Contents

Chapter 4

Mining Coupled Edges

In this chapter, we deal with the problem of mining strong temporal correlations between interactions in a dynamic network, which we call *coupled edges*. Typical dynamic network datasets can contain thousands or millions of edges occurring and re-occurring on a continous streaming basis. It is likely that a majority of these edges are unpredictable with any tractable model, and yet there are likely to be interactions that are quite regular and predictable. In a phone call network, for example, a user's outgoing and incoming phone calls may seem entirely random in aggregate, but there might be a regular phone call that is always made on the first Friday of each month, or between business hours each Tuesday. Similarly, if an e-mail sent from one person to another is frequently followed by a reply within a few hours, or a forward to a third person within five minutes, that is evidence for a specific type of relationship between them. The crux of this chapter is to tease out a few of these *predictive*, tightly coupled relationships solely from the dynamics of the network, *i.e.,* to extract structural correlations solely from network dynamics.

In contrast to other data mining methods, our definition of a strong temporal correlation requires a degree of predictive power on unseen data, rather than being based on descriptive statistics. In the examples above, the temporal correlations have to be in both directions – if an e-mail from A to B is correlated with a reply within a few hours, it should also be that a reply from B to A is preceded by an initial e-mail from A. In other words, an e-mail from A to B *predicts* a reply within a few hours. By using predictive power on unseen data as the property of interest for data mining, we overcome a number of issues: model selection between competing models is trivial (choose whichever model predicts unseen data better), data mining results come with a degree of statistical generality (as opposed to being based on descriptive statistics like frequency), and the significance of a particular mined pattern is easy to assess based on the degree of predictability.

In many cases, most interactions present in a network might be unpredictable without external information, but the ones that are may be of scientific and commercial value. Some of these practical uses include, for example, (i) *community inference*, where regular interactions between two people can signify a special relationship between them, (ii) *marketing*, where advertising opportunities arise from knowing when a particular interaction between two people is going to occur, (iii) *unusual activity detection*, such as churn prediction [Wei and Chiu, 2002], which can be triggered by many of an individual's regular interactions ceasing to occur within a short period of time, (iv) *detecting plausible hidden relationships*, where two otherwise unconnected interactions between different sets of individuals are found to be temporally predictive, and (v)

spam detection based on mechanistic and predictable interaction patterns, even when they are perturbed by random noise.

Specific types of predictable behavior have been studied independently in the literature, and one of our goals here is to automatically and transparently detect many such restricted subsets:

1. **Periodic patterns**, such as a weekly e-mail exchange between two people, which have been found at different timescales in many types of dynamic networks [Lahiri and Berger-Wolf, 2008].

2. **Bursty behavior**, such as periods of inactivity followed by intense periods of activity, has been observed in e-mail networks [Malmgren et al., 2009] and web server connections [Frias-Martinez and Karamcheti, 2003]. Although the authors of [Malmgren et al., 2009] analyzed the aggregate behavior of users, and not specific interactions, there are compelling intuitive reasons (such as the human diurnal cycle) to believe that 'bursts' should be observed at the level of individual interactions as well.

3. **Temporal association rules**, where the occurrence of a particular event a leads to event b occurring after a fixed time interval [Oates et al., 1997, Tung et al., 1999, Frias-Martinez and Karamcheti, 2002, Lahiri and Berger-Wolf, 2007]. A variant of temporal association rules is the *frequent episode* formulation [Mannila et al., 1997a, Laxman et al., 2005].

The methods we describe in this chapter model the temporal relationships above, and also have a number of other attractive features. A typical pre-processing step for dynamic network data is to quantize the time stream of interactions into coarser timesteps. Where the original dataset might have a timestamp for each edge at the resolution of about a second, typical pre-processing groups all edges into shifting windows of hours, days, or even months (see Chapter 2 for a description of various quantization levels used in the literature). Our techniques operate in continuous time and do not require any time quantization; no information is lost, and the user is spared a parameter and an extra preprocessing step. Furthermore, our techniques operate on differenced time series, *i.e.,* by considering the time *delay* between edges rather than their occurrence times, and can thus handle short- and long-range interactions seamlessly within the same framework. Finally, our method is designed specifically for networks and takes advantage of network structure, rather than retrofitting techniques for more limited types of data by assuming, *e.g.,* independence of edges.

4.1 Problem Definition

We begin with interaction data in its unprocessed form: a continuous stream of interactions between uniquely identifiable entities data at the finest possible timescale. In cases like e-mail and phone call networks, this means that each interaction (edge) carries a timestamp on the granularity of about a second. In almost all other types of network analysis, timestamped interactions are quantized into a coarser timescale where timesteps comprise of hours, days, months, or even years. Multiple interactions within the same quantized timestep are either considered a single interaction, or weighted by the count of interactions within that timestamp. For our approach, however, we work in continuous time and do *not* require the user to quantize interaction data. We see this as an advantage of our method.

The focus of our method is to mine *coupled edges*. Figure 4.1 illustrates a pair of edges in a dynamic network stream that are coupled. In this case, occurences of the edge $(1, 3)$ reliably predict occurrences of

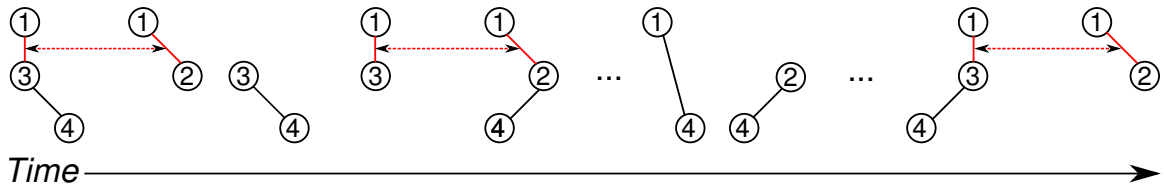


Figure 4.1: Streaming interaction data in continuous time with a coupled pair of edges shown in red. Note that the coupling is directional.

edge $(1, 2)$ after a fixed time period, even though the opposite is not necessarily true. The mining problem is to extract these edges under two additional constraints: there is generally more noise than signal in the dataset (*i.e.,* most edges are likely to be uncoupled), and the temporal relationship between coupled edges might not be a simple fixed time delay, as shown in Figure 4.1. We review other approaches that make a fixed time delay assumption in Section 4.2.

Definition 4.1.1. *(Coupled edges)* An ordered pair of edges $\langle E_1, E_2 \rangle$ is *coupled* if occurences of E_2 can be predicted solely from prior occurrences of E_1, where possibly $E_1 = E_2$.

In the context of a data mining problem, Definition 4.1.1 is the property of interest, and we aim to develop tractable techniques that will extract pairs of edges from large quantities of data where the property holds. Definition 4.1.1 can be broken down into three components: how an edge is predicted, how the prediction of a particular algorithm is evaluated, and how that translates into a mining problem. We deal with each part in the subsequent subsections.

4.1.1 Prediction

Given a pair of candidate edges E_1 and E_2, we want to find the degree to which they are coupled by measuring how well E_1 predicts E_2. The method of determining which edge pairs to test is part of our solution to the problem, and is described in Section 4.3. For now, we assume that we are given an edge pair, where E_1 is called the *trigger* and E_2 the *response*. Let the *timelist* of an edge (u, v) be the ordered sequence of timestamps at which it occurs, denoted $\mathcal{T}(u, v)$. We also assume that there is significant overlap in the timelists of the trigger and response. Without this assumption, the case of time-shifted coupling is very hard to detect. If, for example, five instances of a trigger within an hour are coupled to five instances of a response several hours later, it is hard to distinguish that relationship from noise, or an autocorrelation in the response.

The timelist of the response edge is segmented into training, validation, and testing segments. The training segment is used to learn the parameters of a model that will predict occurrences of the response based on historical observations. The validation segment, in our formulation, is used for model selection and to avoid overfitting, although other regularization or heuristic methods may be used as well. The prediction problem is then as follows.

Definition 4.1.2. *(Prediction)* Given a set of candidate models $M_i(\theta_i)$ each parameterized by θ_i, and training timelists for the trigger and response edges, find the optimal parameters of the model. Choose the

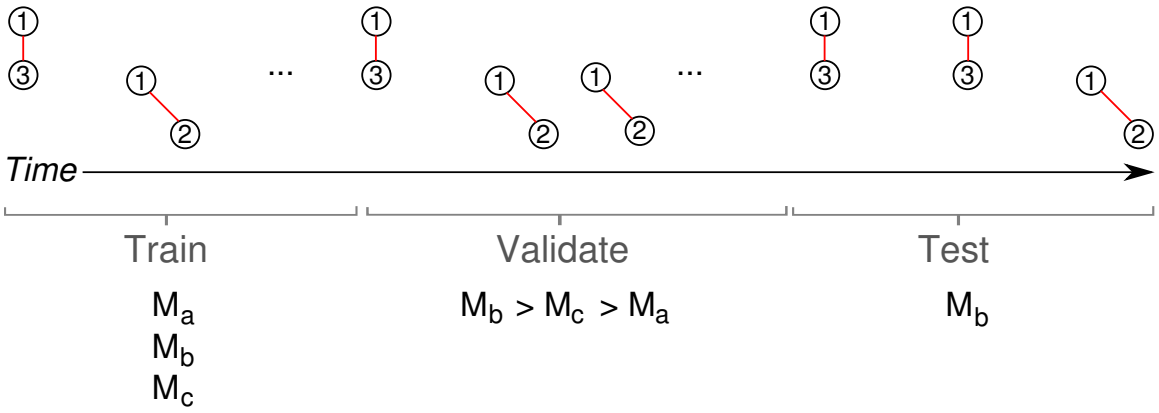


Figure 4.2: Setup for testing the predictability of a pair of edges.

model M' that maximizes an *evaluation score* E on the validation timelists $\mathcal{T}_V$.

$$M' = \arg\max_{M_i} E(M_i(\theta_i), \mathcal{T}_V)$$

Note that this describes a fairly generic machine learning setup. The final segment of the timelist will be used later for testing and mining. We use the validation segment to test the performance of multiple prediction models on unseen data, while still withholding a final test segment to evaluate the chosen model. A number of other statistical mechanisms can be used for model selection, such as the Bayesian Information Criterion [Raftery, 1999], but since our final goal is to test predictable edges, using predictive performance on unseen data is a logical choice.

4.1.2 Evaluation

A key factor of our problem formulation is that the timestamp for each observed interaction is a positive real value, and consequently, predictions of the next occurrence of each interaction are also real values.[1] As a result, matching an edge's predicted occurrences to its observed occurrences is non-trivial, and traditional measures of prediction efficacy based on a confusion matrix, such as the F_1-score or ROC curves, cannot be used directly. Furthermore, since data is received on a streaming basis, predictions are made continuously and incorporate previously seen observations, which further complicates performance evaluation. We therefore propose a novel evaluation framework to handle both these facets of our problem.

To handle both these facets of our problem, we propose an evaluation framework that takes into account the fact that both observations and predictions of interactions occur in continuous time, and that data is received on a continuous, streaming basis.

For a particular edge, a prediction algorithm generates a *predicted* timelist $\mathcal{T}_P$, which is to approximate a *true* (observed) timelist $\mathcal{T}_T$. Since the prediction algorithm also has a partial view of $\mathcal{T}_T$ as it streams by, we are also given a *predicted from* timelist $\mathcal{T}_F$, which contains the timestamp at which each prediction $t_i \in \mathcal{T}_P$ was made, *i.e.,* if x_i is the i^{th} element of $\mathcal{T}_F$ and y_i the i^{th} element of $\mathcal{T}_P$, then the prediction algorithm

[1] In practice, discrete-valued epoch seconds are used as timestamps for communications networks, but our argument applies generally to any fine-grained timescale.

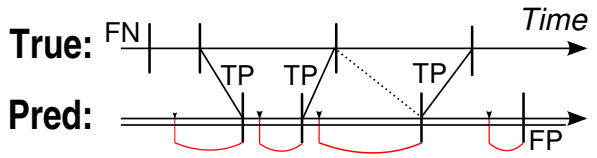


Figure 4.3: Example of matching a true and predicted timelist in continuous time in an online setting. Solid connecting lines are matched pairs, dotted lines are potential matches, and red lines show when each prediction was made.

generated a prediction of time y_i at time x_i, where naturally $y_i > x_i$.

We first define what qualifies as a *correct* prediction, and subsequently the relative numbers of true and false positives and negatives, and then *how close* (in continuous time) correct predictions are to their corresponding true occurrences. This can be formulated as a matching problem, based on the following constraints:

1. A predicted occurrence should only be matched to *at most* one of the true occurrences that are adjacent to it in time, or not at all.

2. A predicted occurrence at t_i cannot be matched to a true occurrence at t_j if the algorithm predicted t_i *at or after* time t_j. This is an intuitive constraint due to the online setting of the problem, and also prevents a trivial algorithm that predicts an occurrence at $t_j + 1$ immediately after observing an occurrence at t_j from achieving near-perfect performance.

3. The benefit of matching a true occurrence $t_i \in \mathcal{T}_T$ and predicted occurrence $t_j \in \mathcal{T}_P$ should be inversely related to the time difference between them, *e.g.,* as $(|t_i - t_j| + \epsilon)^{-1}$, subject to the first constraint, and where ϵ is a small constant to ensure that the quantity is well defined for a perfect algorithm.

Specifically, the constraints above can be formalized as a weighted bipartite maximum matching problem [West, 2001]. Each element of $\mathcal{T}_P$ and $\mathcal{T}_T$ is represented by a node in the two corresponding partitions of a bipartite graph. An edge connects $t_i \in \mathcal{T}_P$ and $t_j \in \mathcal{T}_T$ if they are sequentially adjacent in time (with no other elements of $\mathcal{T}_T$ or $\mathcal{T}_P$ in between) and satisfy constraints 1 and 2. The edge weight of (t_i, t_j) is defined as $(|t_i - t_j| + \epsilon)^{-1}$ for some small constant ϵ. Efficient algorithms exist for finding a maximum-weight matching in such a graph [Galil, 1986]. Figure 4.3 shows an example of the result of matching a predicted and a true timelist, where the red loops represent the *predicted from* timelist.

If M is the set of matched edges, then some traditional measures are conveniently expressed in terms of the cardinality of M, *i.e.,* the number of matched pairs:

$$\begin{aligned} \textit{Precision} &= |M|/|\mathcal{T}_P| \\ \textit{Recall} &= |M|/|\mathcal{T}_T| \\ \textit{Mean Absolute Error (MAE)} &= \frac{1}{|M|} \sum_{(t_i,t_j)\in M} |t_i - t_j| \end{aligned}$$

We summarize the Precision and Recall measures into their geometric mean, the F_1-score, and characterize the performance of a prediction algorithm on a single timelist in terms of the F_1-score and the MAE. The former gives a sense of the *completeness* of the predictions, *i.e.,* how well the number and temporal spacing

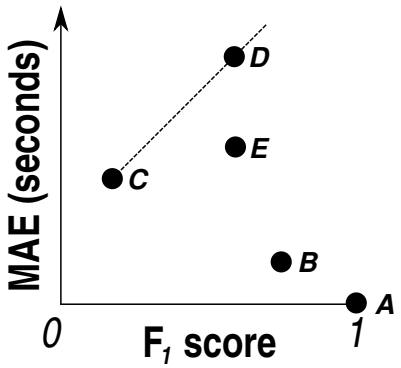


Figure 4.4: Some points in the evaluation plane.

of predictions approximate $\mathcal{T}_t$, and the latter of their *temporal accuracy*. A perfect prediction algorithm would have an F_1-score of 1 and an MAE of 0. Note that it is trivial to construct cases where one quantity is completely maximized or minimized at the cost of the other, and hence both measures need to be taken into account. We refer to the space of (F_1, MAE) pairs as the *evaluation plane*.

It is also often necessary to determine whether a given (F_1, MAE) pair of values is in some sense 'better' than another pair. This is needed, for example, in comparing the performance of two different algorithms or parameter sets, and for the final mining task described in the next subsection. The case of trivial dominance in a ranking is straightforward: one pair will have a higher F_1 score *and* a lower MAE than the other pair. However, for non-trivial cases, we introduce a parameter η that specifies the relative cost of improving one measure over the other. It can be interpreted as the *maximum increase in MAE for a unit gain in F_1-score* for one pair of values to dominate another. As an example, consider the evaluation plane shown in Figure 4.4, where the point labeled **A** represents a perfect prediction algorithm. The slope of the line connecting **C** and **D** is equal to η, and thus both points are considered equivalent in a ranking. Thus, an overall ranking for the points in Figure 4.4 is: $A, B, E, \{C, D\}$.

4.1.3 Mining

Finally, we return to the mining problem of finding the most tightly coupled edges. After model selection is done on the validation segment, the final withheld segment of testing data is used to assess the predictive accuracy of each model. The evaluation scores on this final test segment are used to determine the order of data mining results. Although it would seem that there are a number of tractability issues with the mining framework described in this section, we show how to overcome them by assuming a smaller (but still meaningful) dependency space in Section 4.3.

4.2 Related Work

In this section, we review literature directly related to mining coupled edges in dynamic networks. There are two broad categories of related work: a similar problem in network analysis called *link prediction*, and a number of approaches for other types of data that could, in principle, be applied. We outline the similarities, differences and shortcomings of these approaches in this section.

4.2.1 Link Prediction

There are two variants of the *link prediction* problem for networks: one deals with a single graph in which missing links are to be predicted based on some model of link formation, and another formulation that deals with predicting links over time in dynamic networks. An important difference between static and dynamic link prediction is that the former does not make predictions on a fine temporal scale, but rather just about which new links are likely to be formed 'in the future'. Both versions, however, are generally classification or ranking problems, and as such, focus less on the interpretability of the model than predictive performance. Although we describe both variants below, our problem has more in common with dynamic link prediction.

Static networks

The link prediction problem for static networks is defined as the question of ranking *unseen* edges by their likelihood of appearance in the future, without specifying *when* the edges will form. It therefore deals with the prediction of link formation, rather than the next occurrence of a link in a dynamic setting. The premise of this approach is that social networks tend to have structural similarities throughout the network that can be exploited to discover links that are either missing, or likely to form in the future. Liben-Nowell and Kleinberg [Liben-Nowell and Kleinberg, 2003] were one of the first to define the link prediction problem for social networks. They split a dynamic network along the time axis to form training and testing networks. Various structural node and edge measures were computed on the static network created from the first half of the dynamic network. Unobserved edges were then ranked by their probability of occurrence in the test network. They report promising results at predicting link formation using certain graph-theoretic measures, compared to a random and other simple baseline predictors.

A minor variation of the previous approach using weighted graph measures is described in Murata and Moriyasu [Murata and Moriyasu, 2007]. Kashima and Abe [Kashima and Abe, 2006] build upon the work of Liben-Nowell and Kleinberg [Liben-Nowell and Kleinberg, 2003] and others by first defining a model of network evolution that describes each edge as a probability. An edge label function $\phi^{(t)}$ is used to assign probabilities of existence to each edge, and the evolution of ϕ over time is assumed to be a Markov process. Specifically, a node in the network decides to 'transfer' an edge probability of one of its incident edges to an edge incident on another node in the network. The parameters of the model are the probabilities of each edge pair engaging in such a transfer. The authors describe a transductive algorithm based on Expectation-Maximization that jointly estimates the probabilities of each 'test' edge as the model is learned. It is worth noting that this model is superficially related to the *triadic closure* in social network analysis, which is the assumption that open triangles in a social graph tend to close with high probability [Wasserman and Faust, 1994, Kossinets and Watts, 2006].

O'Madadhain et al. [O'Madadhain et al., 2005] address link prediction in explicitly temporal event data, as well as the evolution of entity rank (importance) over time. For the link prediction component of their paper, the methodology used is similar to Liben-Nowell and Kleinberg [Liben-Nowell and Kleinberg, 2003] in that a dynamic network is split into two segments along the time axis for training and testing. However, instead of using structural graph features to rank unseen edges as in [Liben-Nowell and Kleinberg, 2003], O'Madadhain et al. treat the problem as a classification problem, and use both arbitrary entity attributes (*e.g.* geographic proximity, similarity of publication patterns [O'Madadhain et al., 2005]) as well as structural

features of the graph for link prediction. The learning techniques used are Naive Bayes and logistic regression.

Finally, a number of approaches to link prediction are based on the recent paradigm of *statistical relational learning* [Getoor and Taskar, 2007]. Wang et al. [Wang et al., 2007] learn undirected graphical models in the neighborhood of a pair of nodes whose edge incidence is to be predicted. Popescul and Ungar [Popescul and Ungar, 2003] approach the task by combining logistic regression with a feature generation algorithm that aggregates SQL queries on node attributes to predict future edges in a bibliographic database. An overview of work in this area can be found in Getoor et al. [Getoor et al., 2003] and Jensen and Neville [Jensen and Neville, 2003].

Dynamic networks

Dynamic link prediction is defined as the prediction of *when* previously observed and new interaction are going to occur again in the future, based on temporal and structural correlations. There are two related, but not necessarily equivalent, variants of the structure prediction problem:

1. **(Next Step Prediction)** Given a dynamic network $\mathcal{G}$ of t timesteps, predict a set of interactions (edges) that will occur exactly at timestep $t+1$.

2. **(Next Occurrence Prediction)** Given a dynamic network $\mathcal{G}$ of t timesteps, predict the future timestep when each interaction will next occur.

Huang and Lin [Huang and Lin, 2009] describe a method for both variants using a combination of static graph measures and standard time-series prediction techniques [Chatfield, 2004]. Specifically, they fit an ARIMA[2] model to the time series of binary occurrences or occurrence frequencies of each edge, independently of all other edges. The edge occurrence scores for the next timestep are then blended with the occurrence scores based on various static graph measures, like the ones described in Liben-Nowell and Kleinberg [Liben-Nowell and Kleinberg, 2003]. Due to large number of edges in a typical dynamic network, and the parameter optimization required to fit a single ARIMA model, this approach is unlikely to scale well to realistic networks. Furthermore, their results suggest that incorporating the structure of the network, such as the inter-dependencies between edges, into a predictive model would be advantageous. Although time series models are generally interpretable and applicable to mining coupled edges, the approach here would only allow the mining of coupled autocorrelations (*i.e.,* it would not be possible to mine a coupling between two *different* edges).

An interesting application of network prediction is the approach of Bunke et al. [Bunke, 2003, Bunke et al., 2005] in network anomaly detection. Given a stream of network traffic data, they use decision trees and 'median graphs' to detect connection anomalies. The trees are learned from a sequence of training data, where no anomalies are presumed to be present. This is extended by Pincombe [Pincombe, 2005], who uses Auto-Regressive Moving Average (ARMA) techniques instead. Since their focus is on predictive accuracy, their approach is not applicable to our mining problem.

Phithakkitnukoon and Dantu [Phithakkitnukoon and Dantu, 2007] propose a method for predicting whether a phone user will receive an incoming call at various hours of the day, without attempting to predict who the call is originating from. They make various simplifying assumptions, such as the call inter-arrival

[2]Autoregressive Integrated Moving Average

time and the number of outgoing calls per incoming call both following a Gaussian distribution. They report an error rate of approximately 5% on the call logs of 20 individuals, with no recall or specificity values.

Lahiri and Berger-Wolf [Lahiri and Berger-Wolf, 2007] describe an online technique that probabilistically estimates the delay between pairs of edges and uses this delay to predict a set of edges that will appear at an arbitrary point in the future. To aid the tractability of the approach, they measure the delay between frequent subgraph pairs instead of edge pairs. A simple heuristic mechanism dynamically adjusts which edge/frequent subgraphs pairs are used to make predictions. A limitation of this approach is that frequent subgraphs, or generally subgraphs of interest, have to be pre-specified. This limitation can be overcome by restricting the edge pair correlation to pairs of edges that are likely to be correlated, e.g. those edge pairs that share a common vertex.

Finally, Acar *et al.* [Acar et al., 2009] describe the use of matrix and tensor decompositions for solving both static and dynamic link prediction problems. For the static variant, they use a truncated low-rank Singular Value Decomposition (SVD) of a time-weighted graph, and then predict future edge formation by the scores of each (i, j) entry in the reconstructed matrix. Although the SVD of a sparse matrix can be computed very quickly, re-assembling all link prediction scores in the most general case requires iterating over all cells in the adjacency matrix, an intractable $O(V^2)$ operation for large networks. For the dynamic variant of the problem, they use standard time series prediction on the coefficients of a CANDECOMP/PARAFAC (CP) tensor decomposition of a dynamic network (see [Faber et al., 2003] for a recent review). Although this requires choosing some parameters, it has the advantage that CP tensor decompositions carry a lot of easily interpretable information and can be used for data mining.

As we noted before, the prediction algorithms presented here, with the exception of the CP tensor decomposition of [Acar et al., 2009] and the structure prediction approach of [Lahiri and Berger-Wolf, 2007], are generally black boxes from which it is difficult to extract meaningful relationships between edges.

4.2.2 Event Prediction

Event prediction aims to learn signatures that precede certain target events in a stream of symbolic or numerical data. The overall goal is to be able to predict occurrences of specific events before they happen, such as fraud, hardware failure, or intrusion into a computer network, by monitoring a stream of log messages. It can be seen as a special case of dynamic network next-step prediction, where we do not want to predict the entire graph at timestep t in a dynamic network, but just a specific part of it. y It is also more specialized in the sense that it generally deals with low-dimensionality streams, and can sometimes be treated as a conventional supervised learning problem. As a result, much of the work in this area has focused on specific applications instead of general techniques.

Fawcett and Provost [Fawcett and Provost, 1999] call this class of problems 'activity monitoring', and outline several important issues in how it differs from conventional classification problems. Among their contributions are a generic framework for modeling event prediction problems and an illustration of how prediction performance should be quantified. They point out that using accuracy or error to judge the merits of an event prediction model has its shortcomings, and suggest using ROC curves [Fawcett, 2004] or a specialized variation for activity monitoring, the AMOC curve [Fawcett and Provost, 1999].

A number of approaches to event prediction use conventional classifiers like Support Vector Machines (SVMs) or rule-based ones built from mining association rules, especially in the context of hardware failure

prediction from event logs and network intrusion prediction. Vilalta and Ma [Vilalta and Ma, 2002] mine frequent events from a time window preceding target events, and then combine these rules into a classifier. Domeniconi *et al.* [Domeniconi et al., 2002] combine Singular Value Decomposition on the co-occurrence of events with an SVM to phrase the problem as a conventional classification problem. Liang *et al.* [Liang et al., 2006] look for simple correlations in failure patterns in supercomputer hardware event logs. A more sophisticated approach to the same problem using Hidden Semi-Markov Models is adopted by Salfner and Malek [Salfner and Malek, 2007]. Finally, an interesting approach in a different, numerical time-series domain involves using a support feature machine to perform subset selection as well as prediction of target events [Chaovalitwongse et al., 2007].

Various forms of intrusion detection in computer network traffic analysis can also be thought of as event prediction, where the 'event' being predicted is an anomalous usage pattern indicative of an intrusion. Kannadiga *et al.* [Kannadiga et al., 2007] treat the intrusion prediction problem in much the same way as the event prediction approaches mentioned earlier, although the connection is not explicit. Given a stream of network traffic data, Bunke *et al.* [Bunke, 2003, Bunke et al., 2005] use decision trees and median graphs to detect anomalies. This is extended by Pincombe [Pincombe, 2005], who uses Auto-Regressive Moving Average (ARMA) techniques instead.

A major limiting factor in modeling dynamic network data as a log stream is scalability. Naively, one might consider each edge in a dynamic network as a type of log symbol, much like the itemset representation mapping we used in Chapter 3. However, there are two problems with this: all structural information about the graph is lost, and the number of symbols scales with the number of unique edges. Since many of the approaches mentioned in this section were designed with relatively small alphabets in mind, they might not scale well when the alphabet size increases to hundreds of thousands or even millions of edges. Furthermore, since we cannot exploit network structure to reduce the space of dependencies, the sheer number of possible dependencies between symbols quickly becomes intractable.

4.2.3 Sequential Patterns, Frequent Episodes and Association Rules

Mining association rules and various forms of frequent patterns are probably among the oldest and best studied problems in data mining. Association rules in the form of $a \rightarrow b$ are expressions of the conditional probability of co-occurrence of sets of events [Agrawal and Srikant, 1994, Brin et al., 1997]. A generalization of this question considers temporal association rules where a and b occur in different transactions or timesteps, usually accompanied by an estimate of the distance or delay between them [Tung et al., 1999, Oates et al., 1997, Oates and Cohen, 1996]. Frequent episodes are frequently occurring temporal partial orders of events, where the entire partial ordered must be matched within a fixed time window [Mannila et al., 1997b]. Frequent sequential patterns are also closely related to frequent episodes [Pei et al., 2004, Harms and Deogun, 2004].

Tung *et al.* [Tung et al., 1999] introduce the mining of *inter-transactional association rules*, where the antecedent and consequent of the rule consist of disjoint itemsets and are separated by a temporal interval. They describe an algorithm where a user-defined parameter W controls the maximum allowable delay between the sides of each rule, and the *confidence* of the rule expressed the conditional probability of seeing the antecedent in *at most* W timesteps. Similarly, Oates *et al.* [Oates et al., 1997, Oates and Cohen, 1996] introduce almost identical notation for what they call *temporal dependencies*. The difference from

Tung *et al.* is that instead of having a window in which both antecedent and consequent must occur, the rules express the actual delay to expect between the antecedent and the consequent.

Both association rules and frequent episodes can used to build rule-based predictive models, usually by taking some subset of the most frequently occurring associations. An early system for building such a rule-based prediction model was CBA [Liu et al., 1998], which is a non-temporal methodology that operates on transactional databases. Association rules can also be used as a type of feature generation process to aid further classification [Lesh et al., 1999]. More recently, a connection between frequent episodes and Hidden Markov Models has been made, with applications in building a predictive model for streaming data [Laxman et al., 2008, Laxman et al., 2005]. They model a stream of events using a mixture model of frequent episodes mined from the stream, with sequential correlation between the mixtures. We build on their ideas by modeling the delays between interactions in a network, instead of modeling the occurrences of events themselves.

Mining algorithms for association and related rules generally use support (empirical frequency) as the property of interest, although a wide variety of other measures have been developed [Geng and Hamilton, 2006]. Our mining formulation varies from these approaches in that it does not use descriptive statistics on a single dataset as the mining criteria, but rather predictive models and performance on an unseen segment of data. Our approach also differs from mining approaches such as [Oates et al., 1997] and [Laxman et al., 2008] because we explicitly take advantage of the network structure of data. As mentioned in the previous section, although it is possible to retrofit dynamic networks into a multidimensional discrete symbol stream that can be mined by some of these algorithms, all connectivity information (and subsequently the 'symbol' dependency structure) is lost in such a transformation. The space of possible rules is therefore much larger when the methods described in this section are applied, making the algorithms more intractable. Finally, even though our coupled edge rules superficially resemble the $a \rightarrow b$ rules mined by [Oates et al., 1997], the $\rightarrow$ operator in our case can be a complex HMM-modeled sequential dependency, instead of a fixed constant as used by [Oates et al., 1997] and [Tung et al., 1999].

4.3 Modeling time delays

We now present our primary contribution in this chapter: a flexible model of regular behavior, which encompasses many previously studied forms of predictable patterns, is insensitive to the timescale at which such patterns occur, and does not require time to be discretized. It is a means for solving the prediction part of the mining problem described in Definition 4.1.2, and is the core of our mining algorithm.

Consider the two timelists shown in Figure 4.5, which are typical examples of interactions that can be modeled easily for future prediction. The upper timelist is an example of an interaction that occurs in 'bursts', such as a phone call between two people that occurs mainly during business hours. The lower timelist is an example of a periodically recurring interaction. Our model is based on three assumptions about these timelists: first, that the time delays between consecutive occurrences are drawn from a mixture distribution, second, that there is significant sequential correlation between consecutive time delays, and finally, that the dynamics of a particular edge are stationary for as long as the edge persists.

We describe a novel Hidden Markov Model [Rabiner, 1989] (HMM) formulation for modeling time delays between a *trigger* and a *response* edge, denoted $E_T = (u, v)$ and $E_R = (x, y)$ respectively, and collectively

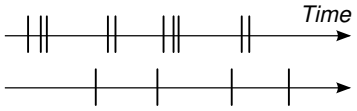


Figure 4.5: Examples of timelists.

referred to as an *edge pair*. E_T and E_R do not have to be solitary or even distinct interactions, but could also be, for example, frequently occurring subgraphs of interactions, or external events, such as $E_T = \{\texttt{First day of month}\}$ and $E_R = (a, b)$.

We model independent temporal dependencies over two types of edge pairs: those in which $E_T = E_R = (u, v)$, or autocorrelations, and pairs in which $E_T \neq E_R$. On observing an occurrence of E_T at time t, an HMM with continuous emissions specific to the $E_T \rightarrow E_R$ edge pair is used to generate a time delay $\delta \in \mathbb{R}^+$ until the next occurrence of E_R, thus generating a prediction of E_R at time $t + \delta$. Depending on its state, however, the HMM might also not generate a prediction at all, which allows more complex relationships to be modeled. We refer the reader to Rabiner [Rabiner, 1989] for more background on canonical HMMs and the notation used here.

A key contribution of our formulation is to model *time delays* between interactions using an HMM, instead of the more conventional approach of directly modeling the presence or absence of an interaction at each discrete timestep [Bunke et al., 2005,Huang and Lin, 2009]. By dealing with this higher-order representation, we overcome two problems that affect dynamic network analysis: the need to quantize interactions into discrete timesteps, and the fact that regular behavior can exist at different timescales [Lahiri and Berger-Wolf, 2008], which can be missed by approaches that use fixed length windows of real time for learning. As a result, it offers significant benefits over other conventional approaches and mining methodologies: there is no need to determine how far back in real time a model should retain data for learning, as is the case with time-series models [Huang and Lin, 2009], and there is no need to determine how much real time should be quantized into a timestep [Lahiri and Berger-Wolf, 2007, Huang and Lin, 2009, Bunke et al., 2005].

4.3.1 Model Setup

Given a trigger timelist $\mathcal{T}(E_T) = \langle t_1, ..., t_A \rangle$ and a response timelist $\mathcal{T}(E_R) = \langle r_1, ..., r_B \rangle$, we first compute a set of *delay pairs* that generate or approximate $\mathcal{T}(E_R)$ based on $\mathcal{T}(E_T)$. Each delay pair is generated by a trigger event and is of the form $\langle c_t, d_t \rangle$, where c_t is a binary value and d_t is a non-negative continuous value. The delay pair represents the action that a prediction algorithm should take to generate a response event at the correct time. If $c_t = 0$, then no response should be predicted. Otherwise, a response should be predicted to occur after a delay of d_t from the current time. The following algorithm computes a sequence of delay pairs from two timelists.

1. Sort the two timelists together into a single timelist $\mathcal{T} = \langle t_1, ..., t_{A+B} \rangle$. In cases of ties, sort responses *before* triggers. Let $type(t_i) = \texttt{trig}$ if element t_i came from the trigger timelist, and $type(t_i) = \texttt{resp}$ otherwise.

2. For each element t_i in the sorted timelist, where $i < A + B$ and $type(t_i) = \texttt{trig}$:

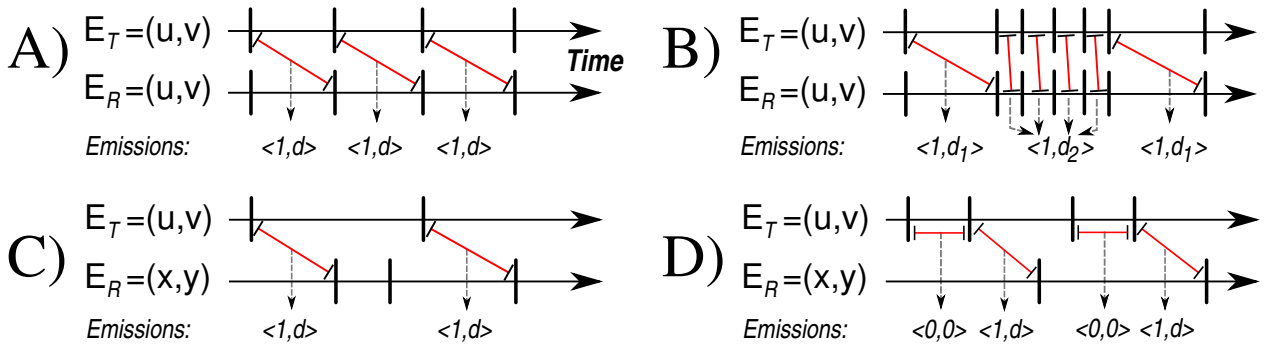
   (a) If $type(t_{i+1}) = \texttt{trig}$, output the delay pair $\langle 0, 0 \rangle$.

N	number of HMM states
$A_{N\times N}$	HMM state transition matrix
π_k	Initial probability of HMM state k
μ_k, σ_k^2	mean and variance of Gaussian emission distribution in state k
τ_k	probability of a continuous delay emission in state k
λ	set of all HMM parameters (all the above)
$q_t \in [1, N]$	HMM state at time t
$\phi(\mu, \sigma^2)$	Gaussian density with parameters (μ, σ^2)
$o_t = \langle c_t, d_t \rangle$	HMM emission at time t consisting of binary c_t and continuous $d_t \geq 0$
$b_k(o_t)$	Probability of emission o_t from state k
$\alpha_i(t)$	Forward variable; joint probability of observations up to time t and final state i
$\Gamma_i(t)$	Probability that emission at time t was generated by state i, given data up to t
$\gamma_i(t)$	Probability of being in state i at time t, given the entire sequence of observations

Table 4.1: NOTATION USED FOR HMM FORMULATION.

(b) Otherwise, output the delay pair $\langle 1, t_{i+1} - t_i \rangle$.

Note that if $E_T = E_R$, then $c_t = 1$ always. Figure 4.6 illustrates the generation of delays pairs from a trigger and response timelist.



Special case	*HMM states*	*Example Transition Matrix*	*Emission Probabilities*
(A) Partial periodic patterns	1	$\begin{pmatrix} 1 \end{pmatrix}$	$\mu_1 = d$ (period) $\tau_1 = 1$
(B) Bursty behavior	2	$\begin{pmatrix} 0.2 & 0.8 \\ 0.9 & 0.1 \end{pmatrix}$	$\mu_1 = d_1 \quad \tau_1 = 1$ $\mu_2 = d_2 \quad \tau_2 = 1$
(C) Temporal association rules	1	$\begin{pmatrix} 1 \end{pmatrix}$	$\mu_1 = d_1 \quad \tau_1 = 1$
(D) Complex temporal association rules	2	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$\mu_1 = 0 \quad \tau_1 = 0$ $\mu_2 = d \quad \tau_2 = 1$

Figure 4.6: Special cases of structure prediction, with corresponding HMM delay pair emissions. Red lines indicate time delays. The table shows examples of delay prediction HMMs that handle the special cases.

4.3.2 HMM Structure

Given a sequence of delay pairs, we use an HMM to model the sequence, where the emissions of the HMM are the delay pairs, and the hidden states correspond to different distributions (clusterings) over time delays. Let the emission of the HMM at position t be denoted $o_t = \langle c_t, d_t \rangle$, where $c_t \in \{0, 1\}$ and $d_t \in \mathbb{R}$. Unlike a typical HMM which generates emissions at every timestep of a discrete time process, our formulation calls for an emission on every incident of a trigger E_T being observed, with the continuous part of the emission specifying a time delay to the next occurrence of the response E_R, conditional on the binary part of the delay pair being true. When $c_t = 0$, the value of d_t is irrelevant, and we define it to be 0 for notational convenience. We chose a univariate Gaussian distribution to model the continuous delays emitted in each hidden state, conditional on $c_t = 1$, but any univariate continuous distribution may be substituted in its place without changing the framework.[3]

The distribution of emissions in each hidden state is governed by three parameters: τ_k, μ_k, σ_k.

$$\begin{aligned} P[c_t = 1 | q_t = k] &= \tau_k \\ P[c_t = 0, d_t = 0 | q_t = k] &= 1 - \tau_k \\ P[d_t | q_t = k, c_t = 1] &\sim \phi(\mu_k, \sigma_k^2) \end{aligned}$$

Assume that the HMM has N hidden states, with λ representing the set of all HMM parameters (see Table 4.1). Based on the expression above, the probability b_k of an emission $o_t = \langle c_t, d_t \rangle$ from state k is defined as:

$$\begin{aligned} P[o_t | q_t = k, \lambda] &= b_k(\langle c_t, d_t \rangle) \\ &= \begin{cases} \tau_k \phi(\mu_k, \sigma_k^2) & : c_t = 1 \\ (1 - \tau_k) & : c_t = 0 \end{cases} \end{aligned} \tag{4.1}$$

In standard HMM theory, the Baum-Welch algorithm is used to estimate maximum-likelihood parameters for an HMM from data, given the number of states N and a suitable parametric form of the emission density b_k at each state k. The emission density in Equation 4.1 is one such suitable parametric form, and the standard Baum-Welch algorithm can be used to estimate HMM parameters from a sequence of paired emissions $O = \langle o_1 = \langle c_t, d_t \rangle, ..., o_T \rangle$ calculated from the training network segment. The derivation is identical to that of the canonical Baum-Welch algorithm [Rabiner, 1989], and we only list the final parameter update equations in terms of delay pairs (the M-step of the Baum-Welch EM algorithm) for completeness. Refer to

[3]Note that although call *durations* appear to be distributed according to a log-logistic function [Vaz de Melo et al., 2010], we model the time between call *initiations*.

Table 4.1 for notation.

$$\mu_i = \frac{\sum_{t=1}^{T} \gamma_i(t)(c_t \cdot d_t)}{\sum_{t=1}^{T} \gamma_i(t) \cdot c_t}$$
$$\sigma_i^2 = \frac{\sum_{t=1}^{T} \gamma_i(t)(c_t \cdot (d_t - \mu_i)^2)}{\sum_{t=1}^{T} \gamma_i(t) \cdot c_t}$$
$$\tau_i = \frac{\sum_{t=1}^{T} \gamma_i(t) \cdot c_t}{\sum_{t=1}^{T} \gamma_i(t)}$$

4.3.3 Prediction

Once the Baum-Welch algorithm has been used to estimate HMM parameters on the training segment, we use the learned model to make predictions on the validation or test segments. We know from the Forward procedure in HMM training that the forward variable $\alpha_i(t) = P[O_t, q_t = i|\lambda]$, which is the joint probability of the observation sequence ending in the state i at time t. We also know that $P[O_T|\lambda] = \sum_{i=1}^{N} \alpha_i(T)$, which is exactly the output of the Forward procedure. In an online setting, marginalizing the α variables yields the probability of being in a particular state i at time t, which we denote $\Gamma_i(t)$[4].

$$\Gamma_i(t) = P[q_t = i|O_t, \lambda] = \frac{\alpha_i(t)}{\sum_j \alpha_j(t)} \tag{4.2}$$

The Γ variables define a distribution over the states that caused the last observed emission. For prediction, however, we require the next likely emission, which in turn requires a distribution over the next state, *i.e.,* $P[q_{t+1}|O_t, \lambda]$. This is easily computed from the HMM transition matrix and the $\alpha_i(t)$ variables [Rabiner, 1989], following which we use two different methods to generate the next emission (delay prediction):

1. *(Expectation)* We first compute the expected probability of the next emission having a continuous component by averaging over the continuous emission probability τ_i of each state:

   $$E[c_{t+1}] = \sum_i P[q_{t+1} = i|O_t, \lambda] \cdot \tau_i$$

   If $E[c_{t+1}] > 0.5$, then we assume that the next emission will produce a time delay, so we compute d_t as the expected delay over all HMM states.

   $$E[d_{t+1}] = \sum_i P[q_{t+1} = i|O, \lambda] \cdot \mu_i$$

   The final prediction is then

   $$o_{t+1} = \begin{cases} \langle 1, E[d_{t+1}] \rangle & : E[c_{t+1}] > 0.5 \\ \langle 0, 0 \rangle & : E[c_{t+1}] \leq 0.5 \end{cases}$$

2. *(Maximum a posteriori)* Instead of computing an expected delay over all states, we can instead pick the most likely next state q' and generate an emission from it. This might be preferable in some cases,

[4] $\Gamma_i(t)$ is distinct from the $\gamma_i(t)$ variable in the Baum-Welch algorithm, which takes backward probabilities into account.

since delay predictions will not lie in between the expected delays of each HMM state, but will rather be drawn from a single HMM state.

$$q' = \arg\max_i P[q_{t+1} = i | O, \lambda]$$

$$o_{t+1} = \begin{cases} \langle 1, \mu_{q'} \rangle & : \tau_{q'} > 0.5 \\ \langle 0, 0 \rangle & : \tau_{q'} \leq 0.5 \end{cases}$$

4.3.4 Model selection and special cases

The only parameter in the learning and prediction processes described in the previous sections, other than the global (F_1, MAE) trade-off parameter η, is the number of HMM states N for each trigger-response pair, and whether to use maximum *a posteriori* (MAP) or expectation to generate predictions. In general, the number of HMM states and its connection topology is generally determined either by prior domain knowledge or other, sometimes heuristic, mechanisms [Rabiner, 1989]. However, since our objective is prediction, we fit models of various state sizes up to a small maximum number, using both MAP and expectation to generate predictions, and then choose the combination that produces the highest ranked (F_1, MAE) pair on the validation segment. The allows us to select a specific model, and also helps avoid overfitting on the training data.

The value that we suggest for the maximum number of HMM states is based on both a theoretical and an empirical argument. Figure 4.6 shows a number of special case HMMs that cover many previously studied types of regular behavior. In all the cases shown, two HMM states are sufficient to model interaction dynamics, suggesting that the maximum number of states to attempt fitting can be similarly low. Furthermore, a 3-state HMM in our formulation has 17 non-trivial parameters to estimate, so if the timelists of interactions are not sufficiently long, then the inference of parameters is likely to be noisy. In the datasets we examine, we found typical timelists to be quite short, which again supports the use of a small maximum number of HMM states. For these reasons, we suggest that a maximum of 3-state HMMs be used to model interaction dynamics in networks comparable to the ones we analyze here.

4.3.5 Tractability

With single interactions comprising E_T and E_R, the approach above is not computationally tractable even for relatively small networks with about 10^4 unique edges. We overcome this using two methods: the first is to only consider edges with timelists long enough to support meaningful statistical inference. Generally, this means that the number of available timelist points should be at least a reasonable constant factor larger than the number of HMM parameters to be estimated. As we will show in Section **??**, this eliminates a large number of the edges in real networks. The second method is to only consider temporal correlations between pairs of interactions where, (a) $E_T = E_R = (u, v)$, or autocorrelations, and (b) $E_T = (u, v)$ and $E_R = (u, w)$, or pairs of interactions that share a common node.

Modeling correlations between pairs of edges that share a common vertex is tractable as a side effect of the skewed degree distributions observed in many real-world networks [Newman, 2003]. The number of delay pairs D is equal to the number of edge pairs that share a vertex, which is exactly the number of edges

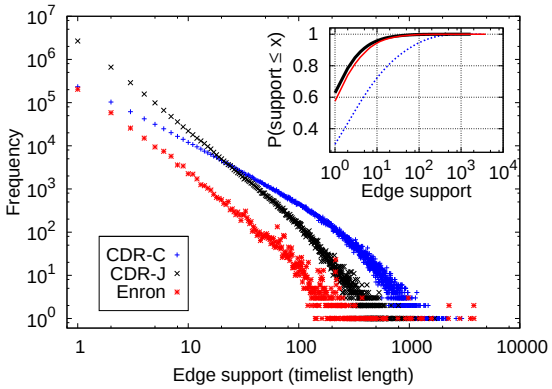


Figure 4.7: Edge support distributions: histogram on a doubly logarithmic scale, and empirical cumulative distribution on a partial logarithmic scale (inset).

in the line graph[5] of the original network. This number D in turn is proportional to the sum of the squares of degrees in the original graph [Lehot, 1974], *i.e.,* $D = \sum_{i=1}^{|V|} \frac{d_i(d_i-1)}{2}$, where d_i is the degree of vertex i. Modeling all such dependencies in real-world networks is therefore tractable if the assumption of a degree distribution skewed towards smaller degrees holds, as it generally has been found to [Newman, 2003, Leskovec and Horvitz, 2008, Nanavati et al., 2006, Chakrabarti and Faloutsos, 2006].

We also note that our algorithm is trivially parallelizable, and should scale linearly with the number of processors used.

4.4 Experimental Results

We ran our mining algorithm on the datasets described in the previous section to determine the extent to which edges in real networks are predictable. We describe two types of results: those specific to our learning algorithm, and applications of mining predictable interactions in general.

In all cases, we designated consecutive thirds of the timelist of each edge to serve as training, validation, and test segments, with the results of mining determined by performance only on the test segment. The maximum number of HMM states to fit is set to 4, in line with the reasons described in Section 4.3. We also require that all edges have timelists of length at least 40 in order to be used for learning and prediction. As shown in Figure 4.7, all three datasets show a commonality in terms of heavy skews in the *support* (timelist length) distribution of edges: only a small percentage of edges in each dataset have timelists of 40 occurrences or more.

Finally, we determined empirically that the η parameter has little impact on prediction performance, either quantitatively in terms of the mean and median (F_1, MAE) scores over all edges, or qualitatively in terms of the most predictable edges. We omit the results for brevity, but one possible reason for the insensitivity of the algorithm to η is that there are few edge pairs where different HMMs or parameter sets offer Pareto-optimal performance for predicting interactions. We therefore use a value of $\eta = 1000$ for our experiments, which corresponds to a trade-off, in our implementation, of at most 15 minutes in MAE for an

[5]Recall that the line graph of graph $G = (V, E)$ contains a vertex v'_i for every edge $e \in E$ in G, with an edge connecting v'_1 and v'_2 in the line graph if the corresponding edges $e_1, e_2 \in E$ in the original graph share a common vertex.

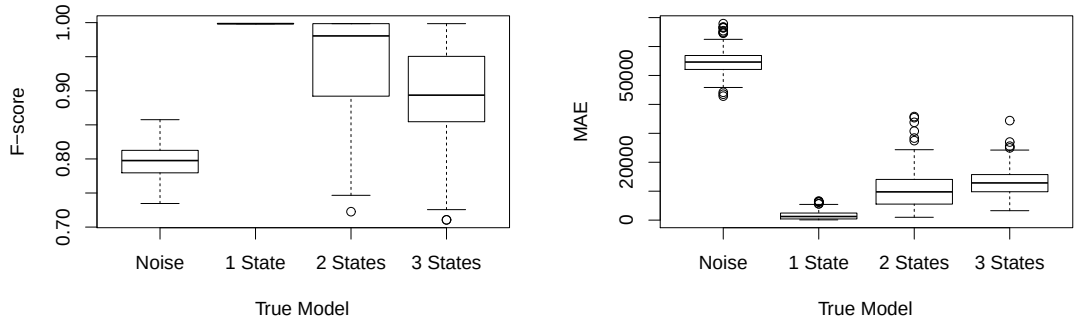


Figure 4.8: Fitted model test error on synthetic data.

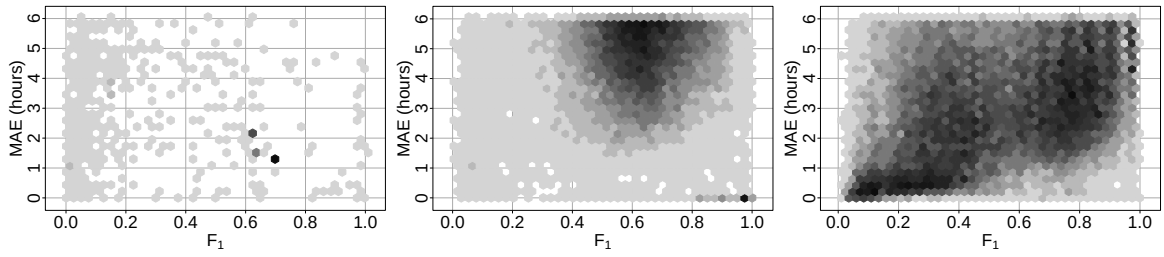
increase of 0.1 in F_1-score.

We ran two sets of experiments to test our algorithm. The first set is on synthetic data to show that our algorithm is able to distinguish between noise and regular interactions of the type described in Section 4.3. The second set of experiments is performed on the real datasets described in the previous section, and illustrates a number of practical applications.

4.4.1 Synthetic Data

We created synthetic data by randomly generating HMMs of the type described in Section 4.3. We created a different edge for each HMM, and placed 1,000 instances of the interaction on a timeline by iterating the HMM to generate delays. A total of 600 such interactions were generated by HMMs of 1, 2, and 3 states, in equal proportions, which encompasses the special cases shown in Figure 4.6. Finally, 200 additional 'noise' interactions were created by placing instances of the interaction uniformly at random on the timeline.

We designated consecutive thirds of the timeline of each interaction to serve as training, validation, and test segments. Figure 4.8 shows the distribution of F-1 and MAE scores of the fitted models for each type of edge on the test segment, where the boxes represent the interquartile range of the data. There is a clear differentiation between the MAEs of interactions generated by HMMs and those generated randomly. In the left figure, the 'noise' edges have the lowest F-1 score and the highest MAE, corresponding to the lower left part of the evaluation plane. Edges generated by a 1-state HMM, which is essentially an autocorrelated edge with a Gaussian time delay, have the highest scores. Even 3-state HMMs, which can generate a complex pattern of delays, are detected with greater efficacy – certainly in terms of MAE, and in a statistically significant way (as indicated by interquartile ranges) in terms of F-1 scare. This confirms that our algorithm and mining methodology can identify the class of regular behavior that encompassed by delay-generating HMMs.



(a) Enron (b) CDR-C (c) CDR-J

Figure 4.9: Hexagonally binned bivariate histograms of the evaluation plane for each dataset, showing the predictability of mined edges with MAE less than 6 hours on test data. Darker bins represent more edges.

4.4.2 Edge coupling on real data

Evaluation Planes

Figure 4.9 shows bivariate histograms of the evaluation planes for each dataset, truncated above an MAE value of 6 hours. Recall that the point $(1, 0)$ represents perfect prediction, so we are mainly interested in the distribution of edges at the lower right corner of the plane. Our first observation is that edges do exist in this region, implying that networks contain edges whose occurrences can be predicted quite accurately. We examine the detailed structure of these predictable edges shortly, in Section 4.4.3.

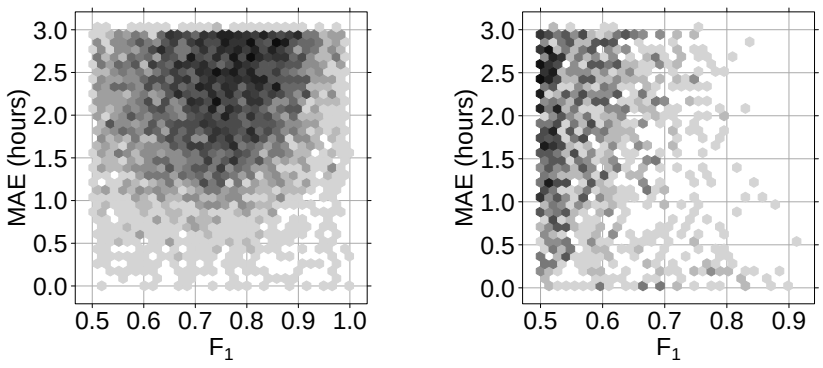
The CDR-C dataset shows two notable features: the dense semi-elliptical region of predictable interactions, and a small set of highly predictable edges with $F_1 > 0.8$ and very low MAE values. The former is possibly an artifact of the data collection bias. Since the CDR-C dataset sampled heavy users who made at least 3 calls a day, MAE values of 3-6 hours are not surprising due just to the frequency of calls. The latter feature is more interesting, and it is not clear from Figure 4.9 whether it is somehow characteristic of the dataset, or caused by, for example, automated systems.

The CDR-J dataset, unlike CDR-C, has no frequency-based sampling bias. It is interesting to note that in spite of the fact that CDR-J contains call records of *all* customers in a large geographical region, there does not seem to be a pronounced strip of highly predictable interactions like the one in CDR-C. Instead, the cluster around $F_1 \sim 0.8$ and MAE ~ 2.5 hours implies that there are a large number of edges that can be predicted with reasonable accuracy.

Finally, the Enron dataset appears to be sparse in terms of predictable edges, although it should be noted that the corpus is derived solely from the personal mailboxes of about 150 former Enron executives, and thus only a partial view of dynamics in the system. There are three prominent bins of edges, which we examine in more detail in Section 4.4.3.

Predictive Relationships

Recall from Section 4.3 that we model temporal dependencies between two types of edge pairs: autocorrelations, where an edge is both the trigger and the response, and correlations between pairs of edges that share a common vertex. Figure 4.10 shows a breakdown by type of edge pair of a window of the evaluation plane of the CDR-J dataset. In this window, autocorrelations of the form $(a, b) \to (a, b)$ are more prominent than



(a) $(a, b) \rightarrow (a, b)$ (b) $(a, b) \rightarrow (a, c)$

Figure 4.10: Distribution of different types of edge pairs for the CDR-J dataset.

pairwise correlations of the form $(a, b) \rightarrow (a, c)$, although a number of instances of the latter rank highly in terms of predictability. This suggests that structural correlations, in addition to temporal correlations, do exist in dynamic networks, and that these can be exploited for prediction purposes, in agreement with the qualitative and quantitative results of prior studies [Huang and Lin, 2009, Lahiri and Berger-Wolf, 2007].

Global performance and the η parameter

The η parameter, defined in Section 4.1.2, is the maximum allowable gain in MAE for a unit gain in F_1-score for one predicted timeline to be considered better than another. Since it is difficult to analytically determine an 'optimal' value for this parameter, we ran a series of experiments to assess the impact of the η parameter. In Table 4.2, we report the mean values of the F_1-score and MAE over all edge pairs for different values of η. Our findings indicate that the parameter has little impact on prediction performance, either quantitatively from Table 4.2, or qualitatively from scatterplots of the evaluation plane.

η (sec.)	F_1	Mean MAE	# States	Median # States
100	0.5430	3.365×10^5	2.36	2
1000	0.5446	3.366×10^5	2.42	2
10000	0.5440	3.365×10^5	2.45	2

Table 4.2: EFFECT OF VARYING η ON THE ENRON DATASET.

One possible reason for the insensitivity of the algorithm to η is that there are few edge pairs where different HMMs offer Pareto-optimal performance. Since the η parameter is only invoked during model selection when one HMM does not trivially dominate another, the insensitivity of performance to η implies that this scenario does not arise often enough to cause a significant difference in performance. We therefore use the value of $\eta = 1000$ for the remainder of our experiments, which corresponds to a trade-off of at most 15 minutes in MAE for a 10% gain in F_1-score when choosing the number of HMM states.

Fitted HMMs

Table 4.3 lists summary statistics of the learned models. Recall from our model description in Section 4.3 that the validation segment is used to choose the number of HMM states, and also between MAP and expectation-based prediction. It is interesting to note that while expectation-based prediction yields better results on the Enron dataset, predicting using the MAP method yields better performance in the CDR datasets, most notably in CDR-J. One possible explanation is that the CDR datasets contain interactions with highly clustered and separated delays; using an expected delay computed over multiple states could result in a predicted delay that lies between clusters, yielding poor performance relative to the MAP method. The reasons for this imbalance, particularly in CDR-J, warrant further study. We also note that the mean number of states is quite low, given the allowable range of $N = [1, 3]$, which indicates that our model selection strategy is successfully preventing overfitting to some degree.

Dataset	HMM states	Prediction method		
	Mean	MAP	Expect.	Equiv.
Enron	2.4 ± 1.07	30%	41%	29%
CDR-C	2.5 ± 1.09	40%	35%	25%
CDR-J	2.7 ± 1.16	56%	19%	25%

Table 4.3: FITTED MODEL PARAMETERS.

4.4.3 Applications of coupled edges

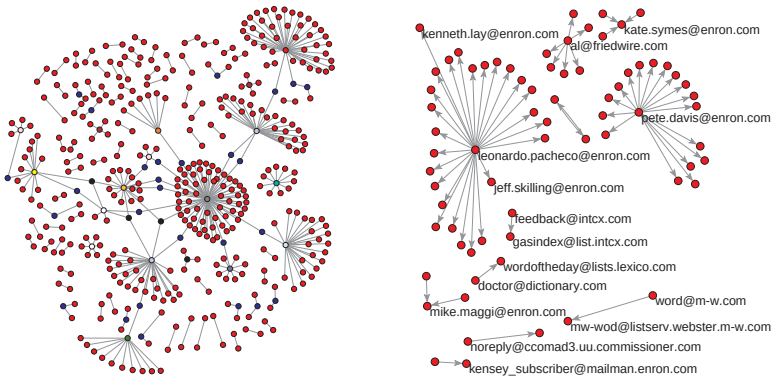
The Structure of Predictable Interactions

Figure 4.9 shows that there are a number of accurately predictable edges in all three networks. However, an important question is whether this is the result of genuinely interesting behavioral patterns and relationships of a wide range of individuals, or whether it is an artifact caused by, for example, an automated telemarketing or spam robot.

Figure 4.11 shows graph layouts of the highest ranked edges for the CDR-C and Enron datasets. In particular, Figure 4.11a shows the edges of CDR-C that are predictable with an average error of less than 10 minutes, and which comprise the dense strip in the bottom right corner of Figure 4.9b. There are several hub-and-spoke structures, which could represent, for example, call centers or businesses, but there are also a large number of isolated edges. It is difficult to determine what kind of specific behaviors are being discovered without further information about these nodes.

In the Enron dataset, however, we have at least partial information about the nodes. In particular, it is generally possible to tell whether an e-mail address corresponds to an automated program or to an Enron employee. Perhaps contrary to intuition, Figure 4.11b shows that the hub-and-spoke structures originate mainly from human accounts, and that several of the isolated edges are mailing lists.

Finally, the CDR-J graph layout (not pictured here) had 2,018 edges with $F_1 \geq 0.7$ and MAE ≤ 2 hours, the vast majority of which were comprised of degree-1 nodes, or isolated edges. There was a conspicuous dearth of hub-and-spoke structures, with just one such occurrence. This is surprising because CDR-J is a more complete dataset than CDR-C in terms of sample selection, and also more extensive in terms of geographical coverage, but does not seem to contain highly predictable hub-and-spoke structures. We can



(a) CDR-C, $F_1 \geq 0.9$, MAE $\leq$ 10 minutes. Node colors correspond to degree: red (1), blue (2), black (3), etc.

(b) Enron, $F_1 \geq 0.6$, MAE $\leq$ 2 hours. Unlabeled nodes are also `@enron.com` addresses.

Figure 4.11: The structure of highly predictable edges.

only speculate that known cultural or economic differences between the regions are responsible for this discrepancy.

Community Identification

A node in a dynamic network may have many frequent contacts with a subset of its neighbors, but it is often more interesting to ask what *type* of relationship exists between nodes. For example, a person's most frequent e-mail contacts could be mailing lists and close associates. It might be difficult to tell these apart based solely on the frequency of communication, but it may be possible to infer more based on the predictability of interactions. The same applies to CDR datasets, where predictable phone calls suggest that the parties involved have a specific reason for maintaining their schedule.

Among the most predictable interactions in the Enron dataset are the e-mails sent from the `pete.davis@enron.com` e-mail address. Figure 4.12 shows the complete set of outlinks for this e-mail address, as well as the predictive performance for each edge. Based on performance in the test segment, there seem to be three types of edges associated with this address. The first is a set of addresses shown in the top-left quadrant for which no model could be learned, either because the edges were too infrequent, or if HMM training repeatedly resulted in singularity solutions.[6] While this may not be considered a specific community, the second and largest subset of addresses is shown in the bottom two quadrants, and is characterized by extremely high F_1-scores and low MAE values, implying a high degree of predictability. The final subset is shown in the top right quadrant and is characterized by much higher MAE values.

Relationship Classification

Table 4.4 shows five of the highest-ranked edge pairs of the form $(a, b) \rightarrow (b, a)$ in the Enron dataset. In this case, the trigger is an e-mail, and its response is literally the reply. In all the cases listed, the HMM

[6] This is a well-known issue with the EM/Baum-Welch algorithm in general, see [Fraley and Raftery, 2007, Rabiner, 1989] for example.

Model			Performance	
a	b	μ_1, σ_1 (min.)	F_1	MAE (min.)
ka..symes	ev..metoyer	74.5 ± 36.6	0.631	293
mi..maggi	mi..nelson	1.6 ± 3.37	0.686	49
ka..symes	ke..thompson	72.3 ± 37.3	0.622	58.6
jo..griffith	al..villarreal	2.18 ± 1.9	0.615	82.2
al..villarreal	jo.griffith	6.77 ± 11.8	0.68	98.2

Table 4.4: HIGHEST RANKED $(a, b) \rightarrow (b, a)$ PAIRS IN THE ENRON DATASET. μ_1 AND σ_1 ARE THE MEAN AND VARIANCE OF ONLY THE MOST DOMINANT HMM STATE.

consisted of one clearly dominant state. It is interesting to note different styles of working relationships – the mean reply times vary between a few minutes to approximately an hour. Furthermore, finding symmetric pairs among the highest ranked interactions seems to be quite rare. This could be a result of fundamentally asymmetric relationships – for example, between an assistant and his superior – or a result of the fact that the Enron dataset offers only a partial view of all e-mail traffic.

4.5 Summary

We have described a novel framework and algorithm for mining edges in dynamic networks that exhibit temporally predictable interaction patterns, which we call *coupled edges*. Although there are a number of practical applications to our technique, we see its value as an exploratory data analysis tool. Its power stems from using a transparent prediction model (HMMs) with an easily interpretable structure that is grounded in the interaction dynamics of real systems. Furthermore, even if HMMs are the wrong model for the vast majority of edge dynamics, with apologies to the statistician George Box, there are certainly useful for the edges that we do mine. The following is a summary of our contributions in this chapter:

1. We formally defined the mining problem and an evaluation framework that deals with dynamic network data in continuous time.

2. We proposed an algorithm that predicts the future occurrence time of an edge, based on the prior occurrence times of either itself or another edge. Our algorithm models time delays between occurrences of edges, and is thus insensitive to timescales. It is also capable of representing and mining previously studied types of predictable patterns, such as partial periodic patterns [Han et al., 1999, Lahiri and Berger-Wolf, 2008], and temporal association rules [Oates et al., 1997].

3. We modeled temporal dependencies between edges that share a common vertex. This allows us to partially incorporate the structure of the network in an intuitive way, and is rendered tractable because of the skewed degree distribution observed in real networks [Newman, 2003]. Modeling all possible edge dependencies in real networks is intractable.

4. We demonstrated the applicability of our method on two industrial phone call datasets, and a publicly available e-mail dataset. Our experiments reveal that a large number of interactions are indeed predictable, sometimes to within a few hours, and that the structure of these interactions is quite complex.

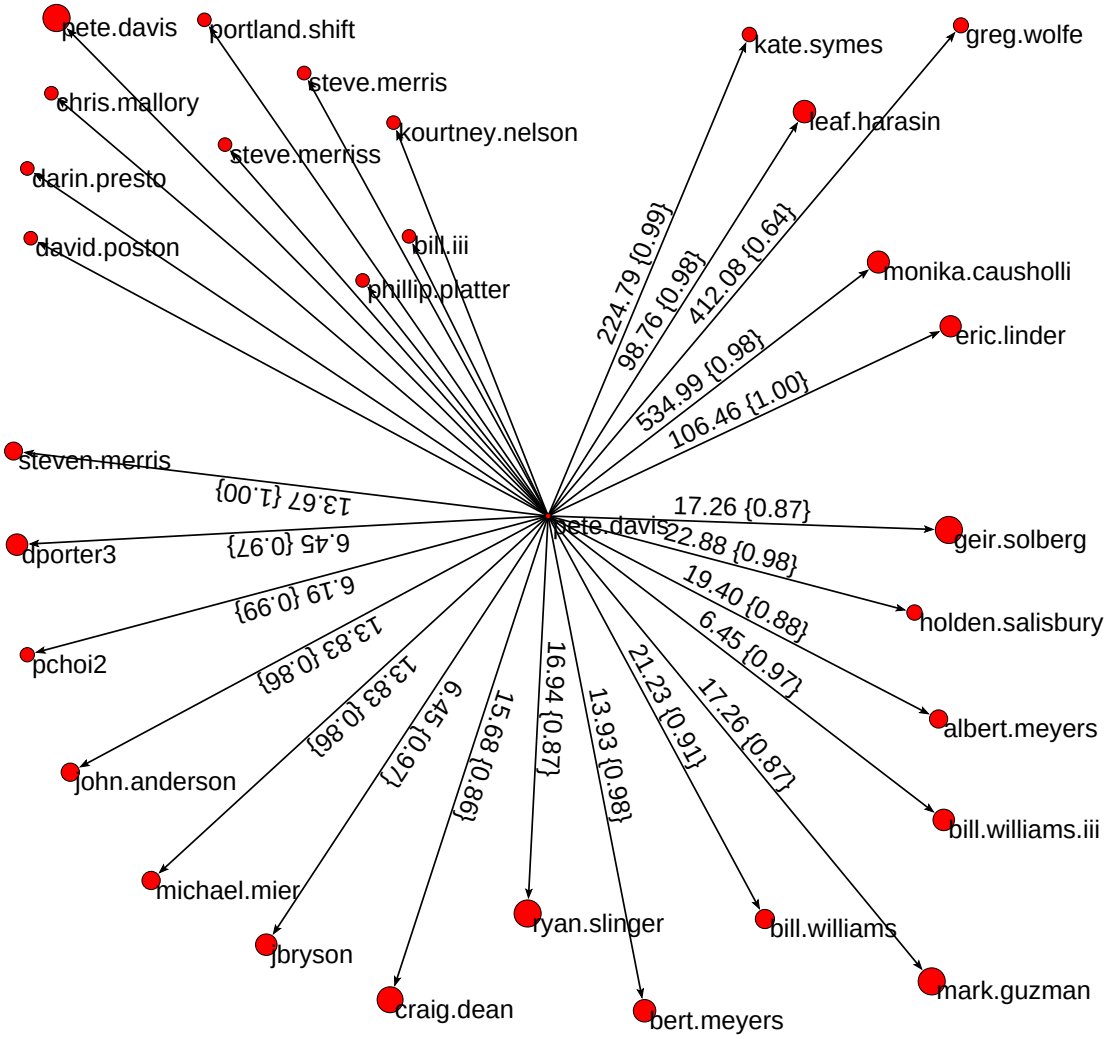


Figure 4.12: An egocentric e-mail network centered on `pete.davis@enron.com`, showing three possible communities. Edge labels are of the form $MAE(F_1)$ with MAE in minutes; vertex sizes are proportional to total number of e-mails sent.

This suggests that although digitally recorded social networks are massive, there exists within them an embedded backbone of predictable connections that might have special meaning.

Bibliography

[Acar et al., 2009] Acar, E., Dunlavy, D., and Kolda, T. (2009). Link prediction on evolving data using matrix and tensor factorizations. In *IEEE Intl. Conf. on Data Mining Wkshps.*, pages 262–269. IEEE.

[Agrawal and Srikant, 1994] Agrawal, R. and Srikant, R. (1994). Fast Algorithms for Mining Association Rules in Large Databases. In *Proc. of the 20th Intl. Conf. on Very Large Data Bases*, pages 487–499, San Francisco, CA. Morgan Kaufmann Publishers Inc.

[Brin et al., 1997] Brin, S., Motwani, R., Ullman, J., and Tsur, S. (1997). Dynamic itemset counting and implication rules for market basket data. In *Proc. of the 1997 ACM SIGMOD Intl. Conf. on Management of data*, pages 255–264. ACM Press New York, NY, USA.

[Bunke, 2003] Bunke, H. (2003). Graph-based tools for data mining and machine learning. *Lecture Notes in Computer Science*, pages 7–19.

[Bunke et al., 2005] Bunke, H., Dickinson, P., Irniger, C., and Kraetzl, M. (2005). Analysis of time series of graphs: Prediction of node presence by means of decision tree learning. In *Proc. of the 4th Intl. Conf. on Machine Learning and Data Mining in Pattern Recognition*, volume 3587, pages 366–375. Springer.

[Chakrabarti and Faloutsos, 2006] Chakrabarti, D. and Faloutsos, C. (2006). Graph mining: Laws, generators, and algorithms. *ACM Comput. Surv.*, 38(1):2.

[Chaovalitwongse et al., 2007] Chaovalitwongse, W. A., Fan, Y.-J., and Sachdeo, R. C. (2007). Support feature machine for classification of abnormal brain activity. In *Proc. of the 13th ACM SIGKDD Intl. Conf. on knowl. discovery and data mining*, pages 113–122, New York, NY, USA. ACM.

[Chatfield, 2004] Chatfield, C. (2004). *The analysis of time series: an introduction*. CRC press.

[Domeniconi et al., 2002] Domeniconi, C., Perng, C.-S., Vilalta, R., and Ma, S. (2002). A classification approach for prediction of target events in temporal sequences. In *Proc. of the 6th European Conf. on Principles of Data Mining and Knowl. Disc.*, pages 125–137, London, UK. Springer-Verlag.

[Faber et al., 2003] Faber, N., Bro, R., and Hopke, P. (2003). Recent developments in CANDECOMP/PARAFAC algorithms: a critical review. *Chemometrics and Intelligent Laboratory Systems*, 65(1):119–137.

[Fawcett, 2004] Fawcett, T. (2004). ROC graphs: Notes and practical considerations for researchers. *Machine Learning*, 31.

[Fawcett and Provost, 1999] Fawcett, T. and Provost, F. (1999). Activity monitoring: noticing interesting changes in behavior. In *Proc. of the 5th ACM SIGKDD Intl. Conf. on Knowledge discovery and data mining*, pages 53–62, New York, NY, USA. ACM.

[Fraley and Raftery, 2007] Fraley, C. and Raftery, A. E. (2007). Bayesian regularization for normal mixture estimation and model-based clustering. *J. Classif.*, 24(2):155–181.

[Frias-Martinez and Karamcheti, 2002] Frias-Martinez, E. and Karamcheti, V. (2002). A prediction model for user access sequences. In *Proc. WebKDD Wkshp.*

[Frias-Martinez and Karamcheti, 2003] Frias-Martinez, E. and Karamcheti, V. (2003). Reduction of user perceived latency for a dynamic and personalized site using web-mining techniques. In *Proc. WebKDD Wkshp.*

[Galil, 1986] Galil, Z. (1986). Efficient algorithms for finding maximum matching in graphs. *ACM Comput. Surv.*, 18(1):23–38.

[Geng and Hamilton, 2006] Geng, L. and Hamilton, H. (2006). Interestingness measures for data mining: A survey. *ACM Computing Surveys*, 38(3):9.

[Getoor et al., 2003] Getoor, L., Friedman, N., Koller, D., and Taskar, B. (2003). Learning probabilistic models of link structure. *Journal of Machine Learning Research*, 3:679–707.

[Getoor and Taskar, 2007] Getoor, L. and Taskar, B. (2007). *Introduction to Statistical Relational Learning*. MIT Press.

[Han et al., 1999] Han, J., Yin, Y., and Dong, G. (1999). Efficient mining of partial periodic patterns in time series database. In *Proc. of the 15th Intl. Conf. on Data Engineering*, pages 106–115, Los Alamitos, CA. IEEE Computer Society.

[Harms and Deogun, 2004] Harms, S. K. and Deogun, J. S. (2004). Sequential association rule mining with time lags. *Journal of Intelligent Information Systems*, 22(1):7–22.

[Huang and Lin, 2009] Huang, Z. and Lin, D. K. J. (2009). The time-series link prediction problem with applications in communication surveillance. *INFORMS Journal on Computing*, 21(2):286–303.

[Jensen and Neville, 2003] Jensen, D. and Neville, J. (2003). Data Mining in Social Networks. In *Dynamic Social Network Modeling and Analysis: Wkshp. Summary and Papers*. National Academy Press.

[Kannadiga et al., 2007] Kannadiga, P., Zulkernine, M., and Haque, A. (2007). E-NIPS: An event-based network intrusion prediction system. *LECTURE NOTES IN COMPUTER SCIENCE*, 4779:37.

[Kashima and Abe, 2006] Kashima, H. and Abe, N. (2006). A parameterized probabilistic model of network evolution for supervised link prediction. In *Proc. of the Sixth IEEE Intl. Conf. on Data Mining*, pages 340–349, Los Alamitos, CA, USA. IEEE Computer Society.

[Kossinets and Watts, 2006] Kossinets, G. and Watts, D. J. (2006). Empirical analysis of an evolving social network. *Science*, 311(5757):88–90.

[Lahiri and Berger-Wolf, 2007] Lahiri, M. and Berger-Wolf, T. Y. (2007). Structure prediction in temporal networks using frequent subgraphs. In *Proc. of IEEE Symposium on Computational Intelligence and Data Mining*, pages 35–42.

[Lahiri and Berger-Wolf, 2008] Lahiri, M. and Berger-Wolf, T. Y. (2008). Mining periodic behavior in dynamic social networks. In *Proc. of the IEEE Intl. Conf. on Data Mining*, pages 373–382.

[Laxman et al., 2005] Laxman, S., Sastry, P., and Unnikrishnan, K. (2005). Discovering frequent episodes and learning hidden markov models: A formal connection. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):1505–1517.

[Laxman et al., 2008] Laxman, S., Tankasali, V., and White, R. W. (2008). Stream prediction using a generative model based on frequent episodes in event sequences. In *KDD '08: Proceeding of the 14th ACM SIGKDD Intl. Conf. on Knowledge discovery and data mining*, pages 453–461, New York, NY, USA. ACM.

[Lehot, 1974] Lehot, P. G. H. (1974). An optimal algorithm to detect a line graph and output its root graph. *Journal of the ACM*, 21(4):569–575.

[Lesh et al., 1999] Lesh, N., Zaki, M. J., and Ogihara, M. (1999). Mining features for sequence classification. In *KDD '99: Proc. of the fifth ACM SIGKDD Intl. Conf. on Knowledge discovery and data mining*, pages 342–346, New York, NY, USA. ACM.

[Leskovec and Horvitz, 2008] Leskovec, J. and Horvitz, E. (2008). Planetary-scale views on a large instant-messaging network. In *Proceeding of the 17th Intl. Conf. on World Wide Web*, pages 915–924, New York, NY, USA. ACM.

[Liang et al., 2006] Liang, Y., Zhang, Y., Sivasubramaniam, A., Jette, M., and Sahoo, R. (2006). Bluegene/l failure analysis and prediction models. *Intl. Conf. on Dependable Systems and Networks*, 0:425–434.

[Liben-Nowell and Kleinberg, 2003] Liben-Nowell, D. and Kleinberg, J. M. (2003). The link prediction problem for social networks. In *Proc. of the twelfth Intl. Conf. on Information and knowledge management*, pages 556–559, New York, NY, USA. ACM.

[Liu et al., 1998] Liu, B., Hsu, W., and Ma, Y. (1998). Integrating classification and association rule mining. In *Proc. of the 4rd Intl. Conf. Knowledge Discovery and Data Mining (KDD-98)*, pages 80–86. AAAI Press.

[Malmgren et al., 2009] Malmgren, R. D., Hofman, J. M., Amaral, L. A., and Watts, D. J. (2009). Characterizing individual communication patterns. In *Proc. of the 15th ACM SIGKDD Intl. Conf. on Knowl. disc. and data mining*, pages 607–616.

[Mannila et al., 1997a] Mannila, H., Toivonen, H., and Inkeri Verkamo, A. (1997a). Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289.

[Mannila et al., 1997b] Mannila, H., Toivonen, H., and Verkamo, A. I. (1997b). Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289.

[Murata and Moriyasu, 2007] Murata, T. and Moriyasu, S. (2007). Link prediction of social networks based on weighted proximity measures. In *Proc. of the IEEE/WIC/ACM Intl. Conf. on Web Intelligence*, pages 85–88, Washington, DC, USA. IEEE Computer Society.

[Nanavati et al., 2006] Nanavati, A. A., Gurumurthy, S., Das, G., Chakraborty, D., Dasgupta, K., Mukherjea, S., and Joshi, A. (2006). On the structural properties of massive telecom call graphs: findings and implications. In *Proc. of the 15th ACM Intl. Conf. on Information and knowledge management*, pages 435–444, New York, NY, USA. ACM.

[Newman, 2003] Newman, M. E. J. (2003). The structure and function of complex networks. *SIAM Review*, 45(2):167–256.

[Oates and Cohen, 1996] Oates, T. and Cohen, P. R. (1996). Searching for structure in multiple streams of data. In *In Proc. of the Thirteenth Intl. Conf. on Machine Learning*, pages 346–354. Morgan Kaufmann.

[Oates et al., 1997] Oates, T., Schmill, M., Jensen, D., and Cohen, P. (1997). A family of algorithms for finding temporal structure in data. In *6th Intl. Wkshp. on A.I. and Statistics*.

[O'Madadhain et al., 2005] O'Madadhain, J., Hutchins, J., and Smyth, P. (2005). Prediction and ranking algorithms for event-based network data. *ACM SIGKDD Explorations Newsletter*, 7(2):23–30.

[Pei et al., 2004] Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., and Hsu, M.-C. (2004). Mining sequential patterns by pattern-growth: The PrefixSpan approach. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1424–1440.

[Phithakkitnukoon and Dantu, 2007] Phithakkitnukoon, S. and Dantu, R. (2007). Predicting calls–new service for an intelligent phone. In *Real-Time Mobile Multimedia Services*, volume 4787 of *LNCS*, pages 26–37. Springer.

[Pincombe, 2005] Pincombe, B. (2005). Anomaly detection in time series of graphs using ARMA processes. *ASOR Bulletin*, 24(4):2–10.

[Popescul and Ungar, 2003] Popescul, A. and Ungar, L. H. (2003). Statistical relational learning for link prediction. In *IJCAI03 Wkshp. on Learning Statistical Models from Relational Data*.

[Rabiner, 1989] Rabiner, L. R. (1989). A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proc. of the IEEE*, 77:257–286.

[Raftery, 1999] Raftery, A. (1999). Bayes factors and BIC. *Sociological Methods & Research*, 27(3):411–417.

[Salfner and Malek, 2007] Salfner, F. and Malek, M. (2007). Using hidden semi-markov models for effective online failure prediction. pages 161–174, Los Alamitos, CA, USA. IEEE Computer Society.

[Tung et al., 1999] Tung, A. K., Lu, H., Han, J., and Feng, L. (1999). Breaking the barrier of transactions: mining inter-transaction association rules. In *Proc. of the 5th ACM SIGKDD Intl. Conf. on Knowl. disc. and data mining*, pages 297–301, New York, NY, USA. ACM.

[Vaz de Melo et al., 2010] Vaz de Melo, P., Akoglu, L., Faloutsos, C., and Loureiro, A. (2010). Surprising patterns for the call duration distribution of mobile phone users. *Machine Learning and Knowledge Discovery in Databases*, pages 354–369.

[Vilalta and Ma, 2002] Vilalta, R. and Ma, S. (2002). Predicting rare events in temporal domains. *2nd IEEE Intl. Conf. on Data Mining*, page 474.

[Wang et al., 2007] Wang, C., Satuluri, V., and Parthasarathy, S. (2007). Local probabilistic models for link prediction. In *Proc. of the Seventh IEEE Intl. Conf. on Data Mining*, pages 322–331.

[Wasserman and Faust, 1994] Wasserman, S. and Faust, K. (1994). *Social Network Analysis: Methods and Applications*. Cambridge University Press.

[Wei and Chiu, 2002] Wei, C. and Chiu, I. (2002). Turning telecommunications call details to churn prediction: a data mining approach. *Expert systems with applications*, 23(2):103–112.

[West, 2001] West, D. B. (2001). *Introduction to graph theory*. Prentice Hall, NJ.