Learning and Predicting the Evolution of Social Networks

Björn Bringmann, *Katholieke Universiteit Leuven*

Michele Berlingerio, *ISTI-CNR*

Francesco Bonchi and Aristides Gionis, *Yahoo Research*

Graph Evolution Rules help in analyzing the evolution of large networks and can be used to predict the future creation of links among nodes.

With the increasing availability of large social network data, there is also an increasing interest in analyzing how those networks evolve over time.[1] Traditionally, the analysis of social networks has focused only on a single snapshot of a network. Researchers have already verified that social networks follow power-law degree distribution,[2] have a small diameter, and exhibit small-world structure[3] and community structure.[4] Attempts to explain the properties of social networks have led to dynamic models inspired by the *preferential attachment model*,[5] which assumes that new network nodes have a higher probability of forming links with high-degree nodes, creating a "rich-get-richer" effect.

Recently several researchers have turned their attention to the evolution of social networks at a global scale. For example, Jure Leskovec and his colleagues empirically observed that networks become denser over time, in the sense that the number of edges grows superlinearly with the number of nodes.[6] Moreover, this densification follows a power-law pattern. They reported that the network diameter often shrinks over time, in contrast to the conventional wisdom that such distance measures should increase slowly as a function of the number of nodes.

Although some effort has been devoted to analyzing global properties of social network evolution, not much has been done to study graph evolution at a microscopic level. A first step in this direction investigated a variety of network formation strategies,[7] showing that edge locality plays a critical role in network evolution.

We propose a different approach. Following the paradigm of association rules and frequent-pattern mining, our work searches for typical patterns of structural changes in dynamic networks. Mining for such local patterns is a computationally challenging task that can provide further insight into the increasing amount of evolving network data. Beyond the notion of *graph evolution rules* (GERs), a concept that we introduced in an earlier work,[8] we developed the Graph

Evolution Rule Miner (GERM) software to extract such rules and applied these rules to predict future network evolution.[9,10]

The merits of our approach go beyond descriptive analysis. In many cases, frequent patterns are used as basic blocks to build more complex data-mining solutions. Our framework can help predict edges among old and new nodes and predict when a new edge is expected to appear. Even when comparing directly with the classic link-prediction problem,[9] we show that simple scores based on our evolution rules represent important features providing good prediction performances.

Rules of Graph Evolution

We use the term $G = (V, E, \lambda)$ to denote a graph G over a set of nodes V and edges $E \subseteq V \times V$. The function $\lambda: V \cup E \mapsto \Sigma$ is an assignment from graph nodes and edges to labels from an alphabet Σ. Those labels represent properties, and for simplicity, we assume that they do not change over time. For example, in a social network nodes are users (with properties such as gender, country, and college), while edge labels might represent the type of link (such as friends or family).

We conceptually represent the graph evolution using a series of graphs $G_1, \ldots, G_T$, so that $G_t = (V_t, E_t)$ represents the graph at time t. Because $G_1, \ldots, G_T$ represent different snapshots of the same graph, we have $V_t \subseteq V$ and $E_t \subseteq E$. For simplicity, we assume that nodes and edges are only added and never deleted as the graph evolves—that is, $V_1 \subseteq V_2 \subseteq \ldots V_T$ and $E_1 \subseteq E_2 \subseteq \ldots E_T$. Our mining algorithm represents the data set by simply collapsing all the snapshots $G_1, \ldots, G_T$ in a graph G, in which edges are time-stamped with their first appearance. Thus, we have $G = (V, E)$ with $V = \bigcup_{t=1}^{T} V_t = V_T$ and $E = \bigcup_{t=1}^{T} E_t = E_T$. We assign a time stamp $t(e) = \min\{j \mid e \in E_j\}$ to each edge $e = \{u, v\}$. Overall, a time-evolving graph is described as $G = (V, E, t, \lambda)$, with t assigning time stamps to the edges. (The number of edge deletions in social networks is small enough to be negligible when analyzing the networks' temporal evolution. However, in our framework, we also can handle deletions by slightly changing the matching operator.[8])

Consider a time-evolving graph G. Intuitively, a pattern P of G is a subgraph of G that in addition to matching edges of G also matches node and edge labels and edge time stamps.

Now consider the pattern shown in Figure 1 extracted from the Digital Bibliography and Library Project (DBLP) coauthorship network (http://dblp.uni-trier.de), where nodes represent authors and edges between nodes represent coauthorship. Arguably, the essence of the pattern in Figure 1 is that two distinct pairs of connected authors—one collaboration created at time 0 and one at time 1—are later (at time 2) connected by a collaboration involving one author from each pair, plus a third author. We want to account for an occurrence of that event even if it was taking place at times, say, 16, 17, and 18.

These considerations lead to the following definition: Let $G = (V, E, t, \lambda)$ and $P = (V_P, E_P, t_P, \lambda_P)$ be graphs, where G is the time-evolving input graph and P is a pattern. We assume that P is connected. We say that P occurs in G if there exists a $\Delta \in \mathbb{R}$ and a function $\varphi : V_P \mapsto V$ mapping the nodes of P to the nodes in G so that the following conditions hold:

1. If (u, v) is an edge in E_P, then $(\varphi(u), (\varphi(v))$ is an edge in E and $\lambda_P(u) = \lambda(\varphi(u))$, $\lambda_P(v) = \lambda(\varphi(v))$, and $\lambda_P((u, v)) = \lambda((\varphi(u), \varphi(v)))$. In other words, an edge in the pattern graph is an edge in the host graph and all labels are matching.
2. If (u, v) is an edge in E_P, then $t(\varphi(u), \varphi(v)) = t(u, v) + \Delta$. That is, the time stamps on the edges are matched, possibly with a Δ offset.

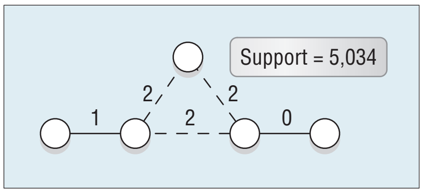

Figure 1. Relative time patterns. We extracted these patterns from the Digital Bibliography and Library Project (DBLP) coauthorship network.

With this definition, we obtain equivalence classes of structurally isomorphic patterns that differ only by a constant on the time stamp of their edges. To avoid redundancies in the search space, we only pick one representative for each equivalence class; the one where the lowest time stamp is zero.

Mining Graph Evolution Rules

Following the association-rule-mining paradigm, we next have to define concepts of support and confidence. Frequent-pattern mining algorithms are based on the notion of support. The support $\sigma(P, D)$ of a pattern P in a data set D is the number of occurrences of P in D. In the classical graph-mining setting where D is a set of graphs (such as molecules), the support is easily defined as the number of these graphs for which P is a subgraph. Various algorithms dealing with this setting exist; gSpan is one of the most efficient.[11] However, in our case, the definition of support is more complex because we must compute the support in a unique time-evolving input graph G.

The most important property of a definition of support is *antimonotonicity*, which is the basis of the Apriori algorithm. This property states

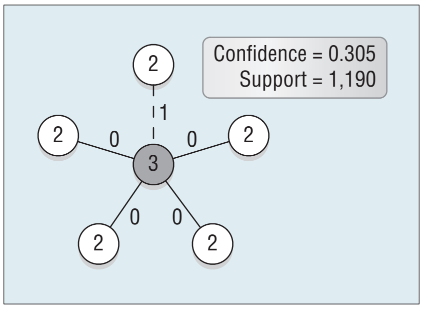

Figure 2. A graph evolution rule (GER) extracted from the Digital Bibliography and Library Project (DBLP) coauthorship network.

that a pattern's support should be at least as large as the support of all its super-patterns. The main idea is that whenever a pattern's support violates the frequency constraint, all its super-patterns can be pruned because they will also violate the minimum frequency constraint. Without antimonotonicity, frequent-pattern mining is essentially infeasible.

For our application of mining rules of evolving networks, we employ the minimum image-based support measure,[12] which is antimonotone and deals with the case of counting pattern occurrences in a single graph, as opposed to other measures that count occurrences in multiple graphs.

To characterize network evolution using frequent patterns, we need to decompose a pattern into the particular sequence of steps and subsequently determine each step's confidence. Each step can be considered as a rule body → head, with both body and head being patterns. Given a frequent pattern head P_H, the corresponding body P_B is uniquely defined as

$$E_B = \{e \in E_H \mid t(e) < \max_{e^* \in E_H}(t(e^*))\}$$

and

$$V_B = \{v \in V_H \mid \deg(v, E_B) > 0\}$$

where $\deg(v, E_B)$ denotes the degree of v with respect to the edges in E_B. Moreover, we constrain P_B to be connected. The support of a GER is the support of its head. Finally, following the association rules framework, we define a rule's confidence as the ratio of the support of the rule's head and body. (This definition yields a unique body for each head and therefore a unique confidence value for each head. This lets us represent the rules by the head only, as in Figure 1. This definition disallows disconnected graphs as body due to the lack of a support definition for disconnected graphs. As a consequence, not all frequent patterns can be decomposed into GERs.)

Figure 2 shows an example of a real GER extracted from the DBLP. The node labels represent the class of the node's degree—that is, a node labeled with a large number is a node with high degree. In this example, label 3 indicates nodes with a degree larger than 50. Additionally, edge labels capture the time information and represent the (relative) year in which the first collaboration between two authors was established. Intuitively, the rule in Figure 2 might be read as a local evidence of preferential attachment: a large-degree node (label 3), which at time t is connected to four medium-degree nodes (label 2), at time $t + 1$ will be connected to a fifth node. The collaboration-rich researcher gets richer.

We developed a tool for mining rules of graph evolution using the algorithms we've just described. Our GERM tool is an adaptation of an earlier algorithm,[12] which is in turn an adaptation of gSpan.[11] (GERM's executable code is freely available at http://www-kdd.isti.cnr.it/GERM.) Thus, GERM, like gSpan, is based on a depth-first search (DFS) traversal of the search space, which leads to low memory requirements. Indeed, in all our experiments, the memory consumption was negligible.

Figure 3 shows the basic structure of the GERM graph miner. The most involved part of the algorithm is the support computation. The starting point of our implementation is the frequent-subgraph mining framework,[12] in which the gSpan support calculation is replaced by the minimum image based support.

One of gSpan's key elements is the use of the *minimum DFS code*, which is a canonical form introduced to avoid multiple generations of the same pattern. We must change this canonical form to enable GERM to mine patterns with relative time. As we explained previously, we only want one representative pattern per equivalence class—the one with the lowest time stamp being zero. We therefore modify the canonical form such that the first edge in the canonical form is always the one with the lowest time stamp.

```
1: if $s \neq \min(s)$ then return // using our canonical form
2: $S \leftarrow S \cup s$
3: enumerate all $s$ potential children with one edge growth as $s'$
4: for all enumerated $s'$ do
5:     // considering $\Delta$ offset and our support definition
6:     if $\sigma(s', G)$ minSupp then
7:         SubgraphMining($G$, $S$, $s'$)
```

Figure 3. The basic structure of the Graph Evolution Rule Miner (GERM). *G* is the input graph, *S* is the set of frequent subgraphs mined by the algorithm, and *s* is the current subgraph, which is added to the solution set *S* if its frequency $\sigma(s, G)$ exceeds the minimum support threshold. The algorithm's initial call is *SubgraphMining*(G, $\varnothing$, ε).

Table 1. Data set statistics.

Data set	Period	$\|V\|$*	$\|E\|$*	z*	T*	CC*	Largest connected component (LCC) $\|V\|$	LCC $\|E\|$	LCC z	Growth rates Total	Growth rates Average
flickr-month	03-05	147,463	241,391	3.27	24	16,357	74,792	182,417	4.88	60,347.80	2.832960
flickr-week	02-05	149,863	246,331	3.29	76	16,661	76,058	186,504	4.90	246,331.00	0.241055
y360-month	04-05	177,278	205,412	2.32	10	17,926	110,627	155,089	2.80	68,470.70	5.150420
y360-week	04-05	177,278	205,412	2.32	41	17,926	110,627	155,089	2.80	68,470.70	0.834340
arxiv92-01	92-01	70,951	289,226	8.15	10	6,563	49,008	260,938	10.65	803.41	1.691140
dblp92-02	92-02	129,073	277,081	4.29	11	13,444	83,606	220,098	5.27	25.52	0.408188

*$|V|$ is the number of nodes, $|E|$ is the number of edges, z is the average degree, T is the number of snapshots, and CC is the number of connected components.

When matching a pattern to the host graph, we implicitly fix a value of Δ, which represents the time gap between the pattern and host graph. To complete the match, all remaining edges must adhere to this value of Δ. If all the edges match with the Δ set when matching the first edge, the pattern matches the host graph with that value of Δ.

GERM's computational complexity is similar to that of gSpan in the multiple input graphs setting, since one of the factors is the total number of nodes in both cases. However, the maximum node degree d is a second factor because it increases the possible combinations that have to be evaluated for each subgraph-isomorphism test. In detail, the worst-case complexity for computing the support of a pattern with m nodes on a database with k graphs, where n is the average number of nodes among the input graphs, equals $O(knd^m)$. In traditional applications of frequent-subgraph mining in the multiple input graphs setting, such as biology and chemistry, where the graphs typically represent molecules, n is approximately 30 to 40 nodes and thus significantly smaller than in a network with $n > 10{,}000$. On the other hand, k equals 1 in the network setting but is significantly bigger in the molecular setting ($k > 1{,}000$), leading to roughly the same number of total nodes. However, d usually reaches much higher values in a network than the molecular setting where the graphs are of small size and low degree. Therefore, although the worst-case complexity is the same as for gSpan, social network mining is harder in practice.

For this reason, we equip GERM with a user-defined parameter on the maximum number of edges in a pattern. This constraint lets us deal more efficiently with the DFS strategy by reducing the search space.[8]

Experimental Results

To evaluate GERM, we experimented on four real-world data sets:

- Flickr (www.flickr.com) is a popular photo-sharing portal. We sampled a set of Flickr users with edges representing mutual friendship and time stamp the moment when the bidirectional contact is established. We generated one set of evolving graphs with monthly granularity and one with weekly granularity.
- Y360 was an online service for blogging, sharing pictures, movies, and staying in contact with friends. We sampled a set of users and processed the data, as in the Flickr data set.
- The DBLP data set is based on a recent snapshot of the DBLP, which has a yearly time granularity. We represented authors using vertices with a connecting edge if they are coauthors. Time stamps on edges represent the year of the first co-authorship.
- arXiv (arxiv.org) is a coauthorship graph on physics publications. The graph we obtained (arxiv92-01) represents coauthorships that emerged between 1992 and 2001 with yearly time granularity.

We only report some of our experimental results here. A more comprehensive set of experiments, including experiments on runtime, scalability, and the influence of the parameters, is available in our previous work.[8]

Table 1 summarizes some basic statistics of the four data sets. For each data set, we report the number of nodes, the number of edges, the average degree (z), the number of snapshots, and the number of connected components (CC) as well as the same statistics for the largest connected component (LCC). The table shows the size of the LCC and the number of connected components in all data sets. We also report the *growth rate*, which we define as the ratio of the number of edges in two different snapshots: the total growth considering the last and the first snapshots and the average growth considering every two consecutive snapshots.

Next, we considered whether the extracted patterns carry information that characterizes the analyzed networks. We used a minimum support

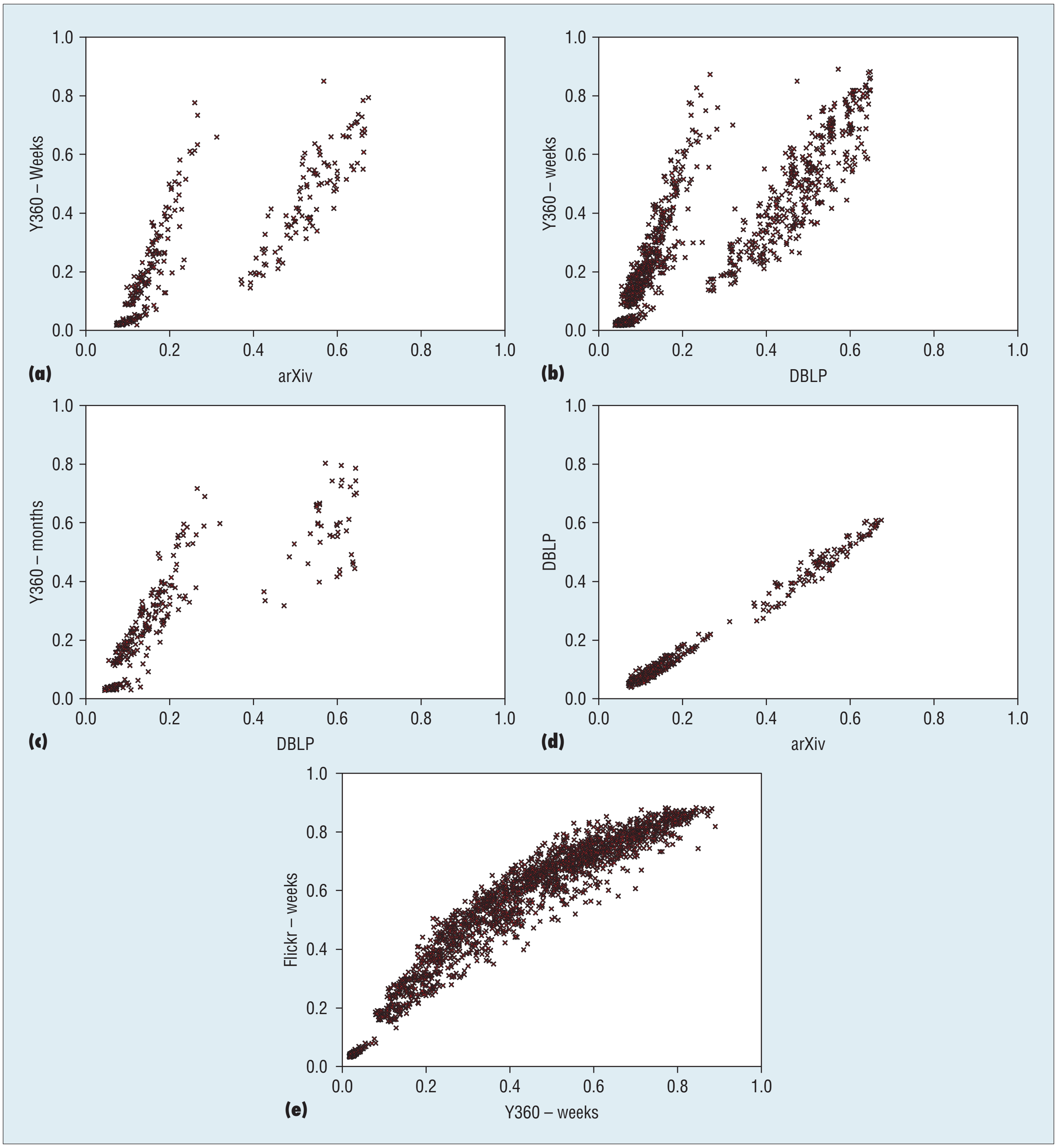

Figure 4. Comparison of graph evolution rules (GERs) confidence in different networks. (a–c) The comparison between a coauthorship network (arXiv or DBLP) and a social network (Y360) shows different confidence values of the rules for each data set. Comparing the (d) two coauthorship networks (arXiv and DBLP) or (e) two social networks (Flickr and Y360) reveals that each rule has similar confidence values in the both data sets.

threshold of 5,000 for all but the data sets involving weeks, for which we used a minimum threshold of 3,000. We compared all pairs of data sets with respect to the confidence of the rules.

Figure 4 shows the pairwise comparison results. From the plots, we can see that the comparison between a coauthorship network (arXiv or DBLP) and a social network (Y360) as in Figures 4a, 4b, and 4c show different confidence values of the rules for each data set (using Flickr instead of Y360 gives the same results).

In contrast, the comparison of two coauthorship networks (arXiv and DBLP in Figure 4d) or two social networks (Flickr and Y360 in Figure 4e) reveals that all rules are in the proximity of the bisector, meaning that each rule has similar confidence values in both data sets. We do not yet have a clear understanding of this striking behavior, but it indicates that GERs capture some aspects characterizing how different types of networks evolve.

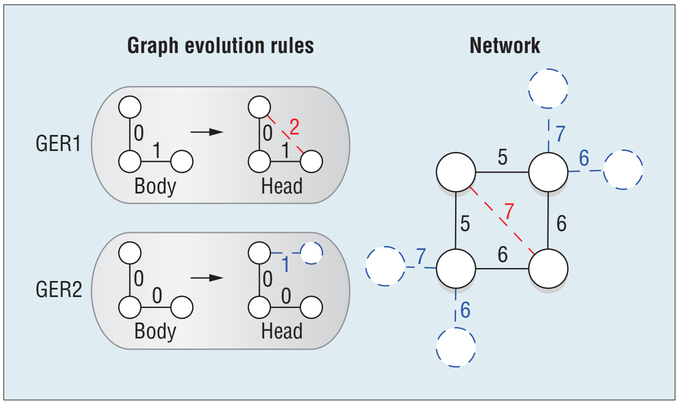

Figure 5. Edge prediction using the Graph Evolution Rule Miner (GERM) algorithm. GER1 is a graph evolution rule with two different embeddings predicting the same edge. GER2 is a rule predicting new nodes.

Predicting the Future

Given a pair of nodes in an evolving network, link prediction is the problem of estimating the likelihood that an edge will eventually be formed between these two nodes. In the context of a bibliographic collaboration network, link prediction estimates the likelihood that two authors will collaborate in the future. In the context of a social network such as Flickr or Facebook, it estimates the likelihood that two users will become friends.

David Liben-Nowell and Jon Kleinberg defined the *link prediction problem* as follows: given a snapshot of a social network at time t and a future time t', we can predict the new social ties that are likely to appear in the network within the time interval $[t, t']$.[9] In their framework, Liben-Nowell and Kleinberg only considered features based on the network's link structure. For instance, they considered a predictor to be the number of common neighbors $CN(i, j)$ of the two nodes i and j at time t. Another predictor that performed well is the Adamic-Adar measure $AA(i, j)$, which is the sum of $\log(d(z))^{-1}$ over all common neighbors z of i and j. (Liben-Nowell and Kleinberg proposed other features based on the ensemble of all paths between i and j. However, even considering those more complex features, Adamic-Adar is among the top-performing predictors in all the networks studied.[9])

We can use the rules extracted using the GERM algorithm to predict edges in future instances of the network. We start with an input network N and a set of GERs $\mathfrak{R}$ extracted from that network. For the sake of simplicity, we do not use all rules extracted, only the rules that have only one edge (v_1, v_2) of difference between the head and the body. Our approach decomposes the prediction problem into two steps. The first step identifies all the embeddings of the body of the rule into the network. Each embedding of a rule results in exactly one candidate for prediction $p_c = (v_1, v_2)$ between nodes v_1 and v_2. We consider those embeddings only as candidate predictions, and we devise a set of scoring functions that let us make a more elaborate decision about those candidate predictions. Each of these prediction candidates p_c might result from different rules or different embeddings of the same rule. Figure 5 shows an example where the rule GER1 predicts the same edge (dashed red) twice. In fact, GER1 can be matched twice in the network, once to the upper right triangle and once to the lower left triangle, in either case with $\Delta = 5$.

As a result, we know for each p_c, the rule R gives rise to the prediction associated with the respective count $\mathrm{count}(p_c, R)$—that is, the number of times a rule can be matched—the support of the rule $\sup(R)$, and its confidence $\mathrm{conf}(R)$. Based on this information, we define four different scores $\mathrm{score}(p_c, R)$ for each candidate p_c and rule R:

- $\mathrm{score}_1(p_c, R) = \mathrm{conf}(R)$
- $\mathrm{score}_2(p_c, R) = \sup(R) * \mathrm{conf}(R)$
- $\mathrm{score}_3(p_c, R) = \mathrm{conf}(R) * \mathrm{count}(p_c, R)$
- $\mathrm{score}_4(p_c, R) = \sup(R) * \mathrm{conf}(R) * \mathrm{count}(p_c, R)$

To obtain one score for each prediction candidate, we need to accumulate the scores for a particular prediction candidate given by all the rules. We decided to take the two simplest ways of accumulation: the maximum and the sum. Additionally, we explored a small variant that ensures that we only account for the score of the most specific rules. We call $\mathfrak{R}_{p_c}$ the set of all rules that yield a candidate prediction p_c, and $\mathfrak{R}^*_{p_c}$ the subset of $\mathfrak{R}_{p_c}$ such that there are no two rules R_1, R_2 in $\mathfrak{R}^*_{p_c}$ with $R_1 \sqsubseteq R_2$. This gives us four possible ways to accumulate:

- $\mathrm{pred}_{1,i}(p_c) = \sum_{R \in \mathfrak{R}_{p_c}} (\mathrm{score}_i(p_c, R))$
- $\mathrm{pred}_{2,i}(p_c) = \max_{R \in \mathfrak{R}_{p_c}} (\mathrm{score}_i(p_c, R))$

Table 2. Statistics on the new edges appearing in a target period.

Data set	Target period	Old-old edges	Old-new edges	New-new edges
dblp92-02	2003	12,121 (18.0%)	29,212 (43.4%)	25,998 (38.6%)
flickr-month	April 2005	34,373 (37.2%)	35,545 (38.5%)	22,512 (24.3%)
flickr-week	6th week, 2005	4,365 (41.8%)	3,885 (37.3%)	2,181 (20.9%)

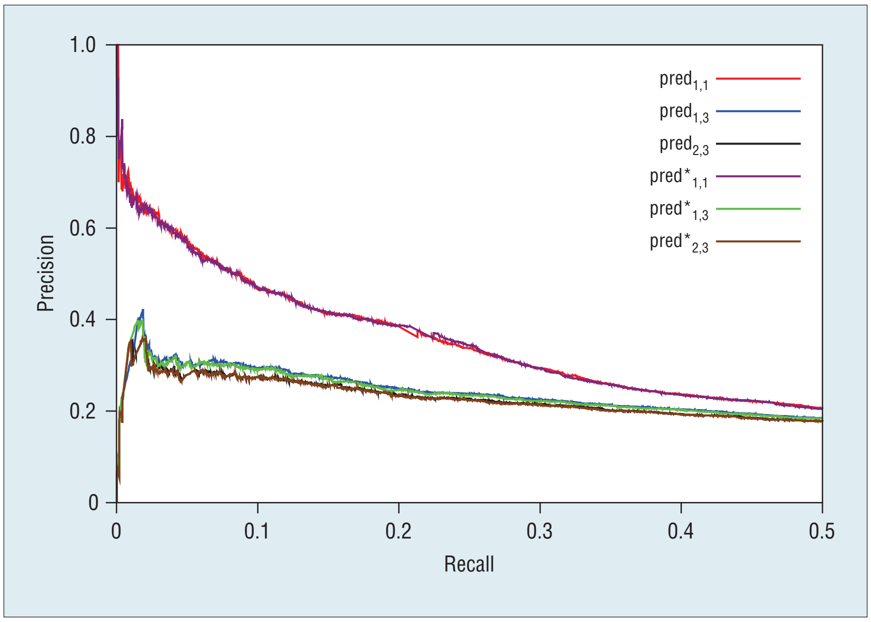

Figure 6. Plot of precision versus recall on the Digital Bibliography and Library Project (DBLP) for the task of predicting old-new edges. Our scores are derived from graph evolution rules (GERs) computed with minSupp = 5,000, minConf = 0, and at most five edges.

- $\text{pred}^*_{1,i}(p_c) = \sum_{R \in \mathfrak{R}^*_{p_c}} (\text{score}_i(p_c, R))$
- $\text{pred}^*_{2,i}(p_c) = \max_{R \in \mathfrak{R}^*_{p_c}} (\text{score}_i(p_c, R))$

Combining these with the four first scores gives us 16 different possible predictive features that we evaluated in our experiments. Since some of the features exhibit similar behavior, we only report on a subset of them in our plots.

Before discussing the details of our experiments, we need to point out an important difference. The task in the classic link-prediction setting is to predict edges appearing in the interval $[t, t']$ that have both incident nodes already part of the graph at time t. In addition, our GERs allow for predicting edges from existing nodes to new nodes that are not yet part of the graph at time t.

Figure 5 depicts an example GER that might be used for predicting a new edge connected to a new node (GER2). The same rule might also match the additional node in the head of the rule to an already existing node in the network, thus helping predict new edges among existing nodes as in the standard Liben-Nowell and Kleinberg framework.

This is an important feature of our framework because many new nodes join the network over time. Table 2 reports some statistics on the target prediction period that we adopted for our experiments on three different data sets. In particular, the table reports the numbers of edges between two old nodes (that is, old-old nodes that existed in the training period), one old and one new node (old-new), and two new nodes (new-new). The old-old category includes those being handled by classical link prediction (and also by our framework), the old-new category are those cases that our framework can handle but classical link prediction cannot. Neither our framework nor the classic link-prediction framework can handle the new-new category. As Table 2 shows, the old-new category represents a large fraction of the newly created edges for all data sets.

Figure 6 reports precision versus recall curves on various scores of our framework for the task of predicting only old-new edges.

Next we report our experimental evaluation in the classic link-prediction setting (predicting only old-old edges). Figures 7 and 8 report the results for flickr-week and flickr-month data sets, respectively. Both figures show precision-recall plots for various methods. In particular, we report *CN* and *AA* as baselines, one of our scores—namely, $\text{pred}_{1,3}$, plus five combinations of our score with *AA*. The combination of our score $\text{pred}_{x,y}$ with *AA* for an edge *e* is obtained by taking $(\text{pred}_{x,y}(e) + 1) \times (AA(e) + 1)$, where *AA* and $\text{pred}_{x,y}$ are normalized before combining.

By comparing *AA* with our "pure" $\text{pred}_{1,3}$ score, we can see that our method achieves a much higher precision for low recall levels, but it performs worse than *AA* at higher recall levels. This observation suggests combining *AA* with our methods. Figures 7 and 8 confirm that the combination achieves the best of the two methods, outperforming each of the methods combined in the majority of cases.

From this preliminary analysis, we can conclude that our method $\text{pred}_{1,3}$ is the most robust and stable among our predictors, and it should be combined with *AA* for an effective link prediction.

Our framework, based on the extraction of local frequent patterns from the past evolution, represents a very different approach from the classic link-prediction method, which adopts features based on the network's link structure at time *t* to predict if an edge will appear in the future interval [*t*, *t*′]. The two features common neighbors and Adamic-Adar can essentially be considered rules that predict closing triangles, such as GER1 in Figure 5. However, even if we consider only the closing-triangle rule in Figure 5 (and not the ensemble of all the extracted rules as our method prescribes), we incorporate some additional information about how the network evolved in the past in two ways. First, the time stamps on the edges allow for a matching sensitive to the current evolution instead of viewing the network without its evolutionary growth. Further, the confidence describes how likely such an evolution was in the past. This information is completely discarded in classic link prediction, which only uses the network's structure. Implicit in our work is the assumption that a model's predictive power can be strengthened by learning from the entire evolution history and not just using information presently available.

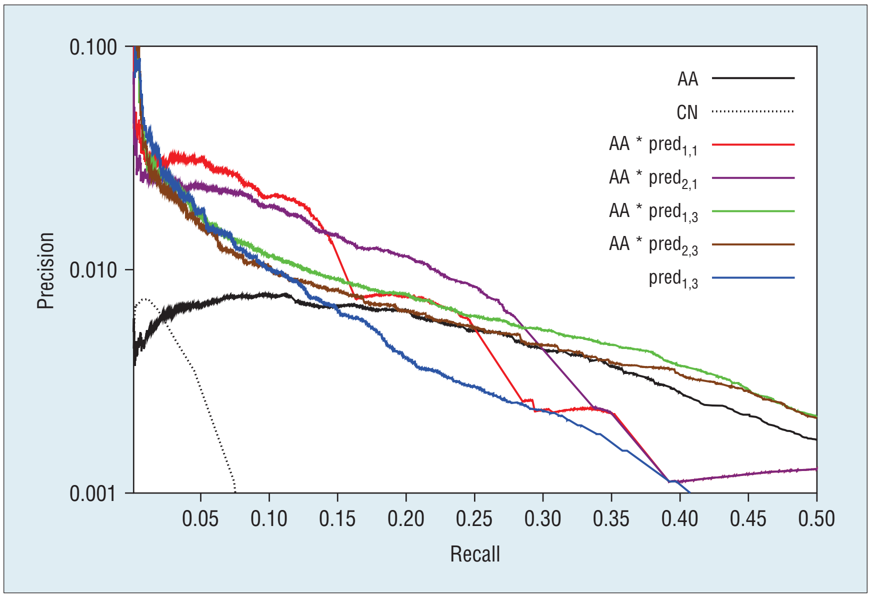

Figure 7. Plot of precision versus recall on flickr-week showing a selection of our scores combined with Adamic-Adar compared with the baselines. Our scores are derived from graph evolution rules (GERs) computed with minSupp = 3,000, minConf = 0, and at most six edges.

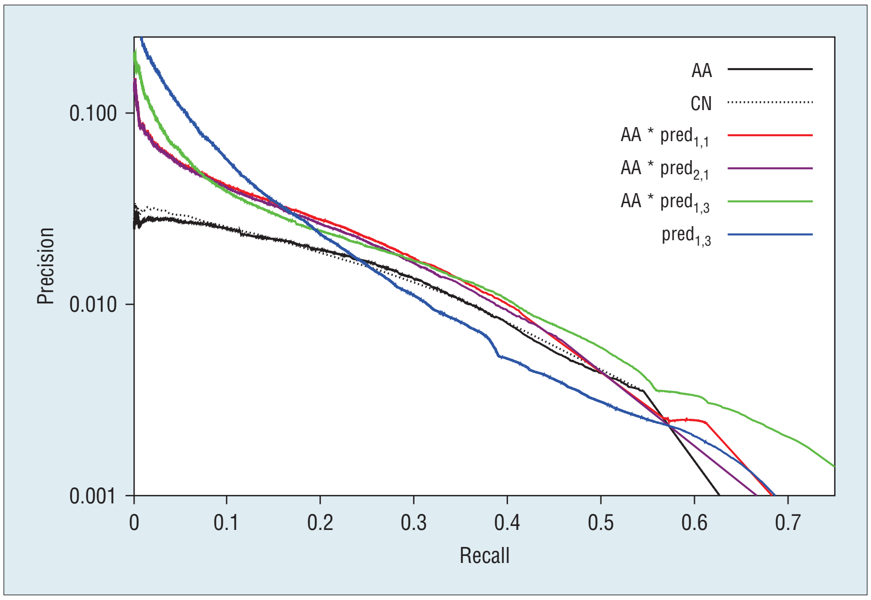

Figure 8. Plot of precision versus recall on flickr-month showing a selection of our scores combined with Adamic-Adar compared with the baselines. Our scores are derived from graph evolution rules (GERs) computed with minSupp = 5,000, minConf = 0, and at most five edges.

Acknowledgments

Björn Bringmann's research is supported by the European Commission under the Seventh Framework Program FP7-ICT-2007-C FET-Open, under contract number BISON-211898.

References

1. F.-Y. Wang et al., "Social Computing: From Social Informatics to Social Intelligence," *IEEE Intelligent Systems*, vol. 22, no. 2, 2007, pp. 79–83.
2. M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On Power-Law Relationships of the Internet Topology," *ACM SIGCOMM Computer Comm. Rev.*, vol. 29, no. 4, 1999, pp. 251–262.
3. D.J. Watts and S.H. Strogatz, "Collective Dynamics of 'Small World' Networks," *Nature*, vol. 393, 1998, pp. 440–442.

THE AUTHORS

Björn Bringmann is a postdoctoral researcher at the Katholieke Universiteit Leuven. His research interests include data mining from molecular data to complex social and biological networks. Bringmann has a PhD in computer science from the Katholieke Universiteit Leuven. Contact him at bjorn.bringmann@cs.kuleuven.be.

Michele Berlingerio is a postdoctoral researcher at the KDDLab of Institute of Information Science and Technologies, Italian National Research Council (ISTI-CNR). His research interests include graph mining, social network analysis, and workflow mining. Berlingerio has a PhD in computer science and engineering from IMT Lucca. Contact him at michele.berlingerio@isti.cnr.it.

Francesco Bonchi is a senior research scientist at Yahoo Research, Barcelona, Spain. His research interests include mining query-logs, social networks, and social media. Bonchi has a PhD in computer science from the University of Pisa. Contact him at bonchi@yahoo-inc.com.

Aristides Gionis is a senior research scientist in Yahoo Research, Barcelona, Spain. His research interests include algorithms for data analysis and applications in the Web domain. Gionis has a PhD in computer science from Stanford University. Contact him at gionis@yahoo-inc.com.

4. M. Girvan and M.E.J. Newman, "Community Structure in Social and Biological Network," *Proc. Nat'l Academy of the Sciences* (PNAS), vol. 99, no. 12, 2002, pp. 7821–7826.
5. R. Albert and A.L. Barabasi, "Emergence of Scaling in Random Networks," *Science*, vol. 286, 1999, pp. 509–512.
6. J. Leskovec, J.M. Kleinberg, and C. Faloutsos, "Graphs Over Time: Densification Laws, Shrinking Diameters and Possible Explanations," *Proc. 11th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining* (KDD), ACM Press, 2005, pp. 177–187.
7. J. Leskovec et al., "Microscopic Evolution of Social Networks," *Proc. 14th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining* (KDD), ACM Press, 2008, pp. 462–470.
8. M. Berlingerio et al., "Mining Graph Evolution Rules," *Proc. European Conf. Machine Learning and Knowledge Discovery in Databases* (ECML PKDD), LNCS 5781, Springer, 2009, pp. 115–130.
9. D. Liben-Nowell and J.M. Kleinberg, "The Link Prediction Problem for Social Networks," *Proc. ACM Int'l Conf. Information and Knowledge Management* (CIKM 03), ACM Press, 2003, pp. 556–559.
10. L. Tang and H. Liu. "Toward Predicting Collective Behavior via Social Dimension Extraction," *IEEE Intelligent Systems*, vol. 25, no. 4, 2010, pp. 19–25.
11. X. Yan and J. Han, "g-Span: Graph-Based Substructure Pattern Mining," *Proc. Int'l Conf. Data Mining* (ICDM), IEEE CS Press, 2002, pp. 721–724.
12. B. Bringmann and S. Nijssen, "What Is Frequent in a Single Graph?" *Proc. 12th Pacific-Asia Conf. Advances in Knowledge Discovery and Data Mining*, (PAKDD 2008), ACM Press, 2008, pp. 858–863.

RUNNING IN CIRCLES LOOKING FOR A GREAT COMPUTER JOB OR HIRE?

Make the Connection - IEEE Computer Society Jobs is the best niche employment source for computer science and engineering jobs, with hundreds of jobs viewed by thousands of the finest scientists each month - **in *Computer* magazine and/or online!**

- > Software Engineer
- > Member of Technical Staff
- > Computer Scientist
- > Dean/Professor/Instructor
- > Postdoctoral Researcher
- > Design Engineer
- > Consultant

IEEE computer society | JOBS

http://www.computer.org/jobs

IEEE Computer Society Jobs is part of the *Physics Today* Career Network, a niche job board network for the physical sciences and engineering disciplines. Jobs and resumes are shared with four partner job boards - *Physics Today* Jobs and the American Association of Physics Teachers (AAPT), American Physical Society (APS), and AVS: Science and Technology of Materials, Interfaces and Processing Career Centers.