# Link Prediction in Dynamic Networks Using Graphlet

Mahmudur Rahman and Mohammad Al Hasan(✉)

Indiana University Purdue University Indianapolis, Indianapolis, USA
{mmrahman,alhasan}@iupui.edu

**Abstract.** Predicting the link state of a network at a future time given a collection of link states at earlier time is an important task with many real-life applications. In existing literature this task is known as link prediction in dynamic networks. Solving this task is more difficult than its counterpart in static networks because an effective feature representation of node-pair instances for the case of dynamic network is hard to obtain. In this work we solve this problem by designing a novel graphlet transition based feature representation of the node-pair instances of a dynamic network. We propose a method GraTFEL which uses unsupervised feature learning methodologies on graphlet transition based features to give a low-dimensional feature representation of the node-pair instances. GraTFEL models the feature learning task as an optimal coding task where the objective is to minimize the reconstruction error, and it solves this optimization task by using a gradient descent method. We validate the effectiveness of the learned feature representations by utilizing it for link prediction in real-life dynamic networks. Specifically, we show that GraTFEL, which use the extracted feature representation of graphlet transition events, outperforms existing methods that use well-known link prediction features. The data and software related to this paper are available at https://github.com/DMGroup-IUPUI/GraTFEL-Source.

## 1 Introduction

Understanding the dynamics of an evolving network is an important research problem with numerous applications in various fields, including social network analysis, information retrieval, recommendation systems, epidemiology, security, and bioinformatics. A key task towards this understanding is to predict the likelihood of a future association between a pair of nodes, given the existing state of the network. This task is commonly known as the *link prediction* problem. Since, its formal introduction to the data mining community by Liben-Nowell et al. [9] about a decade ago, this problem has been studied extensively by many researchers from a diverse set of disciplines. Comprehensive surveys on link prediction methods are available for interested readers [7].

This research is supported by Mohammad Hasan's NSF CAREER Award (IIS-1149851).


P. Frasconi et al. (Eds.): ECML PKDD 2016, Part I, LNAI 9851, pp. 394–409, 2016.
DOI: 10.1007/978-3-319-46128-1_25

The majority of the existing works on link prediction consider a static snapshot of the given network, which is the state of the network at a given time [6,9–11]. Nevertheless, for many networks, additional temporal information, such as the time of link creation and deletion, is available over a time interval. For example, in an online social or a professional network, we may know the time when two persons have become friends; for collaboration events, such as, a group performance or a collaborative academic work, we can extract the time of the event from an event calendar. The networks built from such data can be represented by a *dynamic network*, which is a collection of temporal snapshots of the network. The link prediction task on such a network is defined as follows: *for a given pair of nodes, predict the link probability between the pair at time* $t+1$ *by training the model on the link information at times* $1, 2, \cdots, t$. We will refer this task as dynamic link prediction[1].

A key challenge of dynamic link prediction is finding a suitable feature representation of the node-pair instances which are used for training the prediction model. For the static setting, various topological metrics (common neighbors, Adamic-Adar, Jaccard's coefficient) are used as features, but they cannot be extended easily for the dynamic setting having multiple snapshots of the network. In fact, when multiple (say $t$) temporal snapshots of a network are provided, each of these scalar features becomes a $t$-sized sequence. Flattening the sequence into a $t$-size vector distorts the inherent temporal order of the features. Authors of [5] overcome this issue by modeling a collection of time series, each for one of the topological features; but such a model fails to capture signals from the neighborhood topology of the edges. There exist a few other works on dynamic link prediction, which use probabilistic (nonparametric) and matrix factorization based models. These works consider a feature representation of the nodes and assume that having a link from one node to another is determined by the combined effect of all pairwise node feature interactions [4,18,22]. While this is a reasonable assumption, the accuracy of such models are highly dependent on the quality of the node features, as well as the validity of the above assumption.

Graphlets, which are collection of small induced subgraphs, are increasingly being used for large-scale graph analysis. For example, frequencies of various graphlets are used for classifying networks from various domains [17]. They are also used for designing effective graph kernels [19]. In biological domain, graphlet frequencies are used for comparing structures of different biological networks [15]. In all these works, graphlets are used as a topological building block of a static network. Nevertheless, as new edges are added or existing edges are removed from the given dynamic network, the graphlets which are aligned with the affected edge transition to different graphlets. For illustration, let us consider a dynamic network with two temporal snapshots $G_1$ and $G_2$ (Fig. 1). In this example, $G_2$ has one more edge $(2, 3)$ than $G_1$. We observe three different types of transition events, where a type of graphlet is changed into another type in the subsequent snapshot (see the table in Fig. 1). Here, all the events are triggered by the edge

[1] Strictly speaking, this task should be called link forecasting as the learning model is not trained on partial observation of link instances at time $t+1$; however, we refer it as link prediction due to the popular usages of this phrase in the data mining literature.

(2, 3). In this work, we use the frequency of graphlet transition events associated with a node-pair for predicting link between the node-pairs in a future snapshot of the dynamic network.

A key challenge of using graphlet transition event for dynamic link prediction is to obtain a good feature representation for this task. This is necessary because graphlet transition event matrix is sparse, and on such dataset, low dimensional feature representation effectively captures the latent dependency among different dimensions of the data. There exist a growing list of recent works which use unsupervised methodologies for finding features from raw data representations of various complex objects, including images [13] and audio [8]. For graph data, we are aware of only one such work, namely DeepWalk [14], which obtains the feature representation of nodes for solving a node classification task. However, no such work exists for finding feature representation of node-pair instances for the purpose of link prediction.

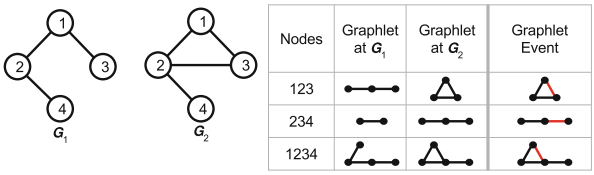



**Fig. 1.** A Toy Dynamic Network. $G_1$ and $G_2$ are two snapshots of the Network. Three different types of graphlet events are observed.

In this work, we propose a novel learning method GraTFEL (**Gra**phlet **T**ransition and **F**eature **E**xtraction for **L**ink Prediction) for obtaining feature representation of node-pair instances from graphlet transition events in the observed snapshots of the given network. GraTFEL considers the feature learning task as an optimal coding problem such that the optimal code of a node-pair is the desired feature representation. The learning can be considered as a two-step process (compression and reconstruction), where the first step compresses the input representation of a node-pair into a code by a non-linear transformation, and the second step reconstructs the input representation from the code by a reverse process and the optimal code is the one which yields the least amount of reconstruction error. The input representation of a node-pair is given as a vector of graphlet transition events (GTEs) associated with the corresponding node-pair. After obtaining an appropriate feature representation of the node-pairs, a traditional supervised learning technique is used (we use SVM and AdaBoost) for predicting link states at future times in the given dynamic network. Below we summarize our contributions in this work:

- We use graphlet transition events (GTEs) for preforming link prediction in a dynamic network. To the best of our knowledge we are the first to use GTEs for solving a prediction task over a dynamic network.
- We propose a learning model (GraTFEL) for unsupervised feature extraction of node-pairs for the purpose of link prediction over a dynamic network.
- We compare the performance of GraTFEL with multiple state-of-the-art methods on three real-life dynamic networks. This comparison results show that our method is superior than each of the competing methods.

## 2 Related Works

Graphlets have been used successfully in static network setup for a multitude of applications. For a given network, Pržulj et al. [16] count the frequencies of various graphlets in the network for designing a fingerprint of a biological graph. In [12], the authors define *signature similarity* function to characterize the similarity of two vertices of a network, thus allowing a user to cluster vertices based on their structural similarity. Sampling based methods are used for computing graphlet degree distribution efficiently [1,17]. While the above methods address graphlet based analysis of static networks, works, exploring dynamic network characterization using graphlet, are yet to come.

A multitude of methodologies have been developed for link prediction on dynamic networks. The methods proposed by Güneş et al. [5] captures temporal patterns in a dynamic network using a collection of time series on topological features. But this approach fails to capture signals from neighborhood topology, as each time series model is trained on a separate $t$-size feature sequence of a node-pair. The method proposed by Bliss et al. [2] applies covariance matrix adaptation evolution strategy (CMA-ES) to optimize weights which are used in a linear combination of sixteen neighborhood and node similarity indices. Matrix and tensor factorization based solutions are presented in [4], considering a tensor representation of a dynamic network. The nonparametric link prediction method presented in [18] uses features of the node-pairs, as well as the local neighborhood of node-pairs. This method works by choosing a probabilistic model based on features (common neighbor and last time of linkage) of node-pairs. Stochastic block model based approaches divide nodes in a network into several groups and generates edges with probabilities dependent on the group membership of participant nodes [22]. While probabilistic model based link prediction performs well on small graphs, they become computationally prohibitive for large networks.

## 3 Problem Definition

Assume $G(V, E)$ is an undirected network, where $V$ is the set of nodes and $E$ is the set of edges $e(u, v)$ such that $u, v \in V$. A dynamic network is represented as a sequence of snapshots $\mathbb{G} = \{G_1, G_2, \ldots, G_t\}$, where $t$ is the number of time stamps for which we have graph snapshots and $G_i(V_i, E_i)$ is a graph snapshot at time stamp $i : 1 \leq i \leq t$. In this work, we assume that the vertex set remains the same across different snapshots, i.e., $V_1 = V_2 = \cdots = V_t = V$. However, the edges appear and disappear over different time stamps. We also assume that, beside the link information, no other attribute data for the nodes or edges are available. We use $n$ to denote the number of nodes ($|V|$), and $m$ to denote all node-pairs $\binom{|V|}{2}$ in the network.

**Problem Statement:** Given a sequence of snapshots $\mathbb{G} = \{G_1, G_2, \ldots, G_t\}$ of a network, and a pair of nodes $u$ and $v$, the link prediction task on a dynamic network predicts whether $u$ and $v$ will have a link in $G_{t+1}$. Note that, we assume that no link information regarding the snapshot $G_{t+1}$ is available, except the fact that $G_{t+1}$ contains the identical set of vertices.

## 4 Methods

A key challenge for dynamic link prediction is choosing an effective feature set for this task. Earlier works choose features by adapting topological features for static link prediction or by considering the feature values of different snapshots as a time series. GraTFEL uses graphlet transition events (GTEs) as features for link prediction. For a given node-pair, the value of a specific GTE feature is a normalized count of the observed GTE involving those node-pairs over the training data. The strength of GTEs as feature for dynamic link prediction comes from the fact that for a given node-pair, GTEs involving those nodes capture both the local topology and their transition over the temporal snapshots.

We consider GTEs involving graphlets up to size five (total 30 graphlets), of which graphlets of size four are shown in Fig. 2[2]. The graphlet size upper bound of five is inspired by the fact that more than 95 % new links in a dynamic network happen between vertices that are at most 3 distances apart in all three real-life dynamic networks that we use in this work. So, for a given node, GTE of a five vertex graphlet in the neighborhood of that node covers a prospective link formation event as a graphlet transition event. Another reason for limiting the graphlet size is the consideration of computation burden, which increases exponentially with the size of graphlets. There are 30 different graphlets of size up to 5 and the number of possible transition event (GTE) is $O(30^2)$. Increasing the size of graphlets to 6 increases the number of GTE to $O(142^2)$.

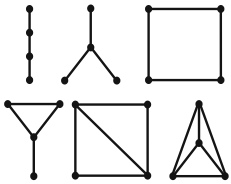


**Fig. 2.** Graphlets of size 4.

Feature representation for a node-pair in a dynamic network is constructed by concatenating GTE features from a continuous set of graph snapshots. Concatenation, being the simplest form of feature aggregation across a set of graph snapshots is not essentially the best feature representation to capture temporal characteristics of a node-pair. So, GraTFEL uses unsupervised feature extraction (UFE) to get optimal feature representation from GTE features. UFE provides a better feature representation by discovering dependency among different data dimensions, which cannot be achieved by simple aggregation. It also reduces the data dimension and overcomes the sparsity issue in GTE features. Once the optimal feature representation of a node-pair is known, GraTFEL uses that for solving the link prediction task using a supervised classification model.

The discussion of the proposed method GraTFEL can be divided into three steps: (1) graphlet transition event based feature extraction (Sect. 4.1), (2) unsupervised (optimal) feature learning (Sect. 4.2), and (3) supervised learning for obtaining the link prediction model (Sect. 4.3).

[2] There are only one graphlet of size 2, two graphlets of size 3 and twenty-one graphlets of size 5. These graphlets are not shown due to space constraint.

### 4.1 Graphlet Transition Based Feature Extraction

Say, we are given a dynamic network $\mathbb{G} = \{G_1, G_2, \ldots, G_t\}$, and we are computing the feature vector for a node-pair $(u, v)$, which constitute a row in our training data matrix. We use each of $G_i : 1 \leq i \leq t-1$ (time window $[1, t-1]$) for computing the feature vector and $G_t$ for computing the target value (1 if edge exist between $u$ and $v$, 0 otherwise). We use $\boldsymbol{e}^{uv}_{[1,t-1]}$ to represent this vector. It has two components: graphlet transition event (GTE) and link history (LH).

The first component, Graphlet Transition Event (GTE), $\boldsymbol{g}^{uv}_{[1,t-1]}$ is constructed by concatenating GTE feature-set of $(u, v)$ for each time stamp. i.e., $\boldsymbol{g}^{uv}_{[1,t-1]} = \boldsymbol{g}^{uv}_1 \mid\mid \boldsymbol{g}^{uv}_2 \mid\mid \ldots \mid\mid \boldsymbol{g}^{uv}_{t-1}$. Here, the symbol $\mid\mid$ represents concatenation of two horizontal vectors (e.g., 0 1 0 $\mid\mid$ 0.5 0 1 = 0 1 0 0.5 0 1) and $\boldsymbol{g}^{uv}_i$ represents $(u, v)$'s GTE feature-set for time stamp $i$, and it captures the impact of edge $(u, v)$ at its neighborhood structure at time stamp $i$. We construct $\boldsymbol{g}^{uv}_i$ by enumerating all graphlet based dynamic events, that are triggered when edge $(u, v)$ is added with $G_i$.

For example, consider the toy dynamic network in Fig. 3. We want to construct the GTE feature vector $\boldsymbol{g}^{36}_1$, which is the GTE feature representation of node-pair $(3, 6)$ at $G_1$. We illustrate the construction process in Fig. 4. In this figure, we show all the graphlet transitions triggered by edge $(3, 6)$ when it is added in $G_1$. These transition events are listed in center table of Fig. 4. Column titled *Nodes* lists the sets of nodes where the graphlet transitions are observed and Column *Current Graphlet* shows the current graphlet structure induced by these nodes. Column *Transformed Graphlet* shows the graphlet structure after $(3, 6)$ is added. The last column *Graphlet Event* is a visual representation of the transition events, where the transition is reflected by the red edges. Once all the transition events are enumerated, we count the frequencies of these events (Table on the right side of Fig. 4). Graphlet transition frequencies can be substantially different for different edges, so the GTE vector is normalized by the largest value of graphlet transition frequencies associated with this edge. Also note that, all possible graphlet transition events are not observed for a given edge. So, among all the possible types of GTE, those that are observed in at least one node-pair in the training dataset are considered in GTE feature-set.

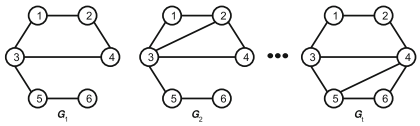



**Fig. 3.** A Toy Dynamic Network with $t$ snapshots. First two and last snapshots are shown in this figure.

The second component of node-pair feature vector is Link History (LH) of node-pair, which is not captured by GTE feature-set, $\boldsymbol{g}^{uv}_{[1,t-1]}$. Link History, $\mathbf{lh}^{uv}_{[1,t-1]}$ of a node-pair $(u, v)$ is a vector of size $t-1$, denoting the edge occurrences between the participating nodes over the time window $[1, t-1]$. It is defined as, $\mathbf{lh}^{uv}_{[1,t-1]} = \mathbf{G}_1(u, v) \mid\mid \mathbf{G}_2(u, v) \mid\mid \ldots \mid\mid \mathbf{G}_{t-1}(u, v)$. Here, $\mathbf{G}_i(u, v)$ is 1 if edge $(u, v)$ is present in snapshot $G_i$ and 0 otherwise.

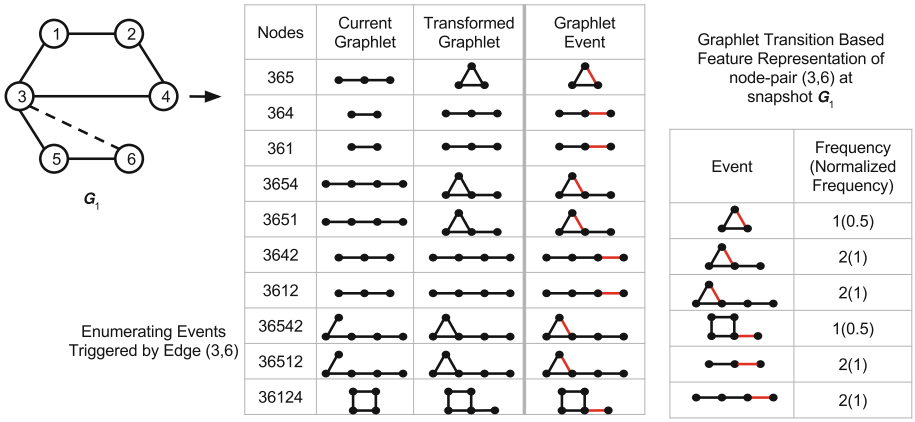



**Fig. 4.** Construction of graphlet transition based feature representation $\boldsymbol{g}_1^{36}$ of node-pair $(3, 6)$ at $1^{st}$ snapshot of the toy network. (Color figure online)

An appearance of an edge in recent time indicates a higher chance of the edge to reappear in near future. So, we consider weighted link history, $\mathbf{wlh}^{uv}_{[1,t-1]} = w_1 \cdot \mathbf{G}_1(u,v) \;||\; w_2 \cdot \mathbf{G}_2(u,v) \;||\; \ldots \;||\; w_{t-1} \cdot \mathbf{G}_{t-1}(u,v)$. here, $w_i = i/(t-1)$ (a time decay function) represents the weight of link history for time stamp $i$. Finally, a frequent appearance of an edge over time indicates a strong tendency of the edge to reincarnate in the future. This motivates us to reward such events by considering cumulative sum. We define Weighted Cumulative Link History, $\mathbf{wclh}^{uv}_{[1,t-1]} = CumSum(\mathbf{wlh}^{uv}_{[1,t-1]})$.

Finally, the feature vector of a node-pair $(u, v)$, $\mathbf{e}^{uv}_{[1,t-1]}$, is the concatenation of GTE feature-set $(\boldsymbol{g}^{uv}_{[1,t-1]})$ and LH feature-set $(\mathbf{wclh}^{uv}_{[1,t-1]})$; i.e., $\mathbf{e}^{uv}_{[1,t-1]} = \boldsymbol{g}^{uv}_{[1,t-1]} \;||\; \mathbf{wclh}^{uv}_{[1,t-1]}$. For predicting dynamic links in time stamp $t+1$, we right-shift the time window by one. In other words, we construct graphlet feature representation $\mathbf{e}^{uv}_{[2,t]}$ by using snapshots from time window $[2, t]$. Final feature representation for all node-pairs,

$$\begin{aligned} \hat{\mathbf{E}} &= \{\mathbf{e}^{uv}_{[1,t-1]}\}_{u,v \in V} \\ \bar{\mathbf{E}} &= \{\mathbf{e}^{uv}_{[2,t]}\}_{u,v \in V} \end{aligned} \tag{1}$$

Here, $\hat{\mathbf{E}}$ is the training dataset and $\bar{\mathbf{E}}$ is the prediction dataset. Both $\hat{\mathbf{E}}$ and $\bar{\mathbf{E}}$ can be represented as matrices of dimensions $(m,k)$. The size of the feature vector is $k = |\mathbf{e}^{uv}_{[1,t-1]}| = c*(t-1)+t-1$, where $c$ is the total number of distinct GTEs that we consider.

**GTE enumeration.** We compute GTEs by using a local growth algorithm. For computing $\boldsymbol{g}^{uv}_i$, we first enumerate all graphlets of $G_i$ having both $u$ and $v$. Starting from the edge graphlet $gl = \{u, v\}$, in each iteration of growth we add a new vertex $w$ from the immediate neighborhood of $gl$ to obtain a larger graphlet

$gl = gl \cup \{w\}$. Growth is terminated when $|gl| = 5$. The enumeration process is identical to our earlier work [17]. After enumeration, GTE is easily obtained from graphlet embedding by marking the edge $(u, v)$ as the transition trigger (see Fig. 4). The computation of GTEs of different node-pairs are not dependent on each other, this makes GTE enumeration task embarrassingly parallel.

### 4.2 Unsupervised Feature Extraction

GraTFEL performs the task of unsupervised feature extraction as learning an optimal coding function $h$. Lets consider, $\boldsymbol{e}$ is a feature vector from either $\hat{\mathbf{E}}$ or $\bar{\mathbf{E}}$ ($\boldsymbol{e} \in \hat{\mathbf{E}} \cup \bar{\mathbf{E}}$). Now, the coding function $h$ compresses $\boldsymbol{e}$ to a code vector $\boldsymbol{\alpha}$ of dimension $l$, such that $l < k$. Here $l$ is a user-defined parameter which represents the code length and $k$ is the size of feature vector. Many different coding functions exist in the dimensionality reduction literature, but GraTFEL chooses the coding function which incurs the minimum loss in the sense that from the code $\boldsymbol{\alpha}$ we can reconstruct $\boldsymbol{e}$ with the minimum error over all possible $\boldsymbol{e} \in \hat{\mathbf{E}} \cup \bar{\mathbf{E}}$. We frame the learning of $h$ as an optimization problem, which we discuss below through two operations: Compression and Reconstruction.

**Compression:** It obtains $\boldsymbol{\alpha}$ from $\boldsymbol{e}$. This transformation can be expressed as a nonlinear function of linear weighted sum of the graphlet transition features.

$$\boldsymbol{\alpha} = f(\boldsymbol{W}^{(c)}\boldsymbol{e} + \boldsymbol{b}^{(c)}) \tag{2}$$

$\boldsymbol{W}^{(c)}$ is a $(k, l)$ dimensional matrix. It represents the weight matrix for compression and $\boldsymbol{b}^{(c)}$ represents biases. $f(\cdot)$ is the Sigmoid function, $f(x) = \frac{1}{1+e^{-x}}$.

**Reconstruction:** It performs the reverse operation of compression, i.e., it obtains the graphlet transition features $\mathbf{e}$ from $\boldsymbol{\alpha}$ which was constructed during the compression operation.

$$\boldsymbol{\beta} = f(\boldsymbol{W}^{(r)}\boldsymbol{\alpha} + \boldsymbol{b}^{(r)}) \tag{3}$$

$\boldsymbol{W}^{(r)}$ is a matrix of dimensions $(l, k)$ representing the weight matrix for reconstruction, and $\boldsymbol{b}^{(r)}$ represents biases.

The optimal coding function $h$ constituted by the compression and reconstruction operations is defined by the parameters $(\boldsymbol{W}, \boldsymbol{b}) = (\boldsymbol{W}^{(c)}, \boldsymbol{b}^{(c)}, \boldsymbol{W}^{(r)}, \boldsymbol{b}^{(r)})$. The objective is to minimize the reconstruction error. Reconstruction error for a graphlet transition feature vector $(e)$ is defined as, $J(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{e}) = \frac{1}{2} \parallel \boldsymbol{\beta} - \boldsymbol{e} \parallel^2$. Over all possible feature vectors, the average reconstruction error augmented with a regularization term yields the final objective function $J(\boldsymbol{W}, \boldsymbol{b})$ which we want to minimize:

$$J(\boldsymbol{W}, \boldsymbol{b}) = \frac{1}{2m} \sum_{\boldsymbol{e} \in \hat{\mathbf{E}} \cup \bar{\mathbf{E}}} (\frac{1}{2} \parallel \boldsymbol{\beta} - \boldsymbol{e} \parallel^2) + \frac{\lambda}{2} (\parallel \boldsymbol{W}^{(c)} \parallel_F^2 + \parallel \boldsymbol{W}^{(r)} \parallel_F^2) \tag{4}$$

Here, $\lambda$ is a user assigned regularization parameter, responsible for preventing over-fitting. $\| \cdot \|_F$ represents the Frobenius norm of a matrix.

**Training:** The training of optimal coding defined by parameters $(\boldsymbol{W}, \boldsymbol{b})$ begins with random initialization of the parameters. Since the cost function $J(\boldsymbol{W}, \boldsymbol{b})$ defined in Eq. (4) is non-convex in nature, we obtain a local optimal solution using the gradient descent approach. The parameter update of the gradient descent is similar to the parameter update in Auto-encoder in machine learning. Unsupervised feature representation of any node-pair $(u, v)$ can be obtained by taking the outputs of compression stage (Eq. (2)) of the trained optimal coding $(\boldsymbol{W}, \boldsymbol{b})$.

$$\boldsymbol{\alpha}^{uv}_{[1,t-1]} = f(\boldsymbol{W}^{(c)}\boldsymbol{e}^{uv}_{[1,t-1]} + \boldsymbol{b}^{(c)}) = h(\boldsymbol{e}^{uv}_{[1,t-1]})$$

$$\boldsymbol{\alpha}^{uv}_{[2,t]} = f(\boldsymbol{W}^{(c)}\boldsymbol{e}^{uv}_{[2,t]} + \boldsymbol{b}^{(c)}) = h(\boldsymbol{e}^{uv}_{[2,t]})$$

**Computational Cost:** We use Matlab implementation of optimization algorithm L-BFGS (Limited-memory Broyden-Fletcher-Goldfarb-Shanno) for learning optimal coding. Non-convex nature of cost function allows us to converge to local optima. We execute the algorithm for limited number of iterations to obtain unsupervised features within a reasonable period of time. Each iteration of L-BFGS executes two tasks for each edge: back-propagation to compute partial differentiation of cost function, change the parameters $\boldsymbol{(W, b)}$. For each edge the time complexity is $O(kl)$; here, $k$ is the length of basic feature representation, $l$ is length of unsupervised feature representation. Therefore, the time complexity of one iteration is $O(mkl)$, where $m$ is the number of possible edges.

### 4.3 Supervised Link Prediction Model

Training dataset, $\hat{\mathbf{E}}$ is feature representation for time snapshots $[1, t-1]$, The ground truth $(\hat{\boldsymbol{y}})$ is constructed from $G_t$. After training the supervised classification model using $\hat{\boldsymbol{\alpha}} = h(\hat{\mathbf{E}})$ and $\hat{\boldsymbol{y}}$, prediction dataset $\bar{\mathbf{E}}$ is used to predict links at $G_{t+1}$. For this supervised prediction task, we experiment with several classification algorithms. Among them SVM (support vector machine) and AdaBoost perform the best.

### 4.4 Algorithm

The pseudo-code of GraTFEL is given in Algorithm 1. For training link prediction model, we split the available network snapshots into two overlapping time windows, $[1, t-1]$ and $[2, t]$. GTE features $\hat{\boldsymbol{E}}$ and $\bar{\boldsymbol{E}}$ are constructed in Lines 2 and 4, respectively. Then we learn optimal coding for node-pairs using graphlet transition features $(\hat{\mathbf{E}} \cup \bar{\mathbf{E}})$ in Line 5. Unsupervised feature representations are constructed using learned optimal coding (Lines 6 and 7). Finally, a classification model $C$ is learned (Line 8), which is used for predicting link in $G_{t+1}$(Line 9).

**Algorithm 1.** GRATFEL

**procedure** GRATFEL($\mathbb{G}, t$)
  **Input :** $\mathbb{G}$: Dynamic Graph, $t$: Time steps
  **Output:** $\bar{\mathbf{y}}$: Forecasted links at time step $t+1$
  $\hat{\mathbf{E}}$=GraphletTransitionFeature($\mathbb{G}$,1,$t-1$)
  $\hat{\mathbf{y}}$=Connectivity($G_t$)
  $\bar{\mathbf{E}}$=GraphletTransitionFeature($\mathbb{G}$,2,$t$)
  $h$=LearningOptimalCoding($\hat{\mathbf{E}} \cup \bar{\mathbf{E}}$)
  $\hat{\boldsymbol{\alpha}}$=$h(\hat{\mathbf{E}})$
  $\bar{\boldsymbol{\alpha}}$=$h(\bar{\mathbf{E}})$
  $C$=TrainClassifier($\hat{\boldsymbol{\alpha}}, \hat{\mathbf{y}}$)
  $\bar{\mathbf{y}}$=LinkForecasting($C, \bar{\boldsymbol{\alpha}}$)
  **return** $\bar{\mathbf{y}}$
**end procedure**

## 5 Experimental Results

### 5.1 Dataset

We use three real world dynamic networks for evaluating the performance of GRATFEL. These datasets are Enron, Collaboration and Facebook. The Enron email corpus consists of email exchanges between 184 Enron employees; the vertices of the network are employees, and the edges are email events between a pair of employees. The dataset has 11 temporal snapshots which are prepared identically as in [22]. The Collaboration dataset is constructed by using citation data from ArnetMiner (arnetminer.org). Each vertex in this dataset is an author and the edges represent co-authorship. We consider the data from years 2000-2009—total 10 snapshots considering each year as a time stamp. Many authors in the original dataset have published in only one or two years out of all ten years, so we select only those who participate in a minimum of 2 collaboration edges in at least 7 time stamps. The final dataset has 315 authors.

Lastly, the dynamic social network of Facebook wall posts [20] has 9 time stamps. We follow the same time-stamp setting and node filtering as [21], resulting in 9 time stamps of data, and 663 nodes. The dynamic link prediction task of all three datasets is to predict links on last snapshot, given the link information of all the previous snapshots. Statistics of the dynamic networks are given in Table 1. Although the number of vertices in these networks are in hundreds,

**Table 1.** Basic statistics of the datasets used.

| | Enron | Collaboration | Facebook |
|---|---|---|---|
| Time Snaps | 11 | 10 | 9 |
| Nodes | 184 | 315 | 663 |
| Avg. Edge | 217 | 255 | 1299 |
| Node−Pairs | 16, 836 | 49, 455 | 219, 453 |
| Avg. Density | .013 | .005 | .006 |

these are large datasets considering the possible node-pairs and multiple temporal snapshots.

### 5.2 Competing Methods

To compare the performance of GraTFEL, we choose link prediction methods from two categories: (1) topological feature based methods and (2) feature time series based methods [5].

For topological feature based method, we consider three leading topological features: common Neighbors ($CN$), Adamic-Adar ($AA$), and Jaccard's Coefficient ($J$). However, in existing works these features are defined for static network only; so we adapt these features for the dynamic network setting by taking the weighted average of the feature values at different time stamps, where the weight of the $t$'th time stamp is 1 and the weight of the $i$'th time stamp is $\frac{1}{t-i+1}$. The justification of such a weighting is due to the common belief that recent interaction is a good indicator of a repeat interaction. We also tried different feature weighting, but the above weighting gives the best performance for the topological feature based dynamic link prediction. We also combine the above three features to construct a combined feature vector of length 3 ($CNAAJ$) and use it with a classifier to build a supervised link prediction method, and include this model in our comparison.

We also compare GraTFEL with time series based neighborhood similarity scores proposed in [5]. In this work, the authors consider several neighborhood-based node similarity scores combined with connectivity information (historical edge information). Authors use time series of similarities to model the change of node similarities over time. Among 16 proposed methods, we consider 4 which are relevant to the link prediction task on unweighted networks, and also have the best performance. $TS$-$CN$-$Adj$ represents time series on normalized score of Common Neighbors and connectivity values at time stamps $[1, t]$. Similarly, we get time series based scores for Adamic-Adar ($TS$-$AA$-$Adj$), Jaccard's Coefficient ($TS$-$J$-$Adj$) and Preferential Attachment ($TS$-$PA$-$Adj$).

### 5.3 Implementation

For implementation of GraTFEL we use a combination of Python and Matlab. Graphlet transition is enumerated using a python implementation. Feature vector construction and unsupervised feature extraction are done using Matlab. The unsupervised feature extraction method runs for a maximum of 50 iterations or until it converges to a local optima. We use coding size $l = 200$ for all three datasets[3]. For link prediction we used several Matlab provided classification algorithms, namely AdaBoostM1, RobustBoost, and Support Vector Machine (SVM). We use Matlab for computing the feature values (CN, AA, J) that we

[3] We experiment with different coding sizes ranging from 100 to 800. The change in link prediction performance is not very sensitive to the coding size. At most 2.9 % change in PRAUC was observed for different coding sizes.

use in other competing methods. Time series methods are implemented using Python. We use the ARIMA (autoregressive integrated moving average) time series model implemented in Python module **statsmodels**. Datasets and source code are available at https://github.com/DMGroup-IUPUI/GraTFEL-Source.

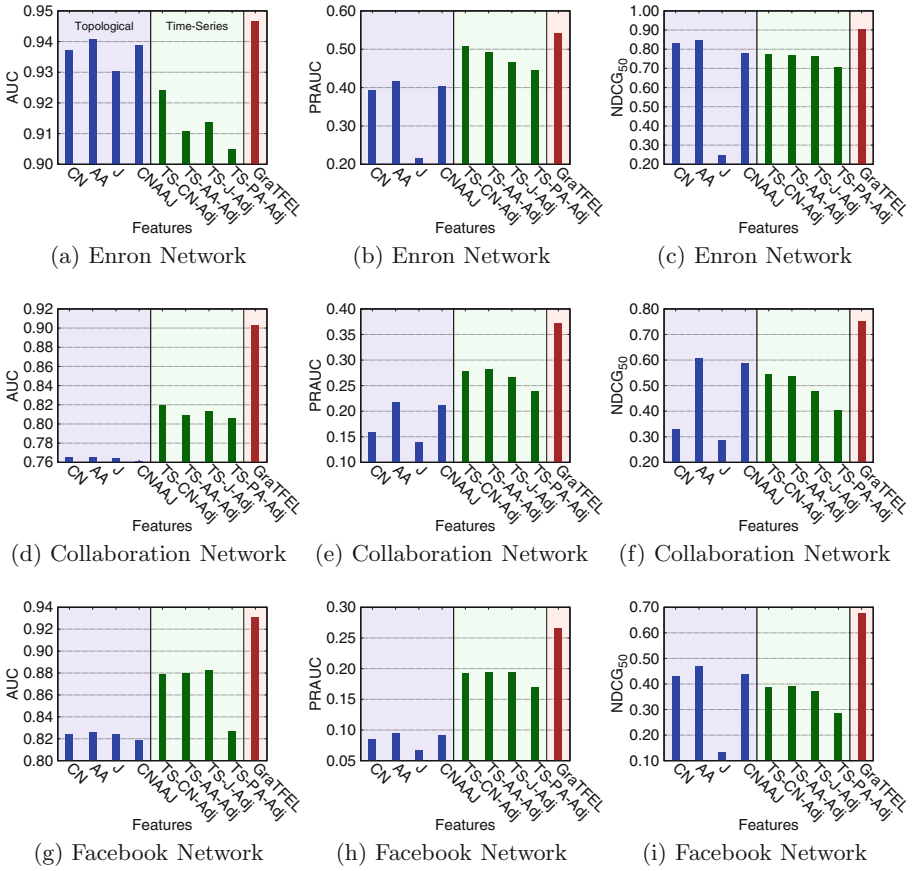
### 5.4 Evaluation Metrics

For evaluating the proposed methods we use three metrics; namely, area under Receiver Operator Characteristics curve (AUC), area under Precision-Recall curve (PRAUC), and Normalized Discounted Cumulative Gain (NDCG). The AUC value for a link prediction problem quantifies the probability that a randomly chosen edge is ranked higher than a randomly chosen node pairs without edge. However, real world datasets for link prediction are generally skewed; the number of edges ($|E|$) is very small compared to the number of possible node-pairs $\binom{|V|}{2}$ in the graph. In such scenarios, PRAUC gives a more informative picture of the algorithm's performance. The reason why PRAUC is more suitable for the skewed problem is that it does not factor in the effect of true negatives. In skewed data where the number of negative examples is huge compared to the number of positive examples, true negatives are not that meaningful. The last metric, NDCG, is an information retrieval metric which is widely used by the recommender systems community. NDCG measures the performance of link prediction system based on the graded relevance of the recommended links. $NDCG_p$ varies from 0.0 to 1.0, with 1.0 representing the ideal ranking of edges; $p$ is a user-defined parameter, which represents the number of links ranked by the method. We use $p = 50$ (unless stated otherwise). NDCG is suitable when it is important to return the ranked list of top $p$ predicted links.

### 5.5 Performance Comparison with Competing Methods

In this section, we present performance comparison of GraTFEL with several competing methods (see Fig. 5). The bar charts in the top, middle, and bottom rows of Fig. 5 display the results for Enron, Collaboration, and Facebook datasets, respectively. The bar charts in a row show comparison results using AUC, PRAUC, and $NDCG_{50}$ (from left to right). In total, there are 9 bars in a chart, each representing a link prediction method, where the height of a bar is indicative of the performance metric value of the corresponding method. From left to right, the first four bars (blue) correspond to the topological feature based methods, the next four (green) represent time series based methods, and the final bar (brown) represents GraTFEL.

We observe that GraTFEL (the last bar) outperforms the remaining eight methods in all the nine charts in this figure. The performance difference using PRAUC (Fig. 5(b, e, f), the charts in the middle column) is more pronounced than the performance difference using the other two metrics. Since, PRAUC is the most informative metric for classification performance on a skewed dataset, the performance difference on this metric is a strong endorsement of the superiority of GraTFEL over other methods. We first analyze the performance




(a) Enron Network (b) Enron Network (c) Enron Network

(d) Collaboration Network (e) Collaboration Network (f) Collaboration Network

(g) Facebook Network (h) Facebook Network (i) Facebook Network

**Fig. 5.** Comparison with competing link prediction methods. Each bar represents a method and the height of the bar represents the value of the performance metrics. Results for Enron are presented in charts(a)–(c), results of Collaboration are in charts (d)–(f), and results of Facebook data are presented in charts (g)–(i). The group of bars in a chart are distinguished by color, so the figure is best viewed on a computer screen or color print. (Color figure online)

comparison between GraTFEL and topological feature based methods (first four bars). The best of the topological feature based methods have a PRAUC value of 0.41, 0.21, and 0.095 in Enron, Collaboration, and Facebook dataset, whereas the respective PRAUC values for GraTFEL are 0.546, 0.36, and 0.26, which translates to 33.2 %, 71.4 %, and 173.7 % improvement of PRAUC by GraTFEL for these datasets. Superiority of GraTFEL over all the topological feature based baseline methods can be attributed to the capability of graphlet transition based feature representation to capture temporal characteristics of local neighborhood. Finally, the performance of time series based method (four green bars) is generally better than the topological feature based methods. The

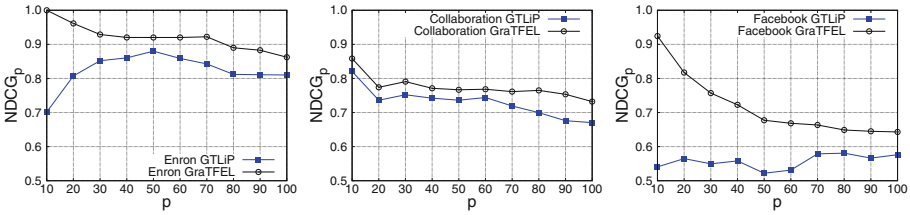
best of the time series based method has a PRAUC value of 0.503, 0.28, and 0.19 on these datasets, and GraTFEL's PRAUC values are better than these values by 8.5 %, 28.6 %, and 36.8 %, respectively. Time series based methods, though model the temporal behavior well, probably fail to capture signals from the neighborhood topology of the node-pairs.

Now we focus on the performance comparison using the information retrieval metric $NDCG_{50}$ (Fig. 5(c, f, i)). This metric puts higher weight on the top-ranked predicted links than the lower ranked predicted links. GraTFEL shows substantial improvement over the other methods using this metric also. The performance improvement of GraTFEL in $NDCG_{50}$ over the best among the remaining 8 methods are 9.8 %, 24 %, and 46.8 % on the three datasets. An interesting observation using this metric is that for all the datasets, the best among the topological feature based methods is better than the best among the time series based methods. It indicates that the top ranked predicted links are explained better by the neighborhood topology than the time series of interaction history. Finally, we discuss the comparison results for the AUC metric (Fig. 5(a, d, g)). GraTFEL is the best performer among all the methods for all three datasets with a percentage improvement over the second best method between 2 % to 17 %. Note that, the performance gap among all the methods are relatively small using AUC. For instance, the AUC values for all the methods in Enron dataset (Fig. 5(a)) are localized around 0.93. In general, AUC has a poor discrimination ability among classifiers in a highly skewed datasets and due to this reason PRAUC should be the preferred metric for such datasets [3].

When we compare the performance of all the algorithms across different datasets, we observe varying performance. For example, across all the metrics, the performance of dynamic link prediction on Facebook graph is lower than the performance on Collaboration graph, which, subsequently, is lower than the performance on Enron graph, indicating that link prediction in Facebook data is a harder problem to solve. The performance improvement of GraTFEL over the second best method for Facebook is the largest among the three datasets across all three metrics.

### 5.6 Contribution of Unsupervised Feature Extraction

GraTFEL has two novel aspects: first, utilization of graphlet transition events (GTEs) as features, and the second is unsupervised feature learning by optimal coding. In this section, we compare the relative contribution of these two aspects in the performance of GraTFEL. For this comparison, we build a version of GraTFEL which we call GTLiP. GTLiP uses GTEs just like GraTFEL, but it does not use optimal coding, rather it uses the GTEs directly as features. In Fig. 6, we show the comparison between GraTFEL, and GTLiP using $NDCG_p$ for different $p$ values for all the datasets. The superiority of GraTFEL over GTLiP for all the datasets over a range of $p$ values is clearly evident from the three charts. GTLiP also outperforms all the competing methods. Comparison of GTLiP and other competing methods is not shown in this figure, but the $NDCG_{50}$ from this figure can be compared with $NDCG_{50}$ charts in Fig. 5




(a) Enron Network (b) Collaboration Network (c) Facebook Network

**Fig. 6.** Performance comparison between link prediction methods with (GraTFEL) and without (GTLiP) unsupervised feature extraction. Y-axis represents the $NDCG_p$ score and X-axis represents the value of $p$.

for confirming this claim. This shows that GTE based features, irrespective of unsupervised coding, improve the dynamic link prediction performance over the existing state of-art dynamic link prediction methods.

## 6 Conclusion

In this paper, we present a dynamic link prediction method, which uses unsupervised coding of graphlet transition events. We demonstrate that the proposed method performs better than several topological feature based methods and several time series based methods on three real-life dynamic graph datasets.

## References

1. Bhuiyan, M.A., Rahman, M., Rahman, M., Hasan, M.A.: GUISE: uniform sampling of graphlets for large graph analysis. In: Proceedings of 12th International Conference on Data Mining, pp. 91–100 (2012)
2. Bliss, C.A., Frank, M.R., Danforth, C.M., Dodds, P.S.: An evolutionary algorithm approach to link prediction in dynamic social networks. J. Comput. Sci. **5**(5), 750–764 (2014)
3. Davis, J., Goadrich, M.: The relationship between precision-recall and ROC curves. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 233–240 (2006)
4. Dunlavy, D.M., Kolda, T.G., Acar, E.: Temporal link prediction using matrix and tensor factorizations. ACM Trans. Knowl. Disc. Data **5**(2), 10:1–10:27 (2011)
5. Güneş, İ., Gündüz-Öğüdücü, Ş., Çataltepe, Z.: Link prediction using time series of neighborhood-based node similarity scores. Data Min. Knowl. Disc. **30**(1), 147–180 (2015)
6. Hasan, M.A., Chaoji, V., Salem, S., Zaki, M.: Link prediction using supervised learning. In: Proceeding of SDM 2006 workshop on Link Analysis, Counterterrorism and Security (2006)
7. Hasan, M., Zaki, M.: A survey of link prediction in social networks. In: Social Network Data Analytics, pp. 243–275 (2011)

8. Lee, H., Pham, P., Largman, Y., Ng, A.Y.: Unsupervised feature learning for audio classification using convolutional deep belief networks. In: Advances in neural information processing systems. pp. 1096–1104 (2009)
9. Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. J. Am. Soc. Inform. Sci. Technol. **58**(7), 1019–1031 (2007)
10. Lichtenwalter, R.N., Lussier, J.T., Chawla, N.V.: New perspectives and methods in link prediction. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 243–252 (2010)
11. Menon, A.K., Elkan, C.: Link prediction via matrix factorization. In: Proceedings of the 2011 European Conference on Machine Learning and Knowledge Discovery in Databases - volume Part II, pp. 437–452 (2011)
12. Milenković, T., Pržulj, N.: Uncovering biological network function via graphlet degree signatures. Cancer Inform. **6**, 257–273 (2008)
13. Ng, J., Yang, F., Davis, L.: Exploiting local features from deep networks for image retrieval. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 53–61 (2015)
14. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 701–710 (2014)
15. Pržulj, N.: Biological network comparison using graphlet degree distribution. Bioinformatics **23**(2), e177–e183 (2007)
16. Pržulj, N., Corneil, D.G., Jurisica, I.: Modeling interactome: scale-free or geometric? Bioinformatics **20**(18), 3508–3515 (2004)
17. Rahman, M., Bhuiyan, M.A., Hasan, M.A.: Graft: An efficient graphlet counting method for large graph analysis. IEEE Trans. Knowl. Data Eng. **26**(10), 2466–2478 (2014)
18. Sarkar, P., Chakrabarti, D., Jordan, M.I.: Nonparametric link prediction in dynamic networks. In: Proceedings of International Conference on Machine Learning, pp. 1687–1694 (2012)
19. Shervashidze, N., Petri, T., Mehlhorn, K., Borgwardt, K.M., Vishwanathan, S.: Efficient graphlet kernels for large graph comparison. In: Proceeding of the 12th International Conference on Artificial Intelligence and Statistics, pp. 488–495 (2009)
20. Viswanath, B., Mislove, A., Cha, M., Gummadi, K.P.: On the evolution of user interaction in facebook. In: Proceedings of the 2nd ACM workshop on Online social networks, pp. 37–42 (2009)
21. Xu, K.S.: Stochastic block transition models for dynamic networks. In: Proceedings of the 18th International Conference on Artificial Intelligence and Statistics, pp. 1079–1087 (2015)
22. Xu, K.S., Hero III, A.O.: Dynamic stochastic blockmodels for time-evolving social networks. IEEE J. Sel. Top. Sign. Proces. **8**(4), 552–562 (2014)