Streaming Link Prediction on Dynamic Attributed Networks

Jundong Li
Arizona State University
jundongl@asu.edu

Kewei Cheng
Arizona State University
kcheng18@asu.edu

Liang Wu
Arizona State University
wuliang@asu.edu

Huan Liu
Arizona State University
huan.liu@asu.edu

ABSTRACT

Link prediction targets to predict the future node interactions mainly based on the current network snapshot. It is a key step in understanding the formation and evolution of the underlying networks; and has practical implications in many real-world applications, ranging from friendship recommendation, click through prediction to targeted advertising. Most existing efforts are devoted to plain networks and assume the availability of network structure in memory before link prediction takes place. However, this assumption is untenable as many real-world networks are affiliated with rich node attributes, and often, the network structure and node attributes are both dynamically evolving at an unprecedented rate. Even though recent studies show that node attributes have an added value to network structure for accurate link prediction, it still remains a daunting task to support link prediction in an online fashion on such dynamic attributed networks. As changes in the dynamic attributed networks are often transient and can be endless, link prediction algorithms need to be efficient by making only one pass of the data with limited memory overhead. To tackle these challenges, we study a novel problem of streaming link prediction on dynamic attributed networks and present a novel framework - SLIDE. Methodologically, SLIDE maintains and updates a low-rank sketching matrix to summarize all observed data, and we further leverage the sketching matrix to infer missing links on the fly. The whole procedure is theoretically guaranteed, and empirical experiments on real-world dynamic attributed networks validate the effectiveness and efficiency of the proposed framework.

ACM Reference Format:
Jundong Li, Kewei Cheng, Liang Wu, and Huan Liu. 2018. Streaming Link Prediction on Dynamic Attributed Networks. In *WSDM 2018: The Eleventh ACM International Conference on Web Search and Data Mining, February 5–9, 2018, Marina Del Rey, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3159652.3159674

1 INTRODUCTION

The pervasiveness of myriad social media platforms significantly alters the conventional ways people interact and communicate. By performing various social activities (e.g., befriend with others, follow a celebrity, retweet a post), online social media users build different social interactions which ease the channel of information seeking and spreading. To this end, understanding and predicting the formation of social interactions will shed light on the underlying evolution mechanisms of the social networks; and also, from a microscopic perspective, it can help us gain more insights on the social behaviors of a particular user. The problem is often referred as the *link prediction problem* in the data mining and network science community [5, 37]. Formally, given a snapshot of network structure with partially observed links, link prediction aims to infer the missing links that may appear in a foreseeable future. The link prediction problem also has practical implications in many real-world applications, ranging from friend suggestion, click through prediction to targeted marketing [17].

Despite much progress has been made in the field of link prediction, existing link prediction algorithms overwhelmingly assume the availability of whole network structure. However, this assumption is untenable as real-world networks are often in large-scale and are measured in terabytes or even petabytes, making them difficult to be materialized in memory before link prediction takes place. In addition, networks are often not static but are dynamically evolving at an unprecedented rate with frequent updates (e.g., additions/deletions of nodes and edges). Furthermore, a vast majority of existing work are devoted to predicting missing links in plain networks. More often than not, nodes in the networks are affiliated with a rich set of attributes such as user profile information and user posts. Recent studies indicate that the formation of network structure highly depends on the associated node attributes and vice versa [28, 29, 34, 43]. On account of this, probing the node attributes could be potentially helpful in achieving better link prediction performance, especially when the network suffers from high sparsity. On top of that, by leveraging node attributes, it enables the prediction for those cold-start users that are otherwise intractable. In many cases, similar to the frequent updates on the network structure, node attributes also change naturally [30, 31], a typical example is the modification of online user posts. We refer such kind of networks with both topology and attribute changes as *dynamic attributed networks*.

As per the fact that node attributes are complementary for link prediction while both network structure and node attributes exhibit high dynamics, we investigate a novel problem of *streaming link prediction on dynamic attributed networks* in this paper. The following five challenges have to be addressed simultaneously.

- **Near Real-Time Prediction**: Dynamic attributed networks are characterized by streaming nodes/edges of high velocity.

WSDM 2018, February 5–9, 2018, Marina Del Rey, CA, USA

ACM ISBN 978-1-4503-5581-0/18/02…$15.00
https://doi.org/10.1145/3159652.3159674

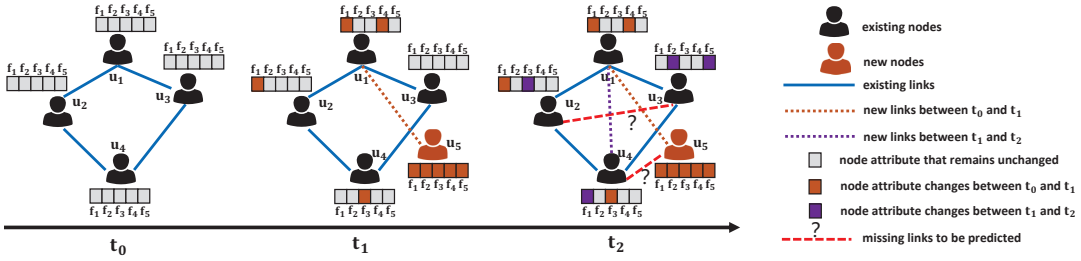


Figure 1: An illustrative example of streaming link prediction on dynamic attributed networks. As shown in the figure, at each time stamp, each node in the network is affiliated with five features ($f_1, ..., f_5$). Edges and nodes are continuously arriving in a streaming fashion. Meanwhile, the attributes of a certain number of nodes may change accordingly. Streaming link prediction attempts to infer the missing links in parallel with the rapid changes of the dynamic attributed networks. For example, given the observations at time stamp t_0, t_1 and t_2, it predicts whether there exists a link between u_2 and u_3 in the near future.

Also, the evolution of networks is often mixed with the changes of node attributes at an unsynchronized rate. As changes are essential components of the system and could occur at any time, link prediction algorithms require to be efficient and are performed in a streaming fashion to predict missing links in close to near real-time.

- **One-Pass of the Data**: The entire size of the network and affiliated node attributes are often unknown at a particular moment and could even be infinite in the worst case. Hence, the streaming link prediction algorithms need to be pass-efficient to make only one pass of the data as the further passes are either expensive or naturally impossible.
- **Space Efficiency**: Data is continuously being generated, the huge volume of data makes the dynamic attributed network hard to be materialized in memory, which necessitates the design of a cost-effective data synopsis with limited memory overhead to summarize the ever-increasing network structure and node attributes.
- **Concept Drift**: With the accumulation of new nodes/edges and the changes of node attributes, the underlying network topology and the content patterns of nodes continuously evolve over time, resulting in the emerging of unseen patterns and the fading of existing patterns, which may significantly impact the link prediction performance. This phenomenon is often referred as *concept drift* in data stream mining. In this regard, link prediction algorithms should be able to tackle the issue of concept drift.
- **Data Heterogeneity**: We examine the link prediction problem in the case of attributed networks and this kind of networks are notoriously difficult to mine due to the bewildering combination of heterogeneous data sources. Even though in the most cases that network structure and node attributes are presented in different modalities, they are often not mutually independent and could influence each other. Link prediction algorithms are supposed to seize the inherent interconnections for accurate prediction.

In this paper, we study a novel problem of *streaming link prediction on dynamic attributed networks*. An illustrative example of the studied problem is shown in Figure 1. Concretely, given a network that is characterized by fast-evolving links and node attributes, we attempt to propose an effective yet efficient (in terms of both time and space) model to enable the prediction of missing links on the fly. The main contributions of this paper are summarized as follows.

- We formally define a novel problem of streaming link prediction on dynamic attributed networks and identify its connection with many real-world applications. To our best knowledge, this is the first attempt to study link prediction in a dynamic and attributed environment.
- We propose a novel link prediction framework - SLIDE that is able to predict the missing links in an online fashion when the attributed network is characterized by both structural and attribute changes. By consuming limited memory, the proposed method leverages a cost-effective data sketch to tackle the concept drift of the underlying attributed networks with only one pass of the data.
- We theoretically prove that the proposed streaming link prediction framework SLIDE achieves similar performance as a costly offline method which stores and uses the whole historically observed data for link prediction.
- Extensive experiments on various real-world dynamic attributed networks reveal the effectiveness and efficiency of the proposed streaming link prediction framework.

2 PROBLEM STATEMENT

We use bold uppercase characters for matrices (e.g., $\mathbf{A}$), bold lowercase characters for vectors (e.g., $\mathbf{a}$), and normal lowercase characters for scalars (e.g., a). Also, we represent the i-th element of vector $\mathbf{a}$ as $\mathbf{a}_i$, the i-th row of matrix $\mathbf{A}$ as $\mathbf{A}_{i*}$, the j-th column as $\mathbf{A}_{*j}$, the (i,j)-th entry as $\mathbf{A}_{ij}$, the transpose of $\mathbf{A}$ as $\mathbf{A}^T$, the trace of $\mathbf{A}$ as $tr(\mathbf{A})$. For any vector $\mathbf{a} \in \mathbb{R}^n$, $\|\mathbf{a}\|_2 = \sqrt{\sum_i \mathbf{a}_i^2}$ denotes its Euclidean norm. For any matrix $\mathbf{A} \in \mathbb{R}^{n\times m}$, its Frobenius norm is $\|\mathbf{A}\|_F = \sqrt{\sum_{i,j} \mathbf{A}_{ij}^2}$, its spectral norm is $\|\mathbf{A}\| = \max_{\|\mathbf{x}\|_2=1} \|\mathbf{A}\mathbf{x}\|_2$. The *singular value decomposition* (SVD) of $\mathbf{A} \in \mathbb{R}^{n\times m}$ is denoted as $\text{svd}(\mathbf{A}) = \mathbf{U}\Sigma\mathbf{V}^T$, $\mathbf{U}$ is a $n \times n$ orthogonal matrix with the rows being left singular vectors $[\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_n]$, $\mathbf{V}$ is a $m \times m$ orthogonal matrix with the columns being the right singular vectors $[\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_m]$, $\Sigma = \text{diag}(\sigma_1, ..., \sigma_n)$ is a $n \times m$ diagonal matrix, where $\sigma_1 \geq \sigma_2 \geq, ..., \geq \sigma_r$ are the singular values of $\mathbf{A}$ and r is the rank of matrix $\mathbf{A}$. The best rank-k ($k \leq r$) approximation of matrix $\mathbf{A} \in \mathbb{R}^{n\times m}$ is $\mathbf{A}_k = \text{argmin}_{\text{rank}(\mathbf{X})\leq k} \|\mathbf{A} - \mathbf{X}\|_F$ and it can

be computed as $\mathbf{A}_k = \mathbf{U}_k\Sigma_k\mathbf{V}_k^T = \sum_{i=1}^{k}\sigma_i\mathbf{u}_i\mathbf{v}_i^T$, where $\mathbf{U}_k$, Σ_k, $\mathbf{V}_k$ are the truncated matrices consisting of the top-k left singular vectors, singular values, and right singular vectors, respectively. $\mathbf{I}$ is an identity matrix.

Definition 2.1. (**Dynamic Attributed Networks**): A dynamic attributed network $G = (G^{t_0}, G^{t_1}, G^{t_2}, ...)$ spans across various time stamps $\{t_0, t_1, t_2, ...\}$. At a particular time stamp t, the corresponding attributed network $G^t = (\mathcal{V}^t, \mathcal{E}^t, \mathbf{X}^t)$ consists of $\mathcal{V}^t$: the set of nodes ($n_t = |\mathcal{V}^t|$); $\mathcal{E}^t \subseteq \mathcal{V}^t \times \mathcal{V}^t$: the set of edges; and the node attributes $\mathbf{X}^t = [\mathbf{x}^1, \mathbf{x}^2, ..., \mathbf{x}^{n_t}]^T$, where $\mathbf{x}^i \in \mathbb{R}^d$ ($i = 1, ..., n_t$) is the attribute information of the i-th node.

The problem of streaming link prediction problem on dynamic attributed networks can be formally defined as follows:

Problem 1. *Given a dynamic attributed network $G = (G^{t_0}, G^{t_1}, ...)$ with fast-evolving node/edge streams and changes of node attributes across multiple time stamps $(t_0, t_1, ...)$. At a particular time stamp t, the **streaming link prediction** problem aims to predict if there exists an edge $e = (u, v)$ for a pair of nodes (u, v) that are not connected previously before time stamp t.*

An illustration of the studied problem is shown in Figure 1. The streaming link prediction problem supports the prediction of missing links in an online fashion. For example, as shown in the figure, given a dynamic attributed network at three different time stamps t_0, t_1 and t_2 with both network structure and node attribute changes, the streaming link prediction problem predicts the missing links that may appear at time stamp t, where $t > t_2$.

3 THE PROPOSED FRAMEWORK - SLIDE

In this section, we present the proposed framework of **S**treaming **L**ink pred**I**ction for **D**ynamic attribut**E**d networks - SLIDE. The basic idea is to maintain and update a low-rank sketching matrix with limited memory overhead to summarize the currently observed links and node attributes. Then given the attributes of a pair of unconnected nodes, we leverage the low-rank sketching matrix to determine if there exists a link between these two end nodes in the future. The low-rank sketching matrix is continuously being updated when new links and new node attributes are observed. The overall workflow of the proposed streaming link prediction framework - SLIDE is shown in Figure 2. As can be observed from the figure, the proposed framework consists of three essential components: (1) maintain and update a sketching matrix to summarize the currently observed data, including all the observed links and node attributes; (2) predict missing links on the fly with the up-to-date sketching matrix; (3) calculate and update the threshold which is used to determine the existence of links.

3.1 Summarization with Matrix Sketching

On a typical dynamic attributed network, a massive amount of edges are continuously arriving at a fast pace. Meanwhile, node attributes also change naturally such that new content patterns may emerge and outdated content patterns will fade. To explicitly store such a dynamic attributed network at a particular time stamp t, we need $O(n_t^2 + n_t d)$ space in the worst case (d is often much smaller than n_t), where n_t is the number of nodes in the network until t and d is the dimensionality of node attributes. The materialization of the attributed networks becomes infeasible when n_t is very large. Hence, it is of vital importance to use cost-effective data synopsis to summarize all observed data including the links and node attributes. Nonetheless, designing a full streaming model with limited and constant memory space is a challenging problem and most of the existing efforts on graph streams are devoted to the so-called *semi-streaming* models [15, 42], which requires $O(n_t\text{polylog}(n_t))$ space. These semi-streaming models are intractable if the available memory is not proportional to the number of nodes n_t in the network. In addition, due to the heterogeneity of two information sources in attributed networks, the resulted data synopsis is expected to summarize both information sources simultaneously.

Motivated by the recent advances in full streaming models for conventional data streams, we propose to use the frequent directions algorithm [38] to maintain a low-rank sketching matrix (with limited memory overhead) to make a structural summary of the currently observed data. One major merit of the frequent directions algorithm is that it operates in a streaming fashion and makes only one pass of the data. However, the frequent directions algorithm cannot be directly applied on dynamic attributed networks for a low-rank approximation in real-time. The reason is that frequent directions algorithm is proposed to summarize conventional data streams where columns of the input matrix are added incrementally and the row of the input matrix is fixed. On dynamic attributed networks, even though the node attributes are presented as a data stream (if the number of node attributes is fixed), the changes of the underlying network structure (often encoded in an adjacency matrix), per se, cannot be simply generalized as a conventional data stream. To this end, we propose to represent the dynamic attributed networks as the feature representation based on the observed links. The similar feature representation mechanism is also widely used in many other learning tasks, such as factorization machines [46] and contextual-bandit collaborative filtering [35, 52].

Definition 3.1. (**Feature Representation of Dynamic Attributed Networks**): Given a dynamic attributed network across multiple time stamps $G = (G^{t_0}, G^{t_1}, ...)$, its feature representation at a particular time stamp t is represented $\mathbf{F}^t \in \mathbb{R}^{2d\times|\mathcal{E}^t|}$. Each column $\mathbf{f} \in \mathbb{R}^{2d}$ in the feature representation $\mathbf{F}^t$ corresponds to an edge in G^t. Now assume that two end nodes of the edge is u_i and u_j ($i < j$), then the corresponding feature representation, i.e., $\mathbf{f}$, can be represented as $\mathbf{f} = [\mathbf{X}^t_{i*}, \mathbf{X}^t_{j*}]^T$, where $\mathbf{X}^t$ is the node attributes of the dynamic attributed network at time stamp t.

By transforming the dynamic attributed networks into feature representations, the changes in the underlying attributed network can be presented as new columns in a conventional data stream. In particular, new columns are introduced in two scenarios: (1) the arrival of new edges; and (2) node attribute information changes. The first scenario is straightforward and easy to understand. Regarding the second scenario, if the node attributes change, then the feature representations of edges that these nodes involved in should also change, and we represent these changes as new columns in the data stream. Now we assume that we can store all the historically generated data, and the feature representation of the dynamic attributed networks until time stamp t is stored in a matrix $\mathbf{G}^t \in \mathbb{R}^{2d\times c_t}$, where c_t is the number of columns in $\mathbf{G}^t$, and $c_t \geq |\mathcal{E}^t|$. It should

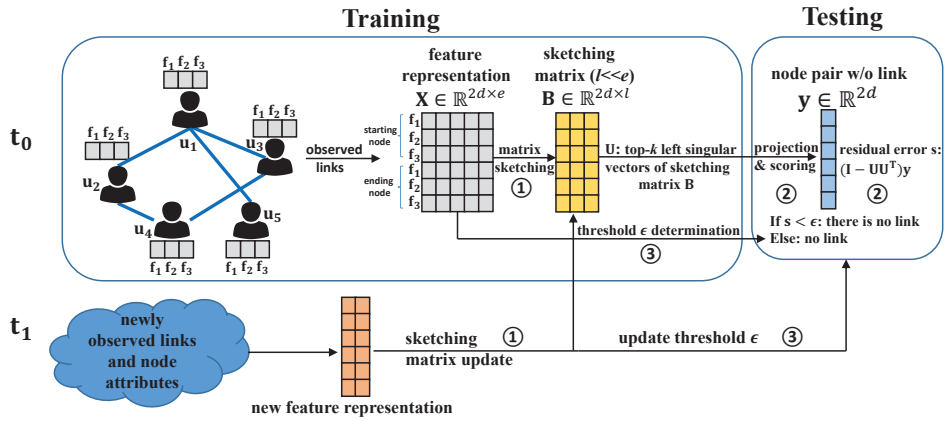


Figure 2: The workflow of the proposed streaming link prediction framework - SLIDE.

be noted that the similar technique is also applied on the dynamic recommendation problem recently [26]. However, it is different from the proposed framework as we focus on the streaming link prediction problem and consider the changes of both network structure and node attributes. While on the other hand, [26] only models the new ratings (with contextual information) as a data stream and fails to consider the changes of contextual information.

Even though we have reformulated the changes in the dynamic attributed networks as new columns in a data stream, the number of columns in $\mathbf{G}^t$ are still too large to be stored in memory, especially when the number of edges is in the scale of billion or trillion. Instead of storing the feature representations of the underlying dynamic attributed network, the frequent directions algorithm is employed to maintain a low-rank sketching matrix, and the sketching matrix can well summarize the observed data with a theoretical guarantee. Specifically, the low-rank sketching matrix $\mathbf{B}^t \in \mathbb{R}^{2d \times l}$ (l is often small) approximates the matrix $\mathbf{G}^t$ well such that $\mathbf{B}^t(\mathbf{B}^t)^T \approx \mathbf{G}^t(\mathbf{G}^t)^T$. More accurately, the approximation error is bounded by the conditions that: (1) $\mathbf{G}^t(\mathbf{G}^t)^T \succeq \mathbf{B}^t(\mathbf{B}^t)^T$; and (2) $\|\mathbf{G}^t(\mathbf{G}^t)^T - \mathbf{B}^t(\mathbf{B}^t)^T\| \leq 2\|\mathbf{G}^t\|_F^2/l$ [38].

Let $\mathbf{D}^t \in \mathbb{R}^{2d \times m_t}$ denotes the new columns generated between time stamp $t-1$ and the following time stamp t such that $\mathbf{G}^t = [\mathbf{G}^{t-1}, \mathbf{D}^t]$, where m_t is the number of new columns generated between time stamps $t-1$ and t. As mentioned above, the generation of new columns pertains to the arrival of new edges or the changes of node attributes. Here the challenges center around how to maintain and update the sketching matrix $\mathbf{B}^t$ based on the newly generated data in $\mathbf{D}^t$. At the very beginning, the sketching matrix $\mathbf{B}^t$ ($t = 0$) is set to be empty, then new columns presented in a data stream are continuously being inserted into the sketching matrix $\mathbf{B}^t$ until there are no empty columns anymore. Then frequent directions algorithm "shrinks" l orthogonal vectors by the same amount to make space for new data in the future. Concretely, a computation of singular value decomposition (SVD) is necessary each time when the sketching matrix $\mathbf{B}^t$ is full. The original frequent directions algorithm assumes that one column arrives at each time stamp, [23, 24] further extended it to tackle the case when more than one columns arrive at each time stamp. As we have more than one column in $\mathbf{D}^t$ in most cases, we leverage this

Algorithm 1 Maintain and update the sketching matrix $\mathbf{B}^t$

Input: Sketching matrix $\mathbf{B}^{t-1} \in \mathbb{R}^{2d \times l}$, new data $\mathbf{D}^t \in \mathbb{R}^{2d \times m_t}$.
Output: New sketching matrix $\mathbf{B}^t \in \mathbb{R}^{2d \times l}$, and $\tilde{\mathbf{U}}_l^t \in \mathbb{R}^{2d \times l}$.
1: $\mathbf{C}^t = [\mathbf{B}^{t-1}, \mathbf{D}^t] \in \mathbb{R}^{2d \times (l+m_t)}$;
2: $[\tilde{\mathbf{U}}_l^t, \tilde{\Sigma}_l^t, \tilde{\mathbf{V}}_l^t] = \mathrm{svd}_l(\mathbf{C}^t)$;
3: $\hat{\Sigma}_l^t = \mathrm{diag}(\sqrt{\tilde{\sigma}_1^2 - \tilde{\sigma}_l^2}, \sqrt{\tilde{\sigma}_2^2 - \tilde{\sigma}_l^2}, \ldots, \sqrt{\tilde{\sigma}_{l-1}^2 - \tilde{\sigma}_l^2}, 0)$;
4: $\mathbf{B}^t = \tilde{\mathbf{U}}_l^t \hat{\Sigma}_l^t$;

general solution to maintain and update the low-rank sketching matrix $\mathbf{B}^t$, upon which the patterns in the observed links and node attributes can be summarized accurately. The detailed pseudo code of the summarization phase using matrix sketching is presented in Algorithm 1. As the number of new columns m_t is much smaller than the number of columns in $\mathbf{G}^t$, we only need to perform SVD on a low-rank matrix (Line 2); its computation is efficient with a complexity of $O(2d(m_t + l)l)$. Also, it is space efficient with a maximum overhead of $O(2d(max_t\{m_t\} + l))$ across all time stamps. All in all, the summarization phase makes only one pass of the data and is both computational and space efficient.

3.2 Infer Missing Links with Sketching Matrix

The aforementioned low-rank sketching matrix $\mathbf{B}^t$ makes a structural summarization of the up-to-date observed data on dynamic attributed networks. Hence, we can leverage it to predict missing links on the fly. To show the underlying mechanism of the link prediction phase, we first assume that the feature representation $\mathbf{G}^t$ of the dynamic attributed network until time stamp t is available (which actually not). The original feature representation $\mathbf{G}^t$ could be very noisy, containing a certain amount of noisy and irrelevant attributes which may degrade the link prediction performance [32, 51]. On top of that, the link information in networks may also be noisy and even erroneous from a network analysis perspective [16]. To alleviate the negative impacts from these noisy attributes and noisy links, we propose to use principal component analysis (PCA) to reduce the noise hidden in the data stream. Formally, PCA projects the data in $\mathbf{G}^t$ onto several principal components such that the total data variance is minimized, and these principal components correspond to the top-k eigenvectors of the estimated covariance matrix

$\frac{1}{c_t}\mathbf{G}^t(\mathbf{G}^t)^T$, which is also equivalent to find the top-k left singular vectors of the matrix $\mathbf{G}^t$. Here, we denote the concatenation of the top-k eigenvectors as $\mathbf{U}_k^t \in \mathbb{R}^{2d\times k}$, and the principal components $\mathbf{U}_k^t$ is often regarded as a good rank-k basis to reconstruct all the data in the original representation $\mathbf{G}^t$. In addition, by using the linear combination of columns in $\mathbf{U}_k^t$ to represent columns in $\mathbf{G}^t$, the noisy information contained can be greatly reduced. As $\mathbf{U}_k^t$ provides a noise-resilient abstraction of patterns of connected node pairs, we can leverage the orthogonal basis to predict missing links for unconnected node pairs. In particular, let $\mathbf{y} \in \mathbb{R}^{2d}$ denotes the feature representation of an unconnected node pair for testing, if $\mathbf{y}$ is close to the space composed of columns in $\mathbf{U}_k^t$, mostly likely there will be a link between the starting node and the ending node of $\mathbf{y}$; otherwise, if $\mathbf{y}$ cannot be well reconstructed by the orthogonal basis $\mathbf{U}_k^t$, it implies that $\mathbf{y}$ deviates from the patterns of connected node pairs and the possibility that the two end nodes of $\mathbf{y}$ connected in the near future is low [23]. The residual error of the reconstruction phase is $\|\mathbf{I} - \mathbf{U}_k^t(\mathbf{U}_k^t)^T\mathbf{y}\|_2^2$. It can be observed that to obtain the residual error for each pair of unconnected nodes, only low-rank matrix multiplication operations are involved, and the computation cost is low, with a complexity of $O(2dk)$. Afterwards, we can take advantage of the residual error to predict whether there exists a link between a pair of unconnected nodes. Specifically, the smaller the residual error of a pair of unconnected nodes, the higher the chance that these two nodes will be linked in the near future.

The above phase assumes that the feature representation $\mathbf{G}^t$ is readily available before the link prediction takes place and the orthogonal basis $\mathbf{U}_k^t$ is the top-k left singular vectors of $\mathbf{G}^t$. However, as mentioned in the previous subsection that the storage of the whole feature representation $\mathbf{G}^t$ is intractable when the underlying attributed network is large; while on the other hand, the sketching matrix $\mathbf{B}^t$ is not only a low-rank matrix with light memory overhead, but also can approximate the original matrix $\mathbf{G}^t$ well. In this regard, we attempt to perform SVD on the low-rank matrix $\mathbf{B}^t$ instead of $\mathbf{G}^t$ to obtain the orthogonal basis, i.e., the top-k left singular vectors. Here, we denote these k left singular vectors of $\mathbf{B}^t$ as $\tilde{\mathbf{U}}_k^t \in \mathbb{R}^{2d\times k}$. And according to Algorithm 1, the sketching matrix $\mathbf{B}^t$ can be efficiently maintained and updated, and its top-l left singular vectors are $\tilde{\mathbf{U}}_l^t$. In this way, the approximation of the top-k left singular vectors of $\mathbf{G}^t$ can be directly obtained from $\tilde{\mathbf{U}}_l$ as long as the condition of $k \le l$ is satisfied. And the residual error of the feature vector $\mathbf{y}$ is $\|\mathbf{I} - \tilde{\mathbf{U}}_k^t(\tilde{\mathbf{U}}_k^t)^T\mathbf{y}\|_2^2$.

3.3 Threshold Determination

For the above phase, the potential links between unconnected nodes can be determined by verifying if the residual error is below a threshold. One simple solution to specify the threshold is to set it as a fixed value. Nonetheless, with the accumulation of new edges and new node attributes in a data stream over time, the intrinsic patterns of data change over time, and this phenomenon is often referred as concept drift in data stream mining [50]. To this end, it is more appealing to continuously update the threshold value for link prediction such that the up-to-date patterns of data can be well captured. Concretely, we propose to obtain the threshold automatically from the presented data stream instead of manually setting it up. For each observed link, i.e., a column in $\mathbf{G}^t$, we calculate its residual error, where $\tilde{\mathbf{U}}_k^t$ can be obtained by the top-k left singular vectors of the current sketching matrix $\mathbf{B}^t$ (Algorithm 1). Let us denote the collection of residual errors of links in $\mathbf{G}^t$ as $R = \{error_1, error_2, ..., error_{c_t}\}$, then the residual error threshold that is used to check the existence of new links can be determined as the largest error among R. In this way, we do not need to manually specify the threshold value and it can be automatically determined from the observed links.

Algorithm 2 SLIDE to predict missing links at time t

Input: Sketching matrix $\mathbf{B}^{t-1} \in \mathbb{R}^{2d\times l}$ and its top-l left singular vectors $\tilde{\mathbf{U}}_l^{t-1}$, new data $\mathbf{D}^t \in \mathbb{R}^{2d\times m_t}$, residual error threshold ϵ^{t-1}, feature representation $\mathbf{y}$ of an unconnected node pair.
Output: If there exists a link between the two end nodes of $\mathbf{y}$.

```
Obtain the new sketching matrix B^t and its top-l left singular vectors Ũ_l^t by Algorithm 1;
Obtain the top-k singular vectors Ũ_k^t from Ũ_l^t (k ≤ l);
Calculate the residual error of links in D^t;
Update the residual error threshold ε^t;
Calculate the residual error of y by ||I − Ũ_k^t (Ũ_k^t)^T) y||_2^2;
if error of y ≤ ε^t then
    There exists a future link between the two end nodes of y;
else
    The two end nodes of y will not be connected;
end if
```

Hence, the whole procedure of streaming link prediction on dynamic attributed networks is summarized in Algorithm 2.

3.4 Theoretical Analysis of SLIDE

Next, we perform a theoretical analysis of the proposed streaming link prediction framework - SLIDE. In the proposed SLIDE framework, to calculate the possibility of connection between a pair of previously unconnected nodes at time stamp t, we make use of $\tilde{\mathbf{U}}_k^t$, which is the top-k left singular vectors of the sketching matrix $\mathbf{B}^t$. Now we compare the error bound of $\tilde{\mathbf{U}}_k^t$ against $\mathbf{U}_k^t$, where $\mathbf{U}_k^t$ is obtained from the original feature representation $\mathbf{G}^t$, i.e., the top-k left singular vectors of $\mathbf{G}^t$. Specifically, motivated by the analysis of the frequent directions algorithm [18] and matrix perturbation theory [47], we show that the Frobenius norm of the difference between $\tilde{\mathbf{U}}_k^t$ and $\mathbf{U}_k^t$ is bounded.

First, we define the k-conditional number of $\mathbf{G}^t$ as $\kappa = \sigma_1/\sigma_k$, where σ_i is the i-th singular value of $\mathbf{G}^t$. Next, we define α as:

$$\alpha = \frac{\kappa^2\|\mathbf{G}_k^t\|_F^2 - \|\mathbf{B}_k^t\|_F^2}{\|\mathbf{G}_k^t\|_F^2 - \|\mathbf{B}_k^t\|_F^2}. \tag{1}$$

Lemma 3.2. *[19] For any unit vector $\mathbf{x} \in \mathbb{R}^{2d}$, the following inequality holds $\|(\mathbf{G}^t)^T\mathbf{x}\|_2^2 \ge \|(\mathbf{B}^t)^T\mathbf{x}\|_2^2$.*

Lemma 3.3. *The inequality of $\|\mathbf{G}_k^t\|_F^2 \ge \|\mathbf{B}_k^t\|_F^2$ holds.*

Proof. Assume that $\mathbf{a}_1, ..., \mathbf{a}_k$ are the top-k left singular vectors of $\mathbf{G}^t$, $\mathbf{b}_1, ..., \mathbf{b}_k$ are the top-k left singular vectors of $\mathbf{B}^t$. Then, we have $\|\mathbf{G}_k^t\|_F^2 = \sum_{i=1}^k \|(\mathbf{G}^t)^T\mathbf{a}_i\|_2^2 \ge \sum_{i=1}^k \|(\mathbf{G}^t)^T\mathbf{b}_i\|_2^2$. According to Lemma 3.2, $\sum_{i=1}^k \|(\mathbf{G}^t)^T\mathbf{b}_i\|_2^2 \ge \sum_{i=1}^k \|(\mathbf{B}^t)^T\mathbf{b}_i\|_2^2$. Therefore,

$\|\mathbf{G}_k^t\|_F^2 \geq \sum_{i=1}^k \|(\mathbf{B}^t)^T\mathbf{b}_i\|_2^2 = \|\mathbf{B}_k^t\|_F^2$, which completes the proof. □

As $\|\mathbf{G}_k^t\|_F^2 \geq \|\mathbf{B}_k^t\|_F^2$, and $\kappa \geq 1$, we have $\alpha \geq 1$. In addition, as κ is a bounded number [47], we have $\alpha = O(1)$. Now, we define β as:

$$\beta = \frac{\kappa^2 - \|\mathbf{B}^t\|^2/\|\mathbf{G}^t\|^2 + 2}{\kappa^2 - \|\mathbf{B}^t\|^2/\|\mathbf{G}^t\|^2}. \quad (2)$$

According to the definition of the sketching matrix, we have $\mathbf{G}^t(\mathbf{G}^t)^T \geq \mathbf{B}^t(\mathbf{B}^t)^T$. Thus, $\|\mathbf{G}^t\|^2 \geq \|\mathbf{B}^t\|^2$. Therefore, we have $\beta = O(1)$.

Theorem 3.4. *[23] If the size of the sketching matrix l is:*

$$l = \Omega(\frac{\sqrt{2d}\kappa^2\alpha\beta k\|\mathbf{G}^t\|^2\|\mathbf{G}^t - \mathbf{G}_k^t\|_F^2}{T^2}),$$

where $T = \min_{i\neq j}|\lambda_i - \lambda_j| > 0$, and λ_i is the i-the eigenvalue of the matrix $\mathbf{G}^t(\mathbf{G}^t)^T$. Then the Frobenius norm difference between $\tilde{\mathbf{U}}_k^t$ and $\mathbf{U}_k^t$ is bounded by the following term:

$$\|\tilde{\mathbf{U}}_k^t - \mathbf{U}_k^t\|_F \leq \frac{\sqrt{2}T}{\sqrt{T + 8\kappa^2\|\mathbf{G}^t\|^2}\sqrt[4]{T^2 + 16\kappa^4\|\mathbf{G}^t\|^4}}.$$

The above l can further be simplified to $l = \Omega(\frac{\sqrt{2d}k\|\mathbf{F}^t\|^2\|\mathbf{F}^t - \mathbf{F}_k^t\|_F^2}{T^2})$ as $0 < \kappa \leq O(1)$, $\alpha = O(1)$ and $\beta = O(1)$.

The above theorem suggests that as long as the size of the sketching matrix l is set within a reasonable range, then the resulted orthogonal basis $\tilde{\mathbf{U}}_k^t$ from the sketching matrix $\mathbf{B}^t$ approximates the orthogonal basis $\mathbf{U}_k^t$ from the whole observed data $\mathbf{G}^t$.

4 EXPERIMENTS

In this section, we perform experiments on real-world dynamic attributed networks to validate the effectiveness and efficiency of the proposed SLIDE framework for streaming link prediction.

4.1 Datasets

We collect three real-world dynamic attributed networks for experimental validation, these datasets range from social media networks to coauthor networks. The detailed descriptions of these three dynamic attributed networks are listed below.

Epinions: Epinions is a product review site in which users build trust relationships to seek advice from others and share their reviews about products. We take each user as a node and regard his/her reviews as node attributes. In particular, we first use the bag-of-words model to extract features from user reviews and then employ state-of-the-art unsupervised feature selection methods for networked data [32] to find the top 100 important features closely hinged with the network topology. Both the network structure and the node attributes are evolving over time. In the collected dataset, there are 25 time stamps (with an interval of one month), the total number of nodes is 14,180 and the total number of edges is 308,136.

DBLP: DBLP is an extracted coauthor network for the authors who publish at least three papers from the year of 1995 to 2011. In the network, each author corresponds to a node. And similar to Epinions, we apply the bag-of-words model and feature selection on the title of their publications to find the most relevant 100 node attributes, i.e., words. As authors gradually form new coauthor relations and their research interests evolve over time, the underlying network is naturally a dynamic attributed network. The resulted dataset has 100,924 nodes and 764,392 edges over 17 time stamps.

ACM: ACM is a similar coauthor network as DBLP. We extract a subgraph consisting of the authors who publish at least three papers in the year of 1995 and 2015, and apply the same mechanism as before to extract 100 important node attributes. Therefore, we obtain a dynamic attributed network with a total amount of 122,567 nodes and 1,551,554 edges over 16 different time stamps.

4.2 Baseline Methods

To verify the effectiveness and efficiency of the proposed SLIDE framework, we compare SLIDE with the following baseline link prediction methods from three different categories: (1) with only network structure; (2) with only node attributes; and (3) with both sources of information.

- Common Neighbors (CN) [37]: CN quantifies the number of common users between node pairs for link prediction.
- Jaccard Coefficient (JC) [37]: JC calculates the similarity of pairs of nodes for link prediction with Jaccard coefficient.
- Adamic-Adar (AA) [37]: AA is an extension of CN which penalizes the common neighbors with high node degrees.
- Rooted PageRank [49]: It performs random walk with start from a root node and then determines the scores of links, i.e., node proximity, to other nodes from the root node.
- NMF [40]: It conducts non-negative matrix factorization on the adjacency matrix of the network to calculate the scores of unconnected node pairs.
- SimAttr [53]: It calculates cosine similarity on node attributes and uses the similarity score to rank links.
- FactLog [44]: It adopts matrix factorization and incorporates both network structure and node attributes in a joint framework for link prediction.
- AttriRank [22]: It performs PageRank on the attributed networks and then the score of each node pair is determined as the product of the PageRank scores of two end nodes.

Among them, CN, JC, AA, Rooted PageRank and NMF belong to the first category by using only network information; SimAttr on the other hand only takes advantage of node attribute information for link prediction; FactLog, AttriRank and the proposed SLIDE are in the third category by combining both sources of information together for link prediction.

4.3 Experimental Settings

In the experiments, we attempt to answer two research questions: (1) how accurate is the proposed SLIDE framework in predicting missing links; (2) how efficient is the proposed streaming algorithm when measured against other offline methods.

We first investigate the effectiveness of the proposed framework SLIDE. Given a dynamic attributed network with T different time stamps, for each time stamp t ($1 \leq t \leq T$), in the training phase, we first perform link prediction with the attributed network G^t, and then test the link prediction performance on G^{t+1}. It should be noted that as most of these baseline methods cannot handle cold-start nodes, we choose to predict the missing links for the nodes that appear in both G^t and G^{t+1}. More investigation on the link prediction for cold-start nodes will be presented later. As a

final result, we output the average link prediction performance over $T-1$ test periods. In the experiments, we set the number of columns in the sketching matrix l according to the suggestions of [23]. Meanwhile, we specify the parameter k the same as l.

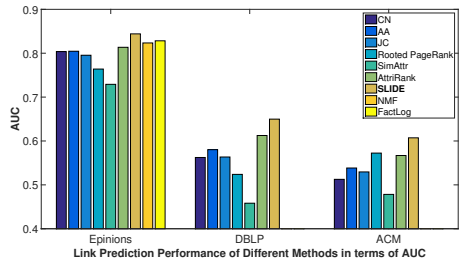
Suppose the number of new links for testing between time stamp t and $t+1$ is e_t, all these baseline methods can be regarded as a ranking model which returns the top e_t possible links from G_t and then compares with the ground truth links in G^{t+1}. To make a fair comparison between SLIDE and baseline methods, in the evaluation, we do not use the residual error threshold ϵ, instead, we rank the candidate links according to the residual errors. Three commonly used evaluation metrics are used to compare the link prediction performance of different methods. They are *area under the curve* (**AUC**) [9], *mean average precision* (**MAP**) [36], *half-life utility* (**HLU**) [45]. The higher the values of AUC, MAP and HLU are, the better the prediction performance is. Specifically, at each time stamp during the testing phase, we treat all the e_t links that will happen at the next time stamp $t+1$ as positive samples, and the other links as negative links.

Different from SLIDE that only makes one pass of the data to predict missing link on the fly, all baseline methods are offline methods that require the access of the whole historical data each time when changes occur. In other words, they need to explicitly materialize the whole attributed networks in memory before link prediction takes place. To have a fair comparison between SLIDE and the baseline methods in terms of efficiency, we allow the storage of the historical data for baseline methods in memory and compare their cumulative running time over all time stamps.

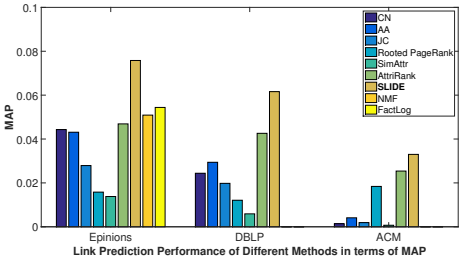
4.4 Effectiveness of the Proposed SLIDE

First, we investigate the effectiveness of the proposed SLIDE framework by comparing its link prediction performance with the aforementioned baseline methods. The average link prediction results over multiple time stamps are presented in Figure 3. We make the following observations from the figure.

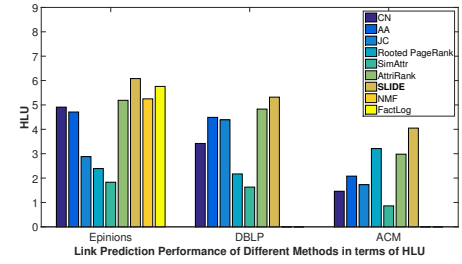
- The proposed streaming link prediction framework SLIDE outperforms all baseline methods in almost all cases. We also perform a pairwise Wilcoxon signed-rank test between SLIDE and these baseline methods. The comparison results indicate that the proposed SLIDE framework is significantly better than others, with a significance level of 0.05.
- CN, AA, JC, Rooted PageRank and NMF only leverage network structure for link prediction, and their performance is superior to SimAttr which relies on node attributes to infer missing links. It implies that the link prediction performance is influenced more by the network structure rather than the node attributes.
- The link prediction methods FactLog, AttriRank and SLIDE that leverage two sources of information achieve better link prediction performance than methods with only one source of information. The observation supports the assumption that node attribute information compliments to network structure for link prediction.
- We do not report the link prediction results of NMF and FactLog on DBLP and ACM datasets as we run out of memory for these two methods. The reason is that these two methods are both matrix factorization based methods and cannot be easily scaled to large-scale networks.



(a) AUC



(b) MAP



(c) HLU

Figure 3: Link prediction performance comparison.

4.5 Efficiency of the Proposed SLIDE

Now we investigate the second research question about the efficiency of the proposed SLIDE framework. Specifically, we report the cumulative running time of different methods across all time stamps in Table 1. As all the baseline methods mentioned are designed for static networks by assuming the materialization of the network structure in memory, we have to rerun these baseline methods repeatedly each time when there are changes on the attributed networks. As can be observed from the table, our proposed SLIDE framework is significantly faster than all baseline methods. The overall running time of SLIDE on Epinions, DBLP and ACM are 25.67 seconds, 291.31 seconds and 689.93 seconds, respectively. Specifically, SLIDE is 49×, 243×, 58×, 12×, 30×, 10×, 278×, and 81× faster than CN, AA, JC, Rooted PageRank, NMF, SimAttr, FactLog, AttriRank, respectively in Epinions. On DBLP and ACM datasets, the cumulative running time of all baseline methods cost more than 3 hours while our method finishes within minutes. In addition to that, as our proposed SLIDE framework maintains and updates a

Table 1: Running time comparison of different methods.

	Epinions	DBLP	ACM
CN	1245.41s	> 3 hours	> 3 hours
AA	6243.32s	> 3 hours	> 3 hours
JC	1489.81s	> 3 hours	> 3 hours
Rooted PageRank	305.41s	> 3 hours	> 3 hours
NMF	774.7s	> 3 hours	> 3 hours
SimAttr	259.09s	> 3 hours	> 3 hours
FactLog	7140.18s	> 3 hours	> 3 hours
AttriRank	2081.53s	> 3 hours	> 3 hours
SLIDE	**25.67s**	**291.31s**	**689.93s**

Table 2: Link prediction results for new users in Epinions.

Metrics	AUC	MAP	HLU
SimAttr	0.6828	0.0286	3.60
AttriRank	0.7067	0.0409	4.29
SLIDE	0.7532	0.0523	4.94

low-rank sketching matrix with light memory overhead, it is also much more space efficient than most baseline methods. All in all, SLIDE achieves promising link prediction performance within a favorable amount of running time with limited memory costs.

4.6 Link Prediction of SLIDE for New Users

It has been widely known that in conventional link prediction problems, new users often suffer from the cold-start problems since we often do not have any data about newly joined users. Fortunately, the rich node attributes can help mitigate this critical issue when link information is not available. To investigate how well the proposed SLIDE framework handles new users for cold-start link prediction problem, we compare SLIDE with SimAttr and AttriRank, as these two methods can also handle the cold-start problem by leveraging node attributes. We focus on the Epinions dataset to investigate the cold-start problem as in DBLP and ACM datasets, authors create coauthor relations with other scholars the same time when they publish a paper and is therefore not suitable for cold-start problem study. In particular, in Epinions, users can first write reviews about products and then build trust relations with others, and we predict missing links for these new users by using their attribute information before they build any trust relations. The link prediction performance comparison in terms of these new users is illustrated in Table 2. It can be shown that SLIDE obtains better link prediction performance than SimAttr and AttriRank for the cold-start problem. The reason is that SLIDE summarizes the connectivity patterns of linked nodes in the sketching matrix; the orthogonal basis from the sketching matrix is noise resilient and can help us predict missing links more accurately.

5 RELATED WORK

In this work, we review related work from three perspectives: (1) link prediction on networks; and (2) mining streaming networks.

5.1 Link Prediction

The past decade has witnessed the development of a great number of link prediction methods [7, 8, 11, 13, 20, 33, 37, 44, 48, 54, 55]. The vast majority of existing link prediction methods can be broadly categorized into two classes - unsupervised methods and supervised methods. A family of unsupervised methods are heavily based on different similarity measures such as the number of common neighbors, Kartz, Jaccard coefficient and Adamic/Adar to measure the node proximity [37, 54]. Another prevalent choice of unsupervised methods is to investigate low-rank matrix techniques to approximate the initial adjacency matrix of the network structure [13, 27]. Supervised methods, on the other hand, treat link prediction as a classification task [6, 39, 44]. Typically, they first extract features from positively labeled instances (existing links) and negatively labeled instances (non-existing links), and then build a classification model to infer the missing labels of a pair of unconnected nodes. The link prediction problem on dynamic networks is also studied in [14, 57]. However, these methods have to store all historical data and cannot be supported in a streaming fashion.

Recent studies imply the existence of autocorrelation between the attributes of connected nodes. To this end, the exploitation of the autocorrelation could advance many mining tasks on attributed networks such as link prediction. For example, [20, 53] first proposed to integrate network structure and node attributes into a joint augmented network and then employ random walk based approaches. Barbieri et al. [7] presented a stochastic generative model to joint factorize social connections and node attributes. The resulted model provides accurate prediction performance and related topical explanations to support the made predictions. Wei et al. [51] further investigated the link prediction problem on networks with partially observable links and node attributes.

5.2 Mining Streaming Networks

Many real-world networks are not static but are presented in a streaming fashion such that a massive amount of interactions among nodes are continuously being received. Such high-velocity edge streams necessitate near real-time analytical methods. In many cases, the size of edge stream could be massive and cannot be easily stored, which further exacerbates the consequent learning tasks. Hence, most of the existing efforts are dedicated to designing effective data structures to summarize the observed network structure in real-time. For example, Aggarwal et al. [4] first proposed to cluster a small graph (or a collection or edges) in a streaming fashion by using the hash-based compression techniques. The similar sketching mechanism is extended to the scenario when node attributes are attached to the continuously generated nodes [56]. In [3], a reservoir sampling method is presented to maintain the structural summary of the underlying network stream for clustering and outlier detection. In terms of classification, Aggarwal [2] employed a min-hash based approach to model the dependencies between the sketched subgraphs and the class labels. Some other efforts tried to leverage the embedding or hashing techniques with kernel methods for classification in streaming networks [21, 25]. In addition, some fundamental graph mining problems such as counting triangles [12], graph matching [41] and eigen-tracking [10] are also extensively studied and a comprehensive overview of these problems could be referred to [1].

6 CONCLUSIONS

A vast majority of existing link prediction algorithms are designed for static networks and assume that the whole network structure is materialized in memory before link prediction happens. However,

many real-world networks are naturally dynamic and are characterized by frequent updates. The updates are often transient and could even be infinite, which puts the applicability of conventional link prediction algorithms in jeopardy. In addition to that, rich node attributes are prevalent and often have a strong connection with the network topology, and they may also change adaptively over time. It remains a daunting task to support the link prediction on such dynamic attributed networks in an online fashion due to some unique challenges. In this paper, we study the novel problem of streaming link prediction on dynamic attributed networks and propose a sophisticated link prediction framework - SLIDE. In particular, we leverage a cost-effective matrix sketching technique to make a summarization of the current observed data by making only one pass of the data, and the sketching matrix, in turn, is used to infer the missing links. Theoretical analysis indicates that our method achieves a similar performance as a cost offline method which stores all historical data. We also perform empirical experimental evaluations on real-world datasets, the results imply that SLIDE not only can predict the missing links more accurately but also is much more computationally efficient than competitors.

ACKNOWLEDGEMENTS

This material is based upon work supported by, or in part by, the National Science Foundation (NSF) grant 1614576, and the Office of Naval Research (ONR) grant N00014-16-1-2257.

REFERENCES

[1] Charu Aggarwal and Karthik Subbian. 2014. Evolutionary network analysis: A survey. *CSUR* (2014).
[2] Charu C Aggarwal. 2011. On Classification of Graph Streams. In *SDM*.
[3] Charu C Aggarwal, Yuchen Zhao, and S Yu Philip. 2011. Outlier Detection in Graph Streams. In *ICDE*.
[4] Charu C Aggarwal, Yuchen Zhao, and Philip S Yu. 2010. On Clustering Graph Streams. In *SDM*.
[5] Mohammad Al Hasan and Mohammed J Zaki. 2011. A Survey of Link Prediction in Social Networks. In *Social Network Data Analytics*.
[6] Lars Backstrom and Jure Leskovec. 2011. Supervised Random Walks: Predicting and Recommending Links in Social Networks. In *WSDM*.
[7] Nicola Barbieri, Francesco Bonchi, and Giuseppe Manco. 2014. Who to Follow and Why: Link Prediction with Explanations. In *KDD*.
[8] Shiyu Chang, Guo-Jun Qi, Charu C Aggarwal, Jiayu Zhou, Meng Wang, and Thomas S Huang. 2014. Factorized Similarity Learning in Networks. In *ICDM*.
[9] Shiyu Chang, Yang Zhang, Jiliang Tang, Dawei Yin, Yi Chang, Mark A Hasegawa-Johnson, and Thomas S Huang. 2016. Positive-Unlabeled Learning in Streaming Networks. In *KDD*.
[10] Chen Chen and Hanghang Tong. 2015. Fast Eigen-Functions Tracking on Dynamic Graphs. In *SDM*.
[11] Chen Chen, Hanghang Tong, Lei Xie, Lei Ying, and Qing He. 2016. FASCINATE: Fast Cross-Layer Dependency Inference on Multi-Layered Networks. In *KDD*.
[12] Lorenzo De Stefani, Alessandro Epasto, Matteo Riondato, and Eli Upfal. 2016. TRIÈST: Counting Local and Global Triangles in Fully-Dynamic Streams with Fixed Memory Size. In *KDD*.
[13] Liang Duan, Charu Aggarwal, Shuai Ma, Renjun Hu, and Jinpeng Huai. 2016. Scaling up Link Prediction with Ensembles. In *WSDM*.
[14] Daniel M Dunlavy, Tamara G Kolda, and Evrim Acar. 2011. Temporal Link Prediction using Matrix and Tensor Factorizations. *TKDD* (2011).
[15] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. 2005. On Graph Problems in a Semi-Streaming Model. *TCS* (2005).
[16] Huiji Gao, Xufei Wang, Jiliang Tang, and Huan Liu. 2013. Network Denoising in Social Media. In *ASONAM*.
[17] Lise Getoor and Christopher P Diehl. 2005. Link Mining: A Survey. *SIGKDD Explorations* (2005).
[18] Mina Ghashami, Amey Desai, and Jeff M Phillips. 2014. Improved Practical Matrix Sketching with Guarantees. In *ESA*.
[19] Mina Ghashami and Jeff M Phillips. 2014. Relative Errors for Deterministic Low-Rank Matrix Approximations. In *SODA*.
[20] Neil Zhenqiang Gong, Ameet Talwalkar, Lester Mackey, Ling Huang, Eui Chul Richard Shin, Emil Stefanov, Elaine Runting Shi, and Dawn Song. 2014. Joint Link Prediction and Attribute Inference using a Social-Attribute Network. *TIST* (2014).
[21] Ting Guo, Lianhua Chi, and Xingquan Zhu. 2013. Graph Hashing and Factorization for Fast Graph Stream Classification. In *CIKM*.
[22] Chin-Chi Hsu, Yi-An Lai, Wen-Hao Chen, Ming-Han Feng, and Shou-De Lin. 2017. Unsupervised Ranking using Graph Structures and Node Attributes. In *WSDM*.
[23] Hao Huang and Shiva Prasad Kasiviswanathan. 2015. Streaming Anomaly Detection Using Randomized Matrix Sketching. *VLDB* (2015).
[24] Hao Huang, Shinjae Yoo, and Shiva Prasad Kasiviswanathan. 2015. Unsupervised Feature Selection on Data Streams. In *CIKM*.
[25] Ling Jian, Jundong Li, and Huan Liu. 2017. Toward Online Node Classification on Streaming Networks. *DMKD* (2017).
[26] Takuya Kitazawa. 2017. Sketching Dynamic User-Item Interactions for Online Item Recommendation. In *CHIIR*.
[27] Jérôme Kunegis and Andreas Lommatzsch. 2009. Learning Spectral Graph Transformations for Link Prediction. In *ICML*.
[28] Timothy La Fond and Jennifer Neville. 2010. Randomization Tests for Distinguishing Social Influence and Homophily Effects. In *WWW*.
[29] Jundong Li, Harsh Dani, Xia Hu, and Huan Liu. 2017. Radar: Residual Analysis for Anomaly Detection in Attributed Networks. *IJCAI* (2017).
[30] Jundong Li, Harsh Dani, Xia Hu, Jiliang Tang, Yi Chang, and Huan Liu. 2017. Attributed Network Embedding for Learning in a Dynamic Environment. In *CIKM*.
[31] Jundong Li, Xia Hu, Ling Jian, and Huan Liu. 2016. Toward Time-Evolving Feature Selection on Dynamic Networks. In *ICDM*.
[32] Jundong Li, Xia Hu, Liang Wu, and Huan Liu. 2016. Robust Unsupervised Feature Selection on Networked Data. In *SDM*.
[33] Jundong Li, Jiliang Tang, Yilin Wang, Yali Wan, Yi Chang, and Huan Liu. 2017. Understanding and Predicting Delay in Reciprocal Relations. *arXiv preprint arXiv:1703.01393* (2017).
[34] Jundong Li, Liang Wu, Osmar R Zaïane, and Huan Liu. 2017. Toward Personalized Relational Learning. In *SDM*.
[35] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A Contextual-Bandit Approach to Personalized News Article Recommendation. In *WWW*.
[36] Yanen Li, Jia Hu, ChengXiang Zhai, and Ye Chen. 2010. Improving One-Class Collaborative Filtering by Incorporating Rich User Information. In *CIKM*.
[37] David Liben-Nowell and Jon Kleinberg. 2007. The Link Prediction Problem for Social Networks. *JASIST* (2007).
[38] Edo Liberty. 2013. Simple and Deterministic Matrix Sketching. In *KDD*.
[39] Ryan N Lichtenwalter, Jake T Lussier, and Nitesh V Chawla. 2010. New Perspectives and Methods in Link Prediction. In *KDD*.
[40] Chih-Jen Lin. 2007. Projected Gradient Methods for Nonnegative Matrix Factorization. *Neural Computation* (2007).
[41] Andrew McGregor. 2005. Finding Graph Matchings in Data Streams. In *APPROX-RANDOM*.
[42] Andrew McGregor. 2014. Graph Stream Algorithms: A Survey. *ACM SIGMOD Record* (2014).
[43] Miller McPherson, Lynn Smith-Lovin, and James M Cook. 2001. Birds of a Feather: Homophily in Social Networks. *Annual Review of Sociology* (2001).
[44] Aditya Menon and Charles Elkan. 2011. Link Prediction via Matrix Factorization. *ECMLPKDD* (2011).
[45] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. 2008. One-class Collaborative Filtering. In *ICDM*.
[46] Steffen Rendle. 2010. Factorization Machines. In *ICDM*.
[47] Gilbert W Stewart. 1990. Matrix Perturbation Theory. (1990).
[48] Yizhou Sun, Jiawei Han, Charu C Aggarwal, and Nitesh V Chawla. 2012. When Will It Happen?: Relationship Prediction in Heterogeneous Information Networks. In *WWW*.
[49] Hanghang Tong, Christos Faloutsos, and Jia-yu Pan. 2006. Fast Random Walk with Restart and Its Applications. In *ICDM*.
[50] Alexey Tsymbal. 2004. The Problem of Concept Drift: Definitions and Related Work. *Computer Science Department, Trinity College Dublin* (2004).
[51] Xiaokai Wei, Linchuan Xu, Bokai Cao, and Philip S Yu. 2017. Cross View Link Prediction by Learning Noise-resilient Representation Consensus. In *WWW*.
[52] Qingyun Wu, Huazheng Wang, Quanquan Gu, and Hongning Wang. 2016. Contextual Bandits in A Collaborative Environment. In *SIGIR*.
[53] Zhijun Yin, Manish Gupta, Tim Weninger, and Jiawei Han. 2010. A Unified Framework for Link Recommendation Using Random Walks. In *ASONAM*.
[54] Peixiang Zhao, Charu Aggarwal, and Gewen He. 2016. Link Prediction in Graph Streams. In *ICDE*.
[55] Tong Zhao, H Vicky Zhao, and Irwin King. 2015. Exploiting Game Theoretic Analysis for Link Recommendation in Social Networks. In *CIKM*.
[56] Yuchen Zhao and Philip S Yu. 2013. On Graph Stream Clustering with Side Information. In *SDM*.
[57] Linhong Zhu, Dong Guo, Junming Yin, Greg Ver Steeg, and Aram Galstyan. 2016. Scalable Temporal Latent Space Inference for Link Prediction in Dynamic Social Networks. *TKDE* (2016).