Structure Prediction in Temporal Networks using Frequent Subgraphs

Mayank Lahiri
Department of Computer Science
University of Illinois at Chicago
Chicago, IL 60607
Email: mlahiri@cs.uic.edu

Tanya Y. Berger-Wolf
Department of Computer Science
University of Illinois at Chicago
Chicago, IL 60607
Email: tanyabw@cs.uic.edu

***Abstract*— There are several types of processes which can be modeled explicitly by recording the interactions between a set of actors over time. In such applications, a common objective is, given a series of observations, to predict exactly when certain interactions will occur in the future. We present a representation for this type of temporal data and a generic, adaptive algorithm to predict the pattern of interactions at any arbitrary point in the future. We test our algorithm on predicting patterns in e-mail logs, correlations between stock closing prices, and social grouping in herds of Plains zebra. Our algorithm averages over 85% accuracy in predicting a set of interactions at any unseen timestep. To the best of our knowledge, this is the first algorithm that predicts interactions at the finest possible time grain.**

I. Introduction

In many applications, a common objective is to model the dynamic behavior of a process and to predict what it will look like in the future. Several types of processes are represented by a set of actors interacting over a period of time. Examples which have recently been studied are the evolution of bibliographic databases [1]–[5], the web and other information networks [6]–[14], disease transmission paths in epidemiological simulations [15]–[21], influence and information spread in a social network [22]–[30] and the graph of stock price correlations above a minimum threshold [31], [32]. Temporal networks[1] are a powerful generic model for such processes [35]. A temporal network consists of a sequence of regular graphs, each being a snapshot of interactions at a particular instant or over a small time interval. Vertices in each graph represent actors and edges between them represent interactions, which can either be directed or undirected (bi-directional). The flexibility of the definition allows temporal networks to be used to model a variety of processes while maintaining the explicit order and concurrency of interactions.

There are many questions that can be posed for processes represented as temporal networks. We focus on the task of predicting the structure of the temporal network at *each timestep*, thereby computing a model of the evolution of the process under consideration. There is an extensive body of work that focuses on problems related to the evolution of networks, without the information on the order and concurrency of interactions. Such networks can be seen as collapsed aggregations of temporal networks. A closely related problem from such network analysis is that of *link prediction* in a graph, which aims to rank all possible edges (interactions) by the likelihood that they will occur in the future [4]. Our work extends that definition to temporal networks by predicting the structure of the graph which is not noise at each timestep. The predictions are based solely on prior observations. Note, that unlike the link prediction problem, we are concerned with the ability to predict exactly *when* groups of interactions will occur, not to predict the likelihood of every possible interaction occurring in the future, regardless of the timing and order of these occurrences.

In this paper, we present a generic, accurate, adaptive, streaming algorithm for efficient structure prediction in temporal networks. Our algorithm does not rely on any domain-specific parameters. The data is assumed to be a stream of graphs, and the algorithm adaptively learns a model for the evolution of the process. Our algorithm uses the idea that probability density functions for the time interval between every pair of interactions can be used to make predictions about subsequent timesteps. However, directly using this approach requires computing on pairs of edges and becomes intractable for even medium-sized graphs. We propose the use of frequent subgraphs in order to aid the tractability of the algorithm as well as to filter out insignificant interactions. We tested our algorithm on three diverse examples of real-world processes, ranging from stock price correlations to dynamics in animal populations. Our algorithm averages 85.7% accuracy when predictions are allowed a single slack timestep. This is the first algorithm, to the best of our knowledge, that predicts interactions between actors at the finest possible time grain.

This paper is organized as follows. In the next section, we review relevant prior work from computer science as well as other fields. In the following section, we list some preliminary definitions and show an equivalence between temporal networks and the traditional transactional, or 'market basket', data that is common in data mining research. We then formally describe the problem and our algorithm in Section II. In Section III, we describe the datasets used and present experimental validation results. This is followed by a discussion, and then conclusions and future work in Section IV.

[1]Similar representations are also known as dynamic networks [33], time-series networks, and longitudinal data in social network analysis [34].

A. Related Work

From a theoretical point of view on the evolution of networks, there have been several papers that characterize the evolution of large, complex networks such as bibliographic databases [1], [36], [37] in terms of, among other measures, changes in vertex degree distributions over time. Behavior such as preferential attachment [37] and structural properties like the 'small-world' effect [1] have been observed in several real-world networks. Similarly, Snijders [34] proposes a sociological behavior model for actors, coupled with Markov Chains in order to build models of temporal dynamics in social networks. The models are evaluated using statistical measures, but not empirically.

Most work in explicit link prediction has used static graphs which aggregate temporal data in some way. The survey by Getoor and Diehl [38] outlines some approaches to link mining in general, and link prediction in particular. Similarly, Desikan and Srivastava [7] outline a different three-tier approach to mining temporal networks with an emphasis on temporal networks generated from web usage logs. Liben-Nowell and Kleinberg [4] test the accuracy of several structural measures from social network analysis and web mining on ranking potential future collaborations in a bibliographic database. Although the relatively simple common neighbors indicator was found to be about as effective as more advanced techniques, the accuracy of link prediction on the aggregated test interval was still quite low.

Given a single graph with a partially known link structure, Popescul and Ungar [39] try to infer which edges are missing from the graph. Test data is generated by removing links from a bibliographic database, following which effective features for classification are extracted by searching over the space of database queries. This is similar to the problem of graph inference, where a partial structure of a single graph is given and the remaining structure has to be inferred based on topological features alone. Vert and Yamanishi [40] describe a supervised learning approach to this problem. Note that this is a different line of research that aims to reconstruct partially specified biological networks in biologically meaningful ways.

There have been few papers that explicitly use temporal data. Kempe, Kleinberg and Kumar [41] define temporal networks as static graphs where every edge is labeled with the time that the interaction took place. They define the inference problem on temporal networks as that of reconstructing time labels for unlabeled edges, given a partial labeling of the graph. O'Madadhain, Hutchins and Smyth [42] outline a method to perform explicit temporal prediction of interactions. Domain-dependent features are extracted from the temporal data, following which a probabilistic classifier is used to predict future interactions. However, their focus is on predicting future interactions which have not occurred before, whereas our method predicts *when* frequent groups of interactions will occur at the finest possible time grain, given that they have occurred at some point in the past.

Finally, our method also relies on results from the field of frequent pattern mining. Recent advances in frequent graph and itemset mining have resulted in efficient algorithms for mining both maximal and closed frequent subgraphs from a database of graphs [31], [43]–[45], as well as specialized frequent structures like cliques [31]. Efficient algorithms have also been proposed for mining frequent closed itemsets [46], [47] from large datasets, which are used in our algorithm.

B. Definitions

In this section, we formally define temporal networks and frequent subgraphs. We assume that a temporal network is being used to model interactions amongst a set of uniquely defined actors.

Definition 1: A *temporal network* is a sequence of graphs $G = \langle G_1, ..., G_T \rangle$, where $G_t = (V_t, E_t)$ is the unweighted graph of pairwise interactions at time $t \in [1, T]$. We let $V = \bigcup_t V_t$ be the universal set of actors.

Note that not all actors have to be present in all timesteps, thus $\forall t,\ V_t \subseteq V$. Each actor $v \in V$ is uniquely labeled, and each v can appear only once at each timestep. Furthermore, for any $v \in V_t$, if v is not connected to any other vertex, then the self-edge $(v, v) \in E_t$ is implied.

Definition 2: For any graph $G' = (V', E')$ such that $V' \subseteq V$, we define the *support set* of G' as $S(G') = \{t \mid G' \subseteq G_t\}$, that is, the set of timesteps for which G' is a subgraph of G_t at that timestep. The cardinality of the support set, $|S(G')|$, is called the *support* of G' in G.

Definition 3: A graph G' is *frequent* in G if $|S(G')| \geq minsup \times T$, where $0 < minsup \leq 1$ is a fixed minimum support threshold.

Since the vertices in each graph G_t are uniquely labeled and appear only once per graph, determining whether a subgraph is contained in another graph does not involve graph isomorphism. This is in contrast to the majority of research in graph mining, where there is no restriction on the vertex labels of a graph and, consequently, the need to check for subgraph isomorphism.

Definition 4: Let $\mathcal{F}$ be the set of all frequent subgraphs of G given a particular $minsup$. A graph $F \in \mathcal{F}$ is *maximal* if $\forall H \in \mathcal{F}\ F$ is not a subgraph of H. A graph $F \in \mathcal{F}$ is *closed* if $\forall H \in \mathcal{F}$, such that $H \neq F$, either F is not a subgraph of H or $S(F) \neq S(H)$.

C. Mapping between Itemsets and Temporal Networks

Temporal networks with the constraints that we have imposed have an equivalence with the more traditional 'market basket' type of data used in frequent itemset mining [48]. Market basket data consists of a set of transactions where each transaction is a set of items drawn from a universal set. In order to transform a temporal network into a transactional database, for each graph $G_t \in G$, each edge $e \in E_t$ can be uniquely represented as a concatenation of the labels of its vertices. Singleton vertices are represented by self-edges and will be included in frequent subgraphs. With any mapping from this concatenation to the set of integers, each $e \in E_t$ becomes an

'item' and each timestep becomes a transaction of items. In this way, tools for mining frequent itemsets can be used to mine frequent subgraphs. Itemset mining does not incur the cost of checking for subgraph isomorphism that is required to varying degrees in general graph mining algorithms.

II. Prediction in Temporal Networks

We now formally define the structure prediction problem for temporal networks and state our prediction algorithm.

A. Problem Statement

We define the structure prediction problem for temporal networks as follows: given a temporal network $G = \langle G_1, ..., G_T \rangle$, predict a set of edges (interactions) for *each* $T' > T$. One possible approach is to directly estimate the probability density function for the number of timesteps between an occurrence of an edge i followed by an edge j, for all i and j. However, this approach would have an inherently $O(|V|^4 T)$ complexity in the number of vertices and timesteps, which is intractable even for relatively small graphs with $|V| = 100$ and $T = 100$. Moreover, a blind density computation over the whole space of edge pairs would include ones which have no real correlation.

We propose a modified version of this approach using frequent subgraphs instead of individual edges. There are two reasons for using this approach. The first being that to reduce the number of edge pairs for which probability density functions have to be calculated, we aim to retain statistically significant groups of interactions. Frequent subgraphs can be used as heuristic approximations of groups of statistically significant concurrent interactions, and algorithms for their extraction are specifically designed to operate efficiently on large databases. The second reason for using frequent subgraphs, and closed frequent subgraphs in particular, is that the number of closed frequent subgraphs is typically asymptotically smaller than the number of edges in a graph. The complexity of the density estimation using frequent subgraphs instead of edge pairs then becomes $O(F^2 T)$, where F is the number of frequent subgraphs. Our problem can now be redefined as one of predicting frequent subgraphs at each timestep.

B. The Algorithm

We now describe our method for the problem of predicting frequent subgraphs at each timestep. Our algorithm consists of two main components: a probability density estimator for pairs of frequent subgraphs and an adaptive prediction module.

1) Extract frequent closed subgraphs (using frequent closed itemsets algorithm): Given a training temporal network, the algorithm first extracts frequent closed subgraphs at a reasonable value of $minsup$. We use the itemsets correspondence and a frequent closed itemsets algorithm to achieve this goal. Frequent closed subgraphs are one of three general approaches to mining frequent subgraphs. A different approach is to extract all frequent subgraphs. According to the downward closure property, every subgraph of a frequent subgraph is also frequent. This often leads to a massive number of frequent subgraphs, limiting the utility of using frequent subgraphs instead of edge pairs. An alternative approach mines only maximal frequent subgraphs. This approach can be overly restrictive because of the emphasis on maximizing the size of subgraphs while excluding smaller subgraphs which might be parts of larger frequent subgraphs, but also exist independently of the larger subgraph at some timestep. A compromise between these two approaches is to use frequent closed subgraphs, which are far fewer than the total number of frequent subgraphs, but not as structurally biased as maximal frequent subgraphs.

2) Learn the probability density function for each frequent subgraph pair and create a prediction estimator for each pair: The algorithm progressively learns probability density functions $f_{ij}(x)$ for each pair of frequent subgraphs (F_i, F_j), representing the time interval to an occurrence of F_j after an occurrence of F_i. That is, $f_{ij}(x)$ is the relative frequency of F_j occurring x timesteps after F_i. Note that the pair (F_i, F_i) is permissible and is equivalent to the probability density function for the distribution across timesteps of a particular frequent subgraph F_i. An estimator, designated Φ, uses the density function approximations to estimate a likely value for each time interval (F_i, F_j). We found that using the highest point estimate of a probability density function as Φ was effective. For each F_i observed at the current timestep t, the pairs (F_i, F_j) are scanned for all j and a prediction is made for F_j at timestep $t + \Phi(F_i, F_j)$.

3) Assign reliability values to each prediction estimator: In order to suppress predictions which are likely not to be reliable, the algorithm dynamically assigns a reliability to different values of $\Phi(F_i, F_j)$. It does so by using positive and negative examples created from the data. A positive example for $\Phi(F_i, F_j) = t$ is defined as F_j being observed in the data stream at an interval of $t > 0$ timesteps from the last occurrence of F_i. A negative example for $\Phi(F_i, F_j) = t$ is defined as an incorrect prediction being made using $\Phi(F_i, F_j) = t$. In that case, all combinations of F_i, F_j and t used to make the prediction are punished. Note, that a positive example is defined as a particular interval, not a prediction, being correctly observed in the data stream. A particular hypothesis $\Phi(F_i, F_j) = t$ is used to make a prediction only when the proportion of positive examples exceeds a pre-defined threshold τ. Although in principle any classification technique can be used for this purpose, we found that using a simple ratio seems to be quite effective and allows noisy Φ functions to fall out of use because of incorrect predictions until they are supported enough by the data to be used again.

The algorithm takes advantage of the fact that temporal network data is inherently labeled. We can assume that vertices of graphs in the test data are labeled as they are in the training data [2]. As a result, streaming algorithms for temporal networks can, in principle, continually evaluate their own performance and adapt to changing processes. Predictions for new data are confirmed before the algorithm is allowed to update its model

[2]Note that a different line of research aims to label graph vertices given a partial labeling [40].

using these data. Although this blurs the distinction between training and testing data, it allows for agile model estimation. We use training data to extract frequent subgraphs and learn initial reliabilities for every density function. However, our algorithm can also run without continual online learning. This, in effect, predicts graph structure using a static model of the evolution of the process learned from the training data alone. We present results for both methods in the next section.

III. Experimental Setup

In this section, we describe the implementation of our algorithm and the datasets used to test its performance and accuracy. Experimental results are then presented, followed by a discussion of the results.

A. Datasets

We use four large, real-world temporal networks to test the accuracy of our algorithm. The datasets range from networks that are relatively sparse but with a large number of vertices and timesteps, to more dense networks with fewer vertices and timesteps. Each dataset is described below and Table I lists statistics for each.

1) Enron e-mails: The Enron e-mail corpus is a publicly-available database of e-mails sent by and to employees at the now defunct Enron Corporation [49]. We used a slightly cleaner version with fewer integrity issues [50]. For each day of the e-mail activity, message headers were extracted based on their timestamps. Actors are identified by their e-mail address, and an interaction occurs if an e-mail was sent between two addresses on a given day. The timestep resolution is a single day. Only one edge per day is incurred even if multiple emails are sent between two actors. The direction of the email is discarded in our experiments.

2) Plains zebras: Social interactions between Plains zebra (*Equus Burchelli*) in Kenya were recorded by direct observations made by behavioral ecologists from Princeton [51]. The data is from visual scans of the populations typically once a day over periods of several months. Each spatially proximate group of animals represents a complete set of interactions amongst those individuals. Individuals in each group are identified. The GPS coordinates of each group are noted.

3) Stock market: Correlations between closing prices of stocks over a sampling period can be represented as a temporal network. Using publicly-available historical data for the New York Stock Exchange (NYSE) ranging from 1988 to 2006 [52], we obtained daily closing prices for 3,416 stocks. For every pair of stocks (S_1, S_2), the correlation coefficient ρ is defined as $\rho = \dfrac{cov(S_1, S_2)}{\sigma_{S_1} \times \sigma_{S_2}}$, where $cov(S_1, S_2)$ is the covariance of the daily closing prices of S_1 and S_2 over a sampling period. For every successive sampling period, an edge (S_1, S_2) exists if $\rho(S_1, S_2) \geq \alpha$, where α is a minimum correlation value. This form of the stock graph has been shown to have the scale-free property common to many real-world social networks [32], as well as frequently occurring patterns [31]. We use two versions of the stock graph, one with a sampling period of 30 days and $\alpha = 0.9$ (Stock Market-1) and another with a sampling period of 5 days and $\alpha = 0.98$ (Stock Market-2). Furthermore, starting from the most recent prices, we scan backwards and discard stocks that cease to be traded. When the number of remaining stocks reaches a fixed number, we terminate and keep only those stocks. In this manner, we obtain properly aligned stock time series of identical length. The smaller that fixed minimum number of stocks, the longer the resulting time series will be. The fixed minimum number of stocks chosen are 1,000 (Stock Market-1) and 2,500 (Stock Market-2).

TABLE I
Test dataset statistics

Dataset	Timesteps	Vertices	Total edges	Unique edges
Enron	1,213	75,026	936,032	295,233
Plains Zebras	228	1,002	435,926	168,787
Stock Market - 1	104	1,000	1,415,815	501,499
Stock Market - 2	121	2,500	4,910,962	2,482,911

B. Experiment Parameters and Results

The algorithm was implemented in C and run on a dual-core Pentium D system running at 3.20GHz with 4GB of RAM and Linux kernel 2.6.12. Each dataset was converted to a transactional itemset representation using the transformation described in Section I-C. In each dataset, the first one-fifth $(\lceil \frac{T}{5} \rceil)$ of timesteps was designated the training network. Although the algorithm makes and tests predictions even when processing the training network, these predictions are not counted toward the final accuracy.

Once training and test networks have been obtained, frequent closed itemsets are extracted from the training network using the publicly-available open-source implementation of the MAFIA algorithm [47]. Depending on the network, $minsup$ is gradually decreased from an initial value of 1 until a reasonable number of frequent closed itemsets are found. We found that a range of between 10 and 2,000 frequent closed itemsets is quite effective, with the difference being in the total number of predictions made.

After mining frequent closed itemsets, our prediction algorithm is run on the testing network once for each set of frequent itemsets found at the different $minsup$ levels. A slack interval S is used for testing predictions. A prediction for a frequent subgraph F_i at time t is considered correct if F_i is observed anywhere in the time interval $[t-S, t+S]$. At $S = 0$, all predictions have to be exactly accurate in each timestep. The minimum reliability parameter τ was set to 0.9 for all runs and learning from test data after checking hypotheses was enabled. The slack parameter for testing predictions was varied in the range $[0, 2]$.

Fig. 1 shows the accuracy of the algorithm on each dataset for different values of $minsup$ and slack. Each run of the

TABLE II
AVERAGE ACCURACY OVER ALL DATASETS.

Slack	Accuracy	
	Online learning	Static model
0	34.4%	36.2%
1	85.7%	76.3%
2	90.3%	78.9%

algorithm took less than 5 minutes. When online learning is enabled, streaming test data is used for continual learning once hypotheses for new data have been tested. The alternative is a static learned model, where only the training network is used to learn predictor reliability. Table II lists the average accuracy over all datasets at the highest mining support for each dataset. Note that the higher the $minsup$, the fewer the number of frequent subgraphs, and therefore the fewer the total number of predictions made by the algorithm.

C. Discussion

The most prominent result of our experiments is that the proposed network structure prediction algorithm is highly accurate (with one slack time step). An important observation is that predicting structure precisely at each timestep is a difficult task. Moreover, it is unlikely that real-world processes, coupled with the error associated with any observation process and an arbitrary quantization into timesteps, would have such perfectly predictable patterns of interaction. This is emphasized by the dramatic improvement in accuracy when a single slack timestep is allowed for structure predictions, and marginally thereafter for larger slack intervals. Although the noise associated with precise predictions holds for all our datasets, predictions made for a particular timestep, give or take a timestep, allow us to model, with a high level of confidence, when interactions will occur in a diverse range of domains.

The use of frequent subgraphs is only one possible method to reduce the space of entities being considered for temporal dependence. We found that a higher mining support, which usually results in fewer frequent subgraphs, generally leads to improved performance. This agrees with the intuitive notion that frequent patterns which occur more frequently than others are likely to be more predictable. The tradeoff is that fewer frequent subgraphs are likely to cover a smaller subset of individuals. Noting that our algorithm predicts the occurrence of frequent subgraphs, this leads to predictions covering fewer individuals overall. A simple way to relax this restriction is to maintain a buffer of the most recent timesteps and continually extract local, as opposed to global, frequent subgraphs. This, and other approaches, will be addressed in future work.

Our algorithm is designed to operate either in online mode, where the model is continually updated once new timesteps are revealed and hypotheses for them tested, or in offline mode, where a static model is built from the training data and never updated. The former approach has the advantage of being able to adapt to non-stationary processes and is appropriate for streaming applications where real-time predictions are required. The offline approach is more useful when a certain amount of data is available and an explicit picture of the evolution of the process is required. Results obtained for the same datasets in offline mode show minor performance degradation in terms of accuracy and fewer predictions being made as a result of the adaptive learning component being disabled after the training set.

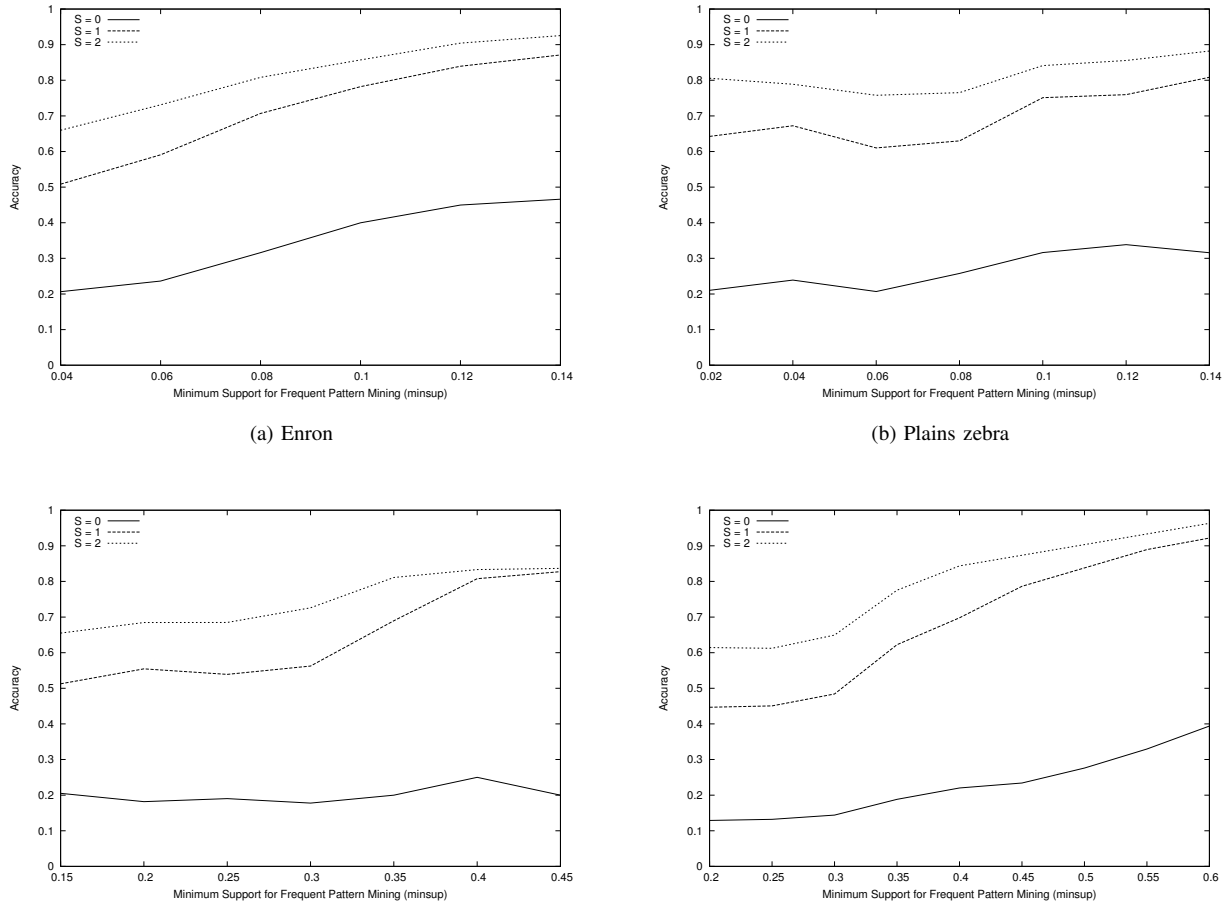
Finally, efficiency is a major motivating factor in our design. With large networks, and moreover, streaming network data becoming available, many theoretical approaches do not scale well. Our work takes advantage of mature, mainstream data mining algorithms which are specifically designed to handle large datasets. Our method can handle streaming data and make real-time predictions.

IV. CONCLUSION AND FUTURE WORK

We have presented a generic, online algorithm for predicting a partial structure of a temporal network. Our algorithm is accurately able to predict the structure of a temporal network at an arbitrary unseen timestep with a slack of one timestep. It can work as an online algorithm by continually updating its model of the underlying process, or be used in offline mode to predict a structure based on a static learned model for an arbitrary number of timesteps. It is thus suitable for applications with streaming data where real-time predictions are required, as well as for more analytical applications where the evolution of a process needs to be modeled precisely. We were able to accurately predict, within one timestep, correlations in stock prices, e-mail activity in a corporation, and social groupings in populations of Plains zebra, with the resolution of a timestep varying from one day for the e-mail domain to 30 days for a version of the stock market graph.

While the prediction of a partial structure of the network at each timestep has many applications, there is also a lot of scope for future work. Perhaps most importantly, in our current approach, frequent subgraphs are extracted once from a training network. This precludes the possibility of capturing groups of interactions which become frequent at a later point. One possible solution is to use locally frequent subgraphs, or subgraphs that have been frequent in the last t timesteps. These could be dynamically extracted from the data stream, possibly replacing older subgraphs.

The traditional focus for frequent subgraph mining has been finding frequent structures in databases of chemical compounds [53], [54]. Subgraph mining in temporal networks suffers from the lack of an equitable support calculation method similar to mining itemsets with multiple minimum supports [55]. For example, if three actors i, j and k are present in fewer than $minsup$ timesteps, then they will not be included in any frequent subgraph. However, if i, j and k always interact when they do appear, then the interaction between them is predictable. What is needed is a method for mining subgraphs based on more than an arbitrary $minsup$ value. One possible method is to extract statistically significant



(a) Enron

(b) Plains zebra

(c) Stock Market - 1

(d) Stock Market - 2

Fig. 1. Accuracy at different levels of minimum mining support and slack.

interactions based on deviations from a random graph model over the same set of actors. However, it is unclear what would be a reasonable null hypothesis for temporal networks. A different approach could combine measures like common neighbors [4], [42] to include interactions which might not be statistically significant in the training graph, but likely to interact in the future.

There are other possible extensions and variations of the algorithm specific to a given application domain. However, the proposed algorithm provides an efficient and accurate basis for those extensions.

ACKNOWLEDGMENTS

The authors would like to thank Dan Rubenstein, Ilya Fischhoff, and Siva Sundaresan of the Department of Ecology and Evolutionary Biology at Princeton University for sharing the Plains Zebra data. Their work was supported by the NSF grants CNS-025214 and IOB-9874523. This work is supported by the Microsoft award 14936.

REFERENCES

[1] A. L. Barabasi, H. Jeong, Z. Neda, E. Ravasz, A. Schubert, and T. Vicsek, "Evolution of the social network of scientific collaborations," *Physica A: Statistical Mechanics and its Applications*, vol. 311, no. 3-4, pp. 590–614, August 2002. [Online]. Available: http://dx.doi.org/10.1016/S0378-4371(02)00736-7

[2] K. Börner, J. Maru, and R. Goldstone, "The simultaneous evolution of author and paper networks," *PNAS*, vol. 101, no. Suppl 1, pp. 5266–5273, 2004.

[3] K. Börner, L. DallAsta, W. Ke, and A. Vespignani, "Studying the emerging global brain: Analyzing and visualizing the impact of co-authorship teams," in *Complexity, Special issue on Understanding Complex Systems*, 2006, in press.

[4] D. Liben-Nowell and J. Kleinberg, "The link prediction problem for social networks," *Proceedings of the twelfth international conference on Information and knowledge management*, pp. 556–559, 2003.

[5] J. J. Ramasco, S. N. Dorogovtsev, and R. Pastor-Satorras, "Self-organization of collaboration networks," *Physical Review E*, vol. 70, p. 036106, 2004.

[6] Z. Bar-Yossef, A. Broder, R. Kumar, and A. Tomkins, "Sic transit gloria telae: Towards an understanding of the web's decay," in *Proceedings of WWW*, 2004.

[7] P. Desikan and J. Srivastava, "Mining temporally evolving graphs." New York, NY, USA: ACM Press, 2004, pp. 13–22.

[8] D. Fetterly, M. Manasse, M. Najork, and J. Wiener, "A large-scale study of the evolution of web pages," in *Proceedings of WWW*, 2003.

[9] J. Kleinberg, "Temporal dynamics of on-line information streams," draft chapter for the forthcoming book Data Stream Management: Processing

High-Speed Data Streams (M. Garofalakis, J. Gehrke, R. Rastogi, eds.), Springer.

[10] W. Koehler, "A longitudinal study of web pages continued: a consideration of document persistence," *Information Research*, vol. 9, no. 2, p. paper 174, 2004, [Available at http://InformationR.net/ir/9-2/paper174.html].

[11] R. Kumar, J. Novak, P. Raghavan, and A. Tomkins, "On the bursty evolution of blogspace," in *Proc. International WWW Conference*, 2003.

[12] A. Ntoulas, J. Cho, and C. Olston, "What's new on the web? The evolution of the web from a search engine perspective," in *Proceedings of WWW*, 2004.

[13] R. Pastor-Satorras and A. Vespignani, *Evolution and structure of the Internet*. Cambridge: Cambridge University Press, 2004.

[14] D. Spinellis, "The decay and failures of web references," *Communications of the ACM*, vol. 46, pp. 71–77, 2003.

[15] S. Eubank, V. Kumar, M. Marathe, A. Srinivasan, and N. Wang, "Structural and algorithmic aspects of massive social networks," *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 718–727, 2004.

[16] S. Eubank, H. Guclu, V. Kumar, M. Marathe, A. Srinivasan, Z. Toroczkai, and N. Wang, "Modelling disease outbreaks in realistic urban social networks," *Nature*, vol. 429, pp. 429:180–184., Nov 2004, supplement material.

[17] M. Keeling, "The effects of local spatial structure on epidemiological invasions," *Proc. R. Soc. Lond. B*, vol. 266, pp. 859–867, 1999.

[18] M. Kretzschmar and M. Morris, "Measures of concurrency in networks and the spread of infectious disease," *Math. Biosci.*, vol. 133, pp. 165–195, 1996.

[19] L. A. Meyers, B. Pourbohloul, M. Newman, D. Skowronski, and R. Brunham, "Network theory and sars: Predicting outbreak diversity," *Journal of Theoretical Biology*, vol. 232, pp. 71–81, 2005.

[20] L. A. Meyers, M. Newman, and B. Pourbohloul, "Predicting epidemics on directed contact networks," *Journal of Theoretical Biology*, vol. 240, pp. 400–418, 2006.

[21] J. M. Read and M. J. Keeling, "Disease evolution on networks: the role of contact structure," *Proc. R. Soc. Lond. B*, vol. 270, pp. 699–708, 2003.

[22] J. Baumes, M. Goldberg, M. Magdon-Ismail, and W. Wallace, "Discovering hidden groups in communication networks," in *Proceedings of the 2nd NSF/NIJ Symposium on Intelligence and Security Informatics*, 2004.

[23] A. Broido and K. Claffy, "Internet topology: connectivity of ip graphs," in *Proceedings of SPIE ITCom*, 2001.

[24] K. Carley, "Communicating new ideas: The potential impact of information and telecommunication technology," *Technology in Society*, vol. 18, no. 2, pp. 219–230, 1996.

[25] L. Chen and K. Carley, "The impact of social networks in the propagation of computer viruses and countermeasures," *IEEE Trasactions on Systems, Man and Cybernetics*, forthcoming.

[26] D. Gruhl, R. Guha, D. Liben-Nowell, and A. Tomkins, "Information diffusion through blogspace," in *WWW'03*, 2003. [Online]. Available: citeseer.ist.psu.edu/article/gruhl04information.html

[27] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003.

[28] M. Tsvetovat, K. Sycara, Y. Chen, and J. Ying, "Customer coalitions in electronic marketplaces," in *Agent-Mediated Electronic Commerce III, Lecture Notes on Artificial Intelligence*, U. C. Frank Dignum, Ed. Springer-Verlag, 2003.

[29] J. Tyler, D. Wilkinson, and B. Huberman, "Email as spectroscopy: Automated discovery of community structure within organizations," in *Proceedings of the First International Conference on Communities and Technologies*, 2003.

[30] B. Wellman, "An electronic group is virtually a social network," in *Culture of the Internet*, S. Kiesler, Ed. Mahwah, NJ: Lawrence Erlbaum, 1997, pp. 179–205.

[31] J. Wang, Z. Zeng, and L. Zhou, "Clan: An algorithm for mining closed cliques from large dense graph databases." Los Alamitos, CA, USA: IEEE Computer Society, 2006, p. 73.

[32] H. Kim, I. Kim, Y. Lee, and B. Kahng, "Scale-Free Network in Stock Markets," *Journal of the Korean Physical Society*, vol. 40, no. 6, pp. 1105–1108, 2002.

[33] R. Breiger, K. Carley, and P. Pattison, Eds., *Dynamic Social Network Modeling and Analysis*. Washington, D.C.: The National Academies Press, 2003.

[34] T. Snijders, "The Statistical Evaluation of Social Network Dynamics," *Sociological Methodology*, vol. 31, no. 1, pp. 361–395, 2001.

[35] D. Kempe, J. Kleinberg, and A. Kumar, "Connectivity and inference problems for temporal networks," *J. Comput. Syst. Sci.*, vol. 64, no. 4, pp. 820–842, 2002.

[36] M. E. J. Newman, "From the Cover: The structure of scientific collaboration networks," *PNAS*, vol. 98, no. 2, pp. 404–409, 2001. [Online]. Available: http://www.pnas.org/cgi/content/abstract/98/2/404

[37] ——, "Clustering and preferential attachment in growing networks," *Physical Review E*, vol. 64, no. 2, p. 25102, 2001.

[38] L. Getoor and C. P. Diehl, "Link mining: a survey," *SIGKDD Explor. Newsl.*, vol. 7, no. 2, pp. 3–12, 2005.

[39] A. Popescul and L. Ungar, "Statistical relational learning for link prediction," *IJCAI Workshop on Learning Statistical Models from Relational Data*, 2003.

[40] J. Vert and Y. Yamanishi, "Supervised graph inference," *Advances in Neural Information Processing Systems*, vol. 17, pp. 1433–1440, 2005.

[41] D. Kempe, J. Kleinberg, and A. Kumar, "Connectivity and inference problems for temporal networks," *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pp. 504–513, 1999.

[42] J. O'Madadhain, J. Hutchins, and P. Smyth, "Prediction and ranking algorithms for event-based network data," *ACM SIGKDD Explorations Newsletter*, vol. 7, no. 2, pp. 23–30, 2005.

[43] X. Yan and J. Han, "Closegraph: mining closed frequent graph patterns," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '03)*. New York, NY, USA: ACM Press, 2003, pp. 286–295.

[44] J. Huan, W. Wang, J. Prins, and J. Yang, "Spin: mining maximal frequent subgraphs from graph databases," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '04)*. New York, NY, USA: ACM Press, 2004, pp. 581–586.

[45] C. Wang, W. Wang, J. Pei, Y. Zhu, and B. Shi, "Scalable mining of large disk-based graph databases," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '04)*. New York, NY, USA: ACM Press, 2004, pp. 316–325.

[46] T. Uno, T. Asai, Y. Uchida, and H. Arimura, "LCM: An Efficient Algorithm for Enumerating Frequent Closed Item Sets," *Proc. IEEE ICDM03 Workshop FIMI03*, 2003.

[47] D. Burdick, M. Calimlim, and J. Gehrke, "Mafia: A maximal frequent itemset algorithm for transactional databases," *Proceedings of the 17th International Conference on Data Engineering*, pp. 443–452, 2001.

[48] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pp. 487–499, 1994.

[49] B. Klimt and Y. Yang, "The Enron Corpus: A New Dataset for Email Classification Research," *Proceedings of the European Conference on Machine Learning*, 2004.

[50] J. I. Adibi, "Enron email dataset." [Online]. Available: {http://www.isi.edu/~adibi/Enron/Enron.htm}

[51] I. R. Fischhoff, S. R. Sundaresan, J. Cordingley, H. M. Larkin, M. J. Sellier, and D. I. Rubenstein, "Social relationships and reproductive state influence leadership roles in movements of plains zebra (*equus burchellii*)," *Animal Behaviour (in press)*.

[52] "Yahoo! Finance," accessed June 20, 2006. [Online]. Available: http://finance.yahoo.com/

[53] M. Kuramochi and G. Karypis, "Frequent subgraph discovery," 2001, pp. 313–320.

[54] A. Inokuchi, T. Washio, and H. Motoda, "An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data," *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, pp. 13–23, 2000.

[55] B. Liu, W. Hsu, and Y. Ma, "Mining association rules with multiple minimum supports," *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 337–341, 1999.