for i = 0, ..., K;
6) randomly select $70\%(K+1)$ networks as the training set to learn a linear regression model;
7) evaluate R^2 and MSE on the remaining 30% test set.

Specifically, to learn a linear regression model, we adopt the node-centric local graphlet distribution $C_i(v_j)$ as features and log Hawkes event count $\log(HE_i)$ as label. Therefore, our regression is of the form

$$log(HE_i(v_j)) = b_i + \sum_{k=0}^{41} b_k \times C_i(v_j)[k] + \epsilon \tag{5}$$

where b_i is the intercept and b_k are the coefficients of the linear regression where the errors are assumed to be normal.

2) Ranking: Starting Hawkes process from different nodes would result in different total Hawkes event counts. In real-life applications, people are usually interested in finding the top k nodes that give rise to the largest event spread. Given that our linear regression model is able to predict the total Hawkes event count HE_{pred}^i for the Hawkes process initiated at node v_i, we can thus rank v_i by HE_{pred}^i, for $i = 1, 2, ..., |V|$, and pick the top k nodes with the largest HE_{pred}. Alternatively, we can assign v_i a label $y_i = 1$, if its actual Hawkes event count HE_{true}^i is among the top k, otherwise a label $y_i = 0$. We can thus learn a logistic classifier with such binary labels.

E. Independent Cascades

1) Cascade Prediction: Our goal is to predict whether a cascade will grow by a certain factor or not, as it has been stipulated in previous research that actual cascade size is inherently difficult to predict [12]. In previous research, researchers used structural as well as temporal variables to correctly predict real-life cascade doubling with a high accuracy (80% for $k = 5$) [12]. In accordance to their approach, we use logistic regression to predict whether a k-size cascade will grow by a chosen factor (X) or not. We obtain our k-size cascades by the following process: we simulate independent cascade from a given number of random nodes until our time limit is reached or the cascade ends naturally. The first k nodes reached make up the k-size cascade spread.

We then use logistic regression to predict whether a given k-size cascade will grow by a certain factor. Cascade spread is very network specific: to determine the correct factor to use for a smaller k-size (i.e. $k = 6 \sim 20$, which is what most applications are interested in) we need to understand the distribution of cascade sizes in a network. For our applications, we test factors $X = \{10, 20\}$. Given a network, k value

of interest and factor X, our data-generating process is as follows:

1) Simulate a cascade from a random node(s) and determine whether it exceeds size k.
2) If it exceeds size k, take the first k nodes reached and find the extended subgraph considering up to 2-hop neighbors of the vertices of initial k-subgraph.
3) Ignoring edges between the k nodes, take the global graphlet count of the k nodes and their up to 2-hop neighbors. These are our graphlet features.
4) If the cascade size exceeds kX, then this cascade is a positive instance of growth for the logistic model; otherwise, it is a negative instance.
5) Repeat steps 1-4 p times. In this paper, we set $p = 100$.

Once we have our training and test data, we use 5-fold cross-validated logistic regression to return 5 ROC-AUC scores of our model. To gain an accurate idea of how well graphlet models predict growth, we randomly reassign weights and repeat the above process multiple p times. In this paper, we set $p = 15$. As a comparison method, we use the aggregated degree of the k nodes and up to 2-hop neighbors, excluding edges between the k nodes. This gives an equivalent of single-node degree centrality for multiple connected nodes.

2) Choosing N-Best Nodes: To choose the N-best nodes in a network, we implement LIR, a measure which gives a relative degree centrality of a node in comparison to its neighbors [20]. Say $G = (V, E)$ is a network with vertices V and edges E. Letting v_i be our node of interest, d_i its degree, and $N(v_i) = \{v_j | (v_i, v_j) \in E\}$, the corresponding LIR score is

$$L(v_i) = \sum_{v_j \in N(v_i)} Q(d_j - d_i) \tag{6}$$

where $Q(d_j - d_i) = 1$ if $d_j - d_i > 0$ and $Q(d_j - d_i) = 0$ otherwise. We call nodes with a LIR score of 0 0-LIR nodes. Identifying the 0-LIR nodes as centers of communities in a network, we run cascade simulations from each of the 0-LIR nodes. By taking the graphlet count features as described in the methods section, we can predict which nodes have the highest probability of growing by factor X.

1) Identify the 0-LIR nodes and choose an existing cascade prediction model to apply. We ideally use a logistic regression model trained on the network of interest via the process in Section F, with an initial size k and growth factor X of interest.
2) For each node, run a cascade simulation with the k used in the chosen model; find the graphlet count of the k nodes and the up to 2-hop neighbors using GUISE[5].
3) Use the logistic regression model to predict the probability that the cascade emanating from that node will grow by X.
4) Once probabilities have been calculated for all 0-LIR nodes, take the top N nodes with the highest probabilities of growing.
5) Repeat so that each node has a cascade simulated from it j times. In this paper, we set $j = 5$. Pick the top N occurring nodes of the aggregated cascade runs as the N nodes we activate initially.

We compare the performance of this procedure to nodes chosen via eigenvector centrality.

IV. Results

A. Hawkes Process

1) Linear regression: For our experiments, 12 real-world networks from two domains are collected from the Network Repository [21]. Among them, socfb-Caltech36, socfb-Reed98, socfb-Haverford76, socfb-Simmons81, socfb-Swarthmore42, and socfb-Bowdoin47 are Facebook friendship networks, while soc-dolphins, soc-wiki-vote, soc-hamsterster, soc-advogato, soc-anybeat, and soc-gplus are social networks. Some basic statistics such as number of vertices ($|V|$), number of edges ($|E|$), largest degree (d_{max}), and average degree (d_{avg}) for these networks are shown in Table I.

TABLE I: networks adopted in experiments

Network	$\|V\|$	$\|E\|$	d_{max}	d_{avg}
socfb-Caltech36	769	17K	248	43
socfb-Reed98	962	19K	313	39
socfb-Haverford76	1K	60K	375	82
socfb-Simmons81	2K	33K	300	43
socfb-Swarthmore42	2K	61K	577	73
socfb-Bowdoin47	2K	84K	670	74
soc-dolphins	62	159	12	5
soc-wiki-vote	889	3K	102	6
soc-hamsterster	2K	17K	273	13
soc-advogato	5K	47K	947	18
soc-anybeat	13K	67K	9K	10
soc-gplus	24K	392K	3K	3.32

By degree-preserving rewiring, we generate 92 rewired networks for each real-world network. For each rewired networks along with the original network, we obtain the largest eigenvalues of their adjacency matrices using ARPACK [18]. Correspondingly, θ can be calculated using Eq. 2 and 3. In our simulations, we adopt $\alpha = 0.99$ for all networks so that Hawkes process can finish in finite steps. It is worth mentioning that instead of using θ of the original network for all rewired networks like in [6], we calculate θ for each rewired network. Had adopted a fix θ, Hawkes process on some rewired networks which have small θ_c might never converge. On the other hand, Hawkes process would finish quickly after few events on rewired networks with large θ_c.

Next we apply E-CLoG [7] on each network including the original and the rewired. Then we convert the resulting edge-centric local graphlet (LG) distribution to node-centric local graphlet distribution which is used as a feature set. In addition, we adopt log Hawkes event count $log(HE)$ as a label set. As mentioned in the Method section, Hawkes process is invoked from every node for 100 times, and HE is the average node-base Hawkes event count. For linear regression, we use 70% training set and 30% test set. The R^2 score and mean square error MSE of each real-world network are shown in Table IV. The regression results are in general very good - except for soc-gplus, all networks have R^2 higher than 90%.

In particular, for the Facebook networks, the R^2 scores are all higher than 95%. Moreover, MSE are small for all networks. As a baseline, we learn linear models using first order degree distribution (FD) which is essentially the degree distribution of each graph and second order degree distribution (SD) which is defined as $d_i\theta + \sum_{j \in N(d_i)} d_j\theta^2$ where the first term is the degree d_i of node v_i scaled by θ and the second term is the sum of the degree d_j of all the neighbors of v_i scaled by θ^2. As shown in Table II, LG outperforms FD and SD in terms of R^2 and MSE.

With a linear regression model in hand, we proceed to identify which local graphlets are predictive of viral process (i.e. Hawkes event count). The counts of local graphlets can be viewed as independent variables in our linear regression model, therefore their p-values can be calculated. In Table III, we list the local graphlets which are significant at the .01 level. It appears that most of the local graphlets are significant in predicting the Hawkes event count. In particular, for soc-hamsterster, all local graphlets are significant at the .01 level.

To further identify the local graphlets with the most predictive power, we calculate the increase in R^2 that each local graphlet produces when it is added to the linear regression model. We start with the local graphlet that gives the largest R^2 when it is the only feature. We then identify the next local graphlet that raises R^2 the most when added on top of the first feature. Along this line, we rank local graphlets by the amount of unique variance in addition to those before them. In Table IV, we list the top 4 local graphlets of each real-network. It appears that most Facebook networks have $g0$ and $g4$ as important local graphlets, while most social networks have $g4$ and $g32$ play important roles. We also perform linear regression using the top 4 local graphlets as the feature set, and it turns out that for the various Facebook networks more than 80% of the variance is captured, while for the social networks the R^2 score ranges from about 50% to 90% (Table IV). In addition, MSE remains small despite that only top 4 local graphlets are adopted (Table IV).

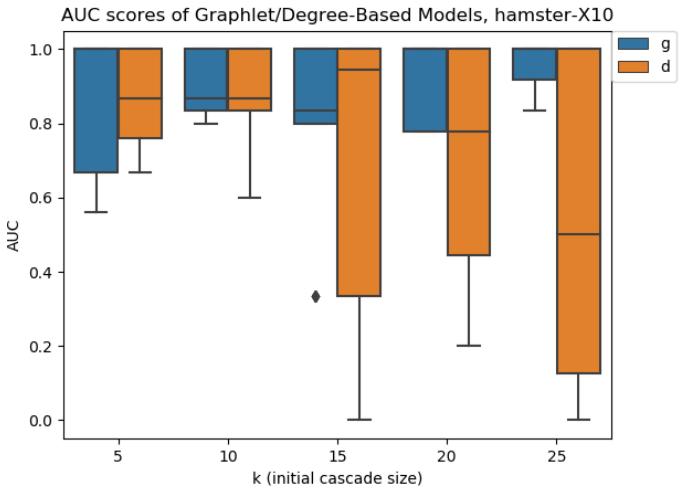
2) Ranking: We further test our linear regression model's ability to identify the k most influential nodes. Specifically, we choose $k = 10$. We rank node v_i by the predicted Hawkes event counts HE^i_{pred}. On the other hand, we rank v_i by their actual Hawkes event counts HE^i_{true}. Let the top 10 nodes given by the former be $U = \{u_1, u_2, ..., u_{10}\}$ and the later be $U' = \{u'_1, u'_2, ..., u'_{10}\}$, where $u_i, u'_i \in V$. We can then calculate the size of their intersection, $|U \cap U'|$ as an estimation to the hit rate (out of 10). As shown in Table V, our linear regression model correctly predict at least 8 out of 10 top nodes for the six networks tested. In particular, for the three social networks (soc-wiki-vote, soc-dolphins, and soc-hamsterster), we achieve a hit rate about 9 out of 10.

As a comparison, we construct a logistic regression model with local graphlet counts as features and whether a node is among the top 10 as labels. Specifically, we label a node 1 if it is one of the top 10 nodes, and 0 otherwise. We use the same dataset and train-test split (30% test set) as the linear regression model. For the six networks tested, this logistic regression model correctly predicts more than 6 out of 10 top nodes (Table V), and for the three social networks the hit rates are higher than 85%. However, as indicated in Table V, the logistic model is outperformed by the linear regression model in most cases.

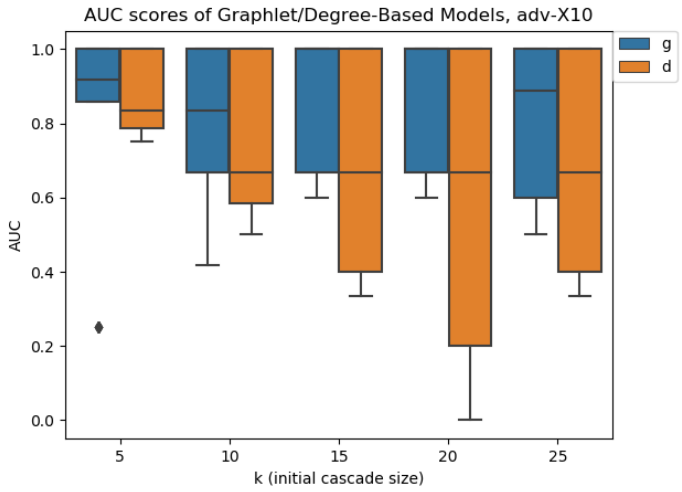
B. Independent Cascades

1) Cascade Logistic Regression: We use 3 real-world social networks and 2 other domain networks from the ones used for Hawkes linear regression: socfb-Caltech36, socfb-Reed98, socfb-Haverford76, soc-Hamsterster, and soc-advogato. To compare performance with an alternative prediction method, we create another logistic regression model using the sum of degrees of an initial cascade's nodes and up to 2-hop neighbors as a feature. This aggregated degree feature is meant to represent the multi-node equivalent of a degree centrality metric. Especially for smaller values of k, logistic regression using graphlets as features outperforms this aggregated degree feature on average (Fig. 2). There appears to be different optimal X for different networks; values of X which are too large lead to poor AUC values for both graphlet and aggregated degree features (Fig. 2e), while X which are too small often lack enough negative classification instances to build a meaningful model. However, in all our tested networks, graphlet features on average outperformed aggregated degree and AUC didn't fall below 0.7, for at least one of the two X values. The fact that graphlet features are able to predict 10-fold growth with relatively high accuracy for all our networks indicates they are useful predictors of virality.

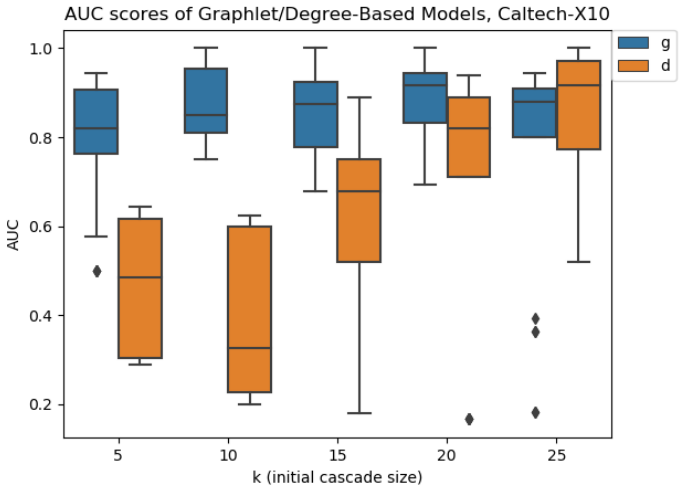
2) Cascade: Best N Nodes: Given a generated logistic model, we apply it to all 0-LIR nodes in a network to determine the most significant independent nodes (if there are fewer 0-LIR nodes than N, we extend our search to 1-LIR nodes). Aggregating the N most common important nodes (we test $N = 3, 4, and 5$), we then run independent cascades originating from those N nodes. Compared to the top N nodes found via eigenvector centrality, this approach works better or about as well in sparse networks with many 0-LIR values and does identically to eigenvector centrality in networks with fewer 0-LIR values (where we also include the 1-LIR values). Even in cases where the algorithm does identically to eigenvector centrality, different nodes are often chosen, which gives users useful alternatives to nodes chosen through eigenvector centrality. When the narrowing-down process was changed from 0 and 1-LIR nodes to the 50 most-central nodes via eigenvector centrality, the graphlet counting process consistently performed worse. This indicates that in most graphs, eigenvector centrality undervalues nodes with high potential of cascade spread which our method is able to discover. The fact that our method usually doesn't outperform the eigenvector method is likely due to our initial narrowing down of nodes to be tested (from all nodes to 0 and 1-LIR nodes). It is too temporally expensive to test all nodes, so this algorithm would benefit from a more effective narrowing-down process.



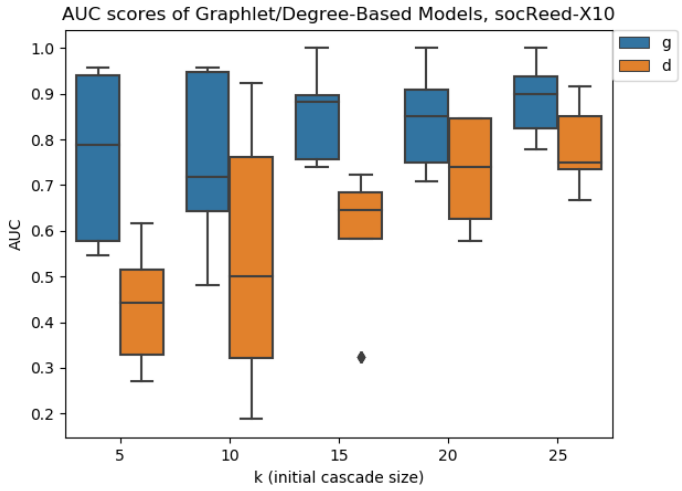
(a) Hamsterster, test growth factor X=10



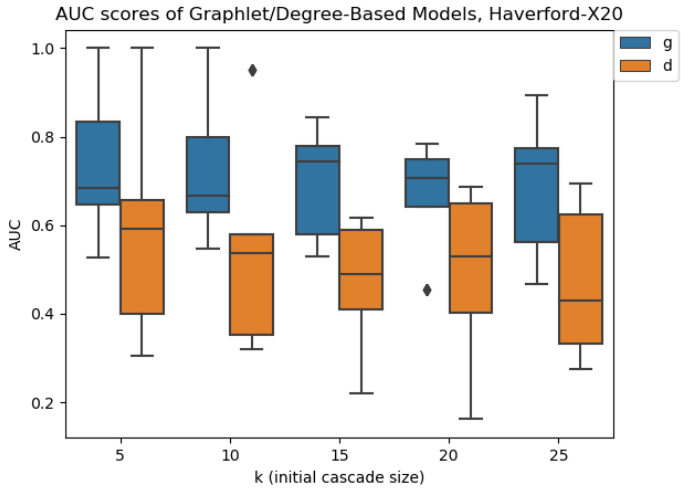
(b) Advogato, test growth factor X=10



(c) Caltech, test growth factor X=10



(d) Reed, test growth factor X=10



(e) Haverford, test growth factor X=20

Fig. 2: Comparison of cascade growth prediction performance between graphlet and aggregate-degree features.

TABLE II: Linear regression for Local graphlet and Degree distribution

Network	LG R^2	LG MSE	FD R^2	FD MSE	SD R^2	SD MSE
socfb-Caltech36	0.984	0.009	0.786	0.126	0.807	0.114
socfb-Reed98	0.978	0.012	0.768	0.121	0.793	0.108
socfb-Haverford76	0.976	0.013	0.787	0.112	0.802	0.104
socfb-Simmons81	0.971	0.016	0.751	0.136	0.769	0.126
socfb-Swarthmore42	0.970	0.016	0.763	0.123	0.784	0.111
socfb-Bowdoin47	0.963	0.021	0.748	0.143	0.767	0.132
soc-dolphins	0.943	0.011	0.869	0.025	0.909	0.017
soc-wiki-vote	0.935	0.023	0.659	0.119	0.748	0.088
soc-hamsterster	0.952	0.025	0.695	0.156	0.739	0.134
soc-advogato	0.936	0.030	0.635	0.168	0.712	0.133
soc-anybeat	0.925	0.021	0.146	0.236	0.485	0.142
soc-gplus	0.819	0.022	0.251	0.093	0.468	0.066

TABLE III: Important variables

Network	Important variables (.01 level)
socfb-Caltech36	g0-7,g9-16,g18,g19,g21-23,g25,g27-29,g31-36,g38-41
socfb-Reed98	g0-20,g22-34,g36-38,g40,g41
socfb-Haverford76	g0-3,g6-12,g14-18,g20,g21,g23-30,g33-40
socfb-Simmons81	g0-7,g10-24,g26-30,g32-41
socfb-Swarthmore42	g0-23,g25-29,g31-41
socfb-Bowdoin47	g0-31,g33-36,g38-40
soc-dolphins	g0-6,g9-14,g16,g18,g21-23,g27,g28,g33,g34,g39,g41
soc-wiki-vote	g0-28,g30-37,g39-41
soc-hamsterster	g0-41
soc-advogato	g0-17,g19-36,g38-41
soc-anybeat	g0-38,g40,g41
soc-gplus	g0-10,g12-41

TABLE IV: linear regression results

Network	R^2	MSE	Top 4 graphlets	Top 4 graphlets R^2	Top 4 graphlets MSE
socfb-Caltech36	0.984	0.009	g4,g32,g0,g10	0.950	0.029
socfb-Reed98	0.978	0.012	g4,g32,g34,g13	0.817	0.096
socfb-Haverford76	0.976	0.013	g0,g10,g2,g28	0.952	0.025
socfb-Simmons81	0.971	0.016	g0,g10,g2,g26	0.921	0.043
socfb-Swarthmore42	0.970	0.016	g4,g23,g0,g12	0.906	0.049
socfb-Bowdoin47	0.963	0.021	g4,g22,g0,g10	0.874	0.072
soc-dolphins	0.943	0.011	g0,g10,g12,g1	0.925	0.014
soc-wiki-vote	0.935	0.023	g4,g23,g18,g1	0.677	0.113
soc-hamsterster	0.952	0.025	g4,g23,g5,g13	0.728	0.139
soc-advogato	0.936	0.030	g4,g32,g34,g35	0.650	0.161
soc-anybeat	0.925	0.021	g4,g32,g24,g23	0.466	0.148
soc-gplus	0.819	0.022	g33,g34,g35,g32	0.449	0.068

TABLE V: top 10 ranking results

Network	Logistic	Linear
socfb-Caltech36	7.3	8.1
socfb-Reed98	6.7	8.0
socfb-Haverford76	6.9	8.0
soc-wiki-vote	8.9	8.8
soc-dolphins	8.5	9.0
soc-hamsterster	8.5	9.0

V. CONCLUSIONS

In this work, we propose a linear regression model for predicting Hawkes process event count and a logistic regression model for forecasting the growth of independent cascade process using local graphlet distribution. We show that local graphlet distribution outperforms other topological metrics for predicting Hawkes event count in terms of accuracy. We also rank the local graphlets by their contributions to the total variance of the model and discover that networks of the same kind share similar important local graphlets. Graphlet counts have potential in determining N-best nodes for independent cascades, performing on par with eigenvector centrality while

uncovering different combinations of nodes. However, a more efficient way of finding these nodes should be researched more. Overall, our paper concludes that local graphlet features are applicable for predicting virality and provides viable models for prediction.

ACKNOWLEDGMENT

This work was supported in part by NSF grants ATD-1737996, REU-1343123, and SCC-1737585. Most of our calculations were conducted on the Big Red II supercomputer at Indiana University and a Linux cluster at School of Science, Indiana University-Purdue University Indianapolis.

REFERENCES

[1] O. Gil-Or, "Building consumer demand by using viral marketing tactics within an online social network," 2010.

[2] L. Weng, F. Menczer, and Y.-Y. Ahn, "Virality prediction and community structure in social networks," *Scientific Reports*, vol. 3, 2013. [Online]. Available: http://dx.doi.org/10.1038/srep02522

[3] D. Chakrabarti, Y. Wang, C. Wang, J. Leskovec, and C. Faloutsos, "Epidemic thresholds in real networks," *ACM Trans. Inf. Syst. Secur.*, vol. 10, no. 4, pp. 1:1–1:26, Jan. 2008. [Online]. Available: http://doi.acm.org/10.1145/1284680.1284681

[4] A. Landherr, B. Friedl, and J. Heidemann, "A critical review of centrality measures in social networks," *Business & Information Systems Engineering*, vol. 2, no. 6, pp. 371–385, Dec 2010. [Online]. Available: https://doi.org/10.1007/s12599-010-0127-3

[5] M. A. Bhuiyan, M. Rahman, M. Rahman, and M. A. Hasan, "Guise: Uniform sampling of graphlets for large graph analysis," in *2012 IEEE 12th International Conference on Data Mining*, Dec 2012, pp. 91–100.

[6] S. Khorshidi, M. Al Hasan, G. Mohler, and M. B. Short, "The role of graphlets in viral processes on networks," *Journal of Nonlinear Science*, May 2018. [Online]. Available: https://doi.org/10.1007/s00332-018-9465-y

[7] V. S. Dave, N. K. Ahmed, and M. A. Hasan, "E-clog: Counting edge-centric local graphlets," in *2017 IEEE International Conference on Big Data (Big Data)*, Dec 2017, pp. 586–595.

[8] A. G. Hawkes, "Spectra of some self-exciting and mutually exciting point processes," *Biometrika*, vol. 58, no. 1, pp. 83–90, 1971. [Online]. Available: http://www.jstor.org/stable/2334319

[9] R. Crane and D. Sornette, "Robust dynamic classes revealed by measuring the response function of a social system," *Proceedings of the National Academy of Sciences*, vol. 105, no. 41, pp. 15 649–15 653, 2008. [Online]. Available: http://www.pnas.org/content/105/41/15649

[10] M. B. Short, G. O. Mohler, P. J. Brantingham, and G. E. Tita, "Gang rivalry dynamics via coupled point process networks," *Discrete & Continuous Dynamical Systems - B*, vol. 19, no. 1531-3492_2014_5_1459, p. 1459, 2014. [Online]. Available: http://aimsciences.org//article/id/cd177559-9306-4b44-881d-fb279ce2f432

[11] Q. Zhao, M. A. Erdogdu, H. Y. He, A. Rajaraman, and J. Leskovec, "SEISMIC: A self-exciting point process model for predicting tweet popularity," *CoRR*, vol. abs/1506.02594, 2015. [Online]. Available: http://arxiv.org/abs/1506.02594

[12] J. Cheng, L. A. Adamic, P. A. Dow, J. M. Kleinberg, and J. Leskovec, "Can cascades be predicted?" *CoRR*, vol. abs/1403.4608, 2014. [Online]. Available: http://arxiv.org/abs/1403.4608

[13] M. Rambaldi, P. Pennesi, and F. Lillo, "Modeling foreign exchange market activity around macroeconomic news: Hawkes-process approach," *Phys. Rev. E*, vol. 91, p. 012819, Jan 2015. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.91.012819

[14] P. Embrechts, T. Liniger, and L. Lin, "Multivariate hawkes processes: an application to financial data," *Journal of Applied Probability*, vol. 48, no. A, p. 367378, 2011.

[15] L. Carstensen, A. Sandelin, O. Winther, and N. R. Hansen, "Multivariate hawkes process models of the occurrence of regulatory elements," *BMC Bioinformatics*, vol. 11, no. 1, p. 456, Sep 2010. [Online]. Available: https://doi.org/10.1186/1471-2105-11-456

[16] C. Gkantsidis, M. Mihail, and E. Zegura, "The markov chain simulation method for generating connected power law random graphs," in *In Proc. 5th Workshop on Algorithm Engineering and Experiments (ALENEX). SIAM*, 2003.

[17] S. Maslov and K. Sneppen, "Specificity and stability in topology of protein networks," *Science*, vol. 296, no. 5569, pp. 910–913, 2002. [Online]. Available: http://science.sciencemag.org/content/296/5569/910

[18] R. Lehoucq, D. Sorensen, and C. Yang, "Arpack users' guide," 1998. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9780898719628

[19] F. Chung, L. Lu, and V. Vu, "Spectra of random graphs with given expected degrees," *Proceedings of the National Academy of Sciences*, vol. 100, no. 11, pp. 6313–6318, 2003. [Online]. Available: http://www.pnas.org/content/100/11/6313

[20] D. Liu, Y. Jing, J. Zhao, W. Wang, and G. Song, "A fast and efficient algorithm for mining top-k nodes in complex networks," *Scientific Reports*, vol. 7, 2017. [Online]. Available: http://dx.doi.org/10.1038/srep43330

[21] R. A. Rossi and N. K. Ahmed, "The network data repository with interactive graph analytics and visualization," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015. [Online]. Available: http://networkrepository.com