Group Link Prediction

Andrew Stanhope[1], Hao Sha[1], Danielle Barman[2], Mohammad Al Hasan[1], and George Mohler[1]
[1]Department of Computer & Information Science, Indiana University Purdue University Indianapolis
[2]Department of Mathematics & Computer Science, University of Wisconsin-River Falls

***Abstract*—Due to its universal applications in the domain of social network analysis, e-commerce, and recommendation systems, the task of link prediction has received enormous attention from the data mining and machine learning communities over the last decade. In its original setting, the task only predicts whether a pair of entities who are not connected at present time will form a connection in future. However, in real-life an entity sometimes join a group (or a community), thus making a connection with the group (or the community), instead of connecting with an individual. Existing solutions to link prediction are inadequate for solving this prediction task. To overcome this challenge, in this work we propose a novel problem named group link prediction which focuses on evaluating the likelihood for a candidate to become a member of a group at a given time. The problem has potential applications such as friendship or group suggestions on Facebook or other social networks, as well as co-authorship suggestion, or group email recommendations. To solve the problem, we propose a Long Short-term Memory based model that inputs the embedding vectors of the group and outputs the conditional probability distributions for the candidates. We also introduce a composite long short-term memory model that integrates keyword information. Experimental results on real-world data sets validate the superiority of our proposed model in comparison to various baseline methods.**

***Index Terms*—Link Prediction, Long Short-Term Memory, Graph Embedding, Group Link Prediction**

I. Introduction

Link prediction [1], [2] is a widely studied problem with successful applications in social networks [1], co-authorship networks [3], protein-protein interactions [4] and item recommendation [5]. For a social network, such as, Facebook, link prediction would view the network as a graph where the users are nodes and their "friendships" are edges. Given an existing state of a network and a pair of users who are not connected in the existing state, link prediction predicts the likelihood of forming a future connection between those pair of users. Such a solution can then be used by the social network platforms for generating friendship recommendation. Similarly, in a co-authorship network, link prediction evaluates the likelihood of future collaboration between a pair of authors who have never published together before. In the bioinformatics domain, experimentally generated protein-protein interaction networks usually suffer from missing edges (high false-negative rate) [4]. Link prediction in such networks can improve the information quality by recovering the missing edges. In item recommendation, users and items can be represented by nodes in a bipartite network, while their interactions can be viewed as links. Link prediction can then be applied to predict potential user-item interaction.

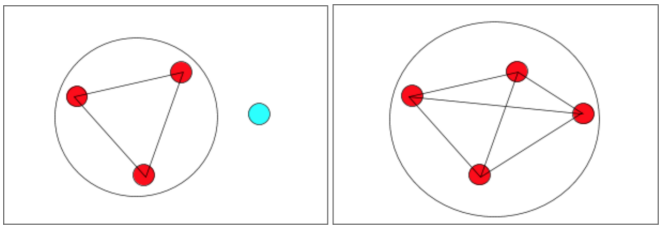


Fig. 1: Group Link Prediction Visual. The red nodes in the circle represent a group while the blue node represents a candidate node. In the second image, the candidate node is integrated into the group with edges forming between it and all of the group members.

The conventional setting of a link prediction task only looks at a pair of entities and evaluates the likelihood of forming a link between them. However, in many real-life scenarios, we are more interested in the possible connections between an individual entity and a group of entities. For instance: a group of researchers may like to expand their network and identify a potential collaborator; An event organizer may want to send out invitations to those who are likely to join the current group of participants; Social network platforms (Say, Facebook or LinkedIn) may want to build a platform app, which recommends a potential person to a group leader; An email application may suggest a missing person's email to a group email-list. In all these applications, the task is to predict a potential entity which would form a link to a given group, not to an individual entity—existing link prediction framework cannot be readily applied to these problems. Therefore, in this work, we formalize a novel problem on network, namely *group link prediction* and propose a recurrent neural network (RNN) [6], [7] based model for solving it. In specific, we define a group as a collection of at least two distinct entities; group link prediction then identifies entities that are most likely to form a link to the given group. The above formulation can also be used for recovering missing members of a given group. Note that, in our group link formulation, we focus on the cases where there is only one member missing from the group. An illustration of our problem formulation is given in Fig. 1. The big circle (left panel) represents a group of three members (red nodes), and we identify the blue node as the missing member; adding it back to the group generates four-member group in the right panel.

Traditionally, link prediction solutions use various topolog-

978-1-7281-0858-2/19/$31.00 ©2019 IEEE

ical similarity measures which can be used for computing a similarity value between two nodes who are not connected [1], [3], [5]. These similarity measures can be further divided into neighbor-based (such as Common Neighbor, Jaccard's Coefficient, Adamic/Adar and Preferential Attachment) and path-based (such as, Graph Distance, Katz$_\beta$, and hitting time). Given the current state of the network, we can evaluate the similarity measures between any disconnected pairs and predict the pairs with high similarity scores to form a link in the future. Link prediction can also be posed as a binary classification problem with the labels indicating whether two entities are connected (positive) or not (negative) [3]. In this approach, the features are no longer confined to the topological properties, rather domain-specific knowledge such as keyword attributes can also be incorporated. Notably, in recent years, various network embedding methods such as Node2Vec [8], LINE [9], and Graph2Gauss [10] are proposed, which learn node/edge features in unsupervised manner so that they can be used for the downstream supervised link prediction task. Link prediction on dynamic networks has also been considered by some works [11]–[13], where the temporal evolution of a network is factored in the prediction task through temporal node embedding. None of these solutions are suitable for the task of group link prediction, which is the focus of this work.

In this work, we propose a group link prediction model by using long short-term memory (LSTM) [7] network. The proposed model is able to utilize temporal interactions of nodes for obtaining embedding of nodes in a continuous time domain, which is useful for better link prediction. Besides, it enables an effective approach for obtaining the embedding of a group from the embedding of nodes. Compared to the dynamic network embedding methods, our LSTM approach has the advantage of being an end-to-end model, where the embedding vectors and the classifier are trained simultaneously and specifically for the group link prediction task. Finally, our LSTM model can take attributes (such as keywords) as inputs, thus allowing the incorporation of domain knowledge within the model. The contributions of this work are three fold:

1) We extend the link prediction problem to predicting the link between an individual and a group. We name this new problem "group link prediction".
2) We propose an LSTM-based model to perform dynamic group link prediction. We also introduce a second LSTM to incorporate entity/group attributes.
3) We compare our model with various baselines for four real-world datasets to show the superiority of the proposed model for predicting the link between an individual entity and a given group.

The rest of the paper is organized as follows. In Section II we discuss the design of the Long Short-term Memory (LSTM) method and give background on network embedding. In section Section III, we review related works in the area of link and group link prediction. Section IV gives detailed description of our method. In Section V, we describe the experiments and present the results. Finally, we summarize our work in section VI.

II. Background

A. Long Short-Term Memory Architecture

Long Short-Term Memory (LSTM) [7] is a special kind of Recurrent Neural Network (RNN) [6], for modeling dynamic behaviors. It resolves the vanishing/exploding gradient problem of traditional RNNs [14], [15] by introducing a cell state. Thus, the current states are collectively determined by the current input and the previous hidden and cell states. Such design effectively reduces the multiplicative effect of the small or large gradients. LSTM has been successfully applied to various sequence related learning problems, such as, speech recognition [16], language translation [17], handwriting synthesis [18], and image generation [19]. Here we adopt an LSTM whose cell is defined as the following:

$$
\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{V}_i\mathbf{c}_{t-1} + \mathbf{b}_i),\\
\mathbf{f}_t &= \sigma(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{V}_f\mathbf{c}_{t-1} + \mathbf{b}_f),\\
\mathbf{c}_t &= \mathbf{f}_t\mathbf{c}_{t-1} + \mathbf{i}_t \odot tanh(\mathbf{W}_c\mathbf{x}_t + \mathbf{U}_c\mathbf{h}_{t-1} + \mathbf{b}_c),\\
\mathbf{o}_t &= \sigma(\mathbf{W}_o\mathbf{x}_t + \mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{V}_o\mathbf{c}_t + \mathbf{b}_o),\\
\mathbf{h}_t &= \mathbf{o}_t \odot tanh(\mathbf{c}_t),
\end{aligned}
\tag{1}
$$

where σ is the sigmoid function and $\odot$ is the Hadamard product. $\mathbf{x}_t$, $\mathbf{h}_t$, and $\mathbf{c}_t$ are the input, the hidden state, and the cell state, respectively. $\mathbf{i}_t$, $\mathbf{f}_t$, and $\mathbf{o}_t$ are intermediate states.

B. Network Embedding

Many real-life datasets are in the form of networks or can be converted to networks. As networks can contain billions of nodes and edges, it would be intractable to perform complex inferences over the entire network. To alleviate this difficulty, various graph embedding methods have been proposed. In this work, we focus on the methods that embed nodes, while other methods embed edges [20] and (sub)graphs [21]. Node embedding methods essentially map nodes to some low dimensional latent space, making it possible to use existing machine learning algorithms for the downstream tasks (e.g. link prediction, classification, or community detection). Traditionally, embedding vectors are obtained by applying dimensionality reduction (such as PCA [22], MCA [23], and Laplacian Eigenmaps [24]) on the adjacency matrix. Inspired by the recent developments in neural language models [25], a series of deep learning methods [8], [9], [26] are proposed to learn node embeddings in the network context. The DeepWalk model [26] first samples a collection of short random walks and then apply the Skip-gram [25] framework to learn the node embeddings. LINE [9] and Node2Vec [8] are extentions to DeepWalk, as the former uses a breadth-first search and later combines both breadth-first search and the depth-first search. In this work, we adopt LINE and Node2Vec to learn node embeddings in our baseline methods.

III. Related works

The link prediction problem has gotten a lot of attention in recent years. One of the most effective and cost-efficient ways to analyze the link prediction problem is to employ network embeddings [27]. There have been a number of network embedding methods proposed, ranging from static embeddings like DeepWalk [26], Node2Vec [8], and LINE [9], to dynamic embeddings like DynamicTriad [12] and TNE [13]. In additon, attribute embeddings such as Graph2Gauss [10] and Neural-Brane [28] have been proposed to incorporate node attributes. Despite their advantages, static models are not able to capture the time dependency in the dataset. Although the dynamic methods alleviate this issue, they do not allow attributes and solely depend on the network topology. Recently, a dynamic attributed network embedding model, DANE [29], has been proposed. Nevertheless, it is not an end-to-end model specifically designed for the group link prediction task. To the best of our knowledge, this is the first work that formally defines the group link prediction problem and provides a novel LSTM-based approach that enables end-to-end training.

IV. Methods

A. Problem Description

We are given a set of n members $S = \{v_1, v_2, ..., v_n\}$, where the subscript of v represents identifier of the members. Also given a sequence of events, $\{(t_1, s_1), (t_2, s_2), \cdots, (t_i, s_i)\}$, in which each event (say, i) has an event time-stamp, t_i and the member-list, $s_i \subseteq S$, who collectively perform the event. For an event s_i, we use $s_{i,1}, s_{i,2}, \cdots, s_{i,k}$ to denotes the ids of the k members of the event i. Now, for a future event (say, $n+m$), the task of group link prediction is to predict the missing member $v_{s_{n+m},j}$ given that, $v_{s_{n+m},1}, v_{s_{n+m},2}, ..., v_{s_{n+m},j-1}$ are known to participate in the event $n+m$.

B. Data Processing

We organized the datasets into sequences of time-group pairs $\{(t_i, s_i)\}$, sorted in time ascending order (i.e. $t_{i-1} < t_i$). Now we split each data set into a training, validation, and test set (split 80-10-10). We further filtered out the group members that do not appear in the training set from the validation and test sets and we removed the data points with less than three members in each set. Let S_{train}, S_{valid}, and S_{test} denote the set of members in the training set, validation set, and test set, respectively. The resulting data sets then satisfy (1) $S_{valid} \subseteq S_{train}$ (2) $S_{test} \subseteq S_{train}$, and (3) $S = S_{train}$.

C. Keyword Extraction

For the DBLP and Enron email datasets we extracted keywords from the article titles and email bodies, respectively, to obtain attributes. We followed the typical natural language processing process - tokenizing the text, stemming/lemmatization and removing filler words as well as common stopwords. We then ranked the words by their global frequencies and retrieved the keywords based on the frequencies. Then we assigned the keywords to the corresponding authors or email addresses. The finalized keywords are presented in Fig. 2. Notice that for the DBLP dataset, we also extracted key-phrases of size 2, 3 and 4 using the POS tagging method suggested in [30].

DBLP top 4 one, two, three and four word phrases

system: 19190	wireless sensor network: 1669
network: 16171	cognitive radio network: 700
data: 16070	massive mimo system: 524
model: 12599	multi agent system: 445
neural network: 5870	deep convolutional neural network: 361
real time: 2421	non orthogonal multiple access: 275
social network: 1661	vehicular ad hoc network: 213
large scale: 1556	cloud radio access network: 175

Enron top 6 keywords

corp: 5021	price: 3705	pm: 3653
power: 3589	gas: 3549	market: 3289

Fig. 2: Top keywords for DBLP and Enron data sets. The DBLP data set had one, two, three and four word phrases while Enron only had single word keywords. The number displayed with each word is the global frequency which is the number of times that word showed up in the text.

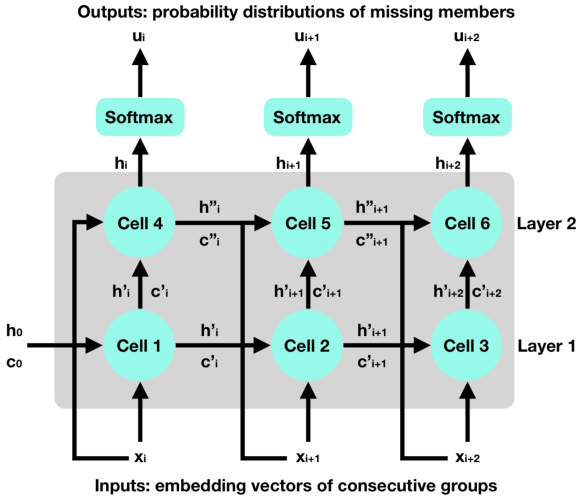


Fig. 3: Network Architecture for the LSTM model without keywords. The shaded area represents LSTM while the cyan circles represent cells. For illustration purpose, we only show two layers and six cells. $\mathbf{x}_{i+j}$ denotes the j'th input to the LSTM. $\mathbf{h}_{i+j}$ and $\mathbf{c}_{i+j}$ denote the hidden states and cell states respectively. The hidden states of the last layer are fed to the fully-connected layer with Softmax action. $\mathbf{u}_{i+j}$ denotes the j'th output, i.e. the probability of the missing members at $i+j$ time-step.

D. Model Formulation

LSTM without attributes. We proposed a multi-layer LSTM architecture which is illustrated in Fig. 3. The LSTM (shaded area in Fig. 3) contains a stack of recurrently connected memory cells. Note that the number of layers and cells are adjustable and not necessarily the same as those in Fig. 3. Summarized in Eq. 2, a cell in LSTM takes an input $\mathbf{x}_i$, a hidden state $\mathbf{h}_i$ and a cell state $\mathbf{c}_i$, and outputs a new hidden state $\mathbf{h}_{i+1}$ and a new cell state $\mathbf{c}_{i+1}$. As shown in Fig. 3, these new states are then passed horizontally to the next cell in the

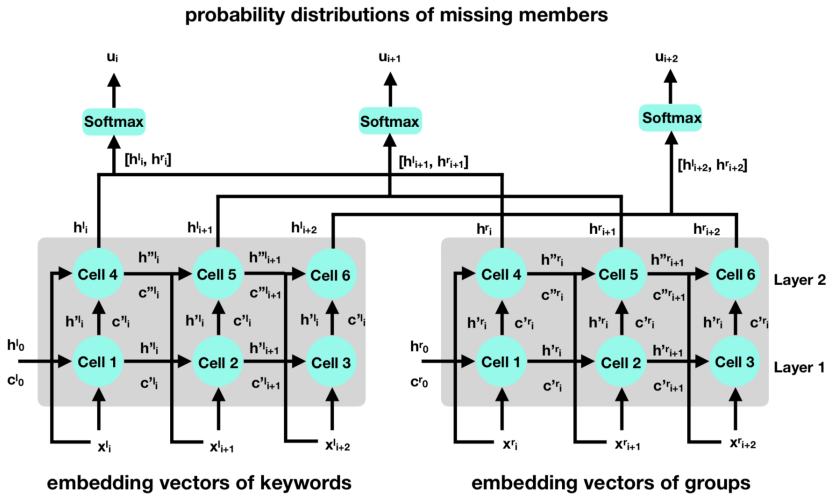


Fig. 4: Network Architecture for the LSTM model with keywords. The shaded areas represent LSTMs while the cyan circles represent cells. The left LSTM takes embedding vectors of keywords as input; while the right LSTM takes embedding vectors of groups as input. $\mathbf{x}_{i+j}^{l(r)}$ denotes the j'th input to the LSTM. $\mathbf{h}_{i+j}^{l(r)}$ and $\mathbf{c}_{i+j}^{l(r)}$ denote the hidden states and cell states respectively. The hidden states of the last layers of the LSTMs are concatenated and fed to the fully-connected layer with Softmax action. $\mathbf{u}_{i+j}$ denotes the j'th output, i.e. the probability of the missing members at $i+j$ time-step.

same layer and vertically to the cell in the next layer. The final outputs of the LSTM are fed to a fully-connected layer with weight $\mathbf{W}$, bias $\mathbf{b}$, and a Softmax activation, to generate a predictive distribution $\mathbf{u}_i$ over the candidates for the missing group member.

$$\begin{aligned}(\mathbf{h}_{i+1}, \mathbf{c}_{i+1}) &= cell_{LSTM}(\mathbf{x}_i, \mathbf{h}_i, \mathbf{c}_i) \\ \mathbf{u}_i &= Softmax(\mathbf{W}\mathbf{h}_i + \mathbf{b})\end{aligned} \tag{2}$$

The input of the LSTM architecture is a series of embedding vectors representing consecutive groups. A member (candidate) is mapped to a d dimensional embedding vector that can be learned upon training. Given a group at time t_i, we randomly hold out a member as our target, and sum the embedding vectors of the other members as $\mathbf{x}_i$. The vectors $\mathbf{x}_i, \mathbf{x}_{i+1}, \mathbf{x}_{i+2}, ..., \mathbf{x}_{i+k-1}$, denoting k consecutive groups, are fed to the corresponding cells. Note that the hidden states and the cell states are also d dimensional vectors. The outputs are the hidden states of the last layer, $\mathbf{h}_i, \mathbf{h}_{i+1}, ..., \mathbf{h}_{i+k-1}$. They then go through a fully-connected layer with Softmax activation to generate a series of vectors $\mathbf{u}_i, \mathbf{u}_{i+1}, ..., \mathbf{u}_{i+k-1}$, representing the probability distributions over the candidates for the missing members.

Let $\mathbf{y}_i, \mathbf{y}_{i+1}, ..., \mathbf{y}_{i+k-1}$ denote the one-hot encoding of the actual missing members. The cross-entropy loss can thus be written as the following:

$$L = -\frac{1}{k}\sum_{j=1}^{k}\sum_{l=1}^{N} y_{i+j}^{l} log(u_{i+j}^{l}), \tag{3}$$

where y_{i+j}^{l} and u_{i+j}^{l} are the l'th elements of $\mathbf{y}_{i+j}$ and $\mathbf{u}_{i+j}$, respectively. The summation is over the k time-steps and the number of candidates N.

LSTM with attributes. For the datasets where keywords could be extracted, we proposed a network architecture consisting of two LSTMs as illustrated in Fig. 4. The LSTM on the left inputs the embedding vectors of attributes (keywords) attached to the groups at each time step; while the LSTM on the right inputs the embedding vectors for the groups as the model without keywords. The hidden states of the two LSTMs are concatenated at the end and fed to the Softmax layer to generate a probability distribution of the missing members. Then the same loss function (Eq. 3) can be calculated and minimized to update the network parameters.

V. Experiments

We run our proposed model without keywords (Fig. 3) on four real-world datasets and compare the performance with the competing methods. Here we focus on predicting the missing member of a group provided that the rest of the group are known at time t_i. This falls into the definition of group link prediction, as we attempt to predict the link between a candidate and a group. For instance, we attempt to identify one author from a pool of candidates that is most likely to publish a paper with a group of known authors.

To train our model, at time t_i, we randomly shuffle the members in the group s_i, and hold out the last member $v_{i,k}$ as the target. Our model then takes the rest of the group as input, and outputs a conditional probability $u_{i,v} = P(v|v_{i,1}, .., v_{i,k-1}, t_i)$ $\forall v \in S$. The loss defined in Eq. 3 is thus evaluated and minimized for each batch. In the validation set and the test set, we hold out the last member $v_{i,k}$ of the group at time step t_i without shuffling. We then rank the candidates based on the conditional probabilities $P(v|v_{i,1}, .., v_{i,k-1}, t_i)$, given by our model. As a performance measure, we evaluate the hit rates based on the ranking and the true target. For instance, if the target member is among the five highest ranking candidates, we count it as a hit for the Hit@5 rate. We estimate the mean hit rates on the validation set to determine the optimal set of

TABLE I: Data set properties. Number of Events gives the number of events in each data set (E.g. number of papers, emails, or shopping carts). $|V|$ and $|E|$ are the number of nodes and number of edges for each network, respectively.

Data set	Number of Events	$\lvert V\rvert$	$\lvert E\rvert$
DBLP	290483	70431	645018
Enron	13011	3153	17903
SFHH	3331	405	128976
Hypertext	2960	111	3065

hyper-parameters. Lastly, we run the model with the optimal hyper-parameters on the test set and calculate the mean hit rates as the ultimate measure of the performance.

In addition, we perform the similar task using the LSTM model with keywords (Fig. 4). The keywords are obtained from the text attached to the groups using the procedure described earlier. For example, for the Enron Email dataset, we extracted keywords from the email texts and assigned keywords to the participants of the email conversation. Then we compared the hit rates to those given by the model without keywords, to examine whether or not including keywords could improve performance.

We also performed additional experiments for validating model convergence, and robustness. In the following sections, we describe the data sets on which we run our experiments and the competing models with which we compare our LSTM model.

A. Data Description

DBLP: The DBLP data set contains bibliographical information of articles published in major computer science journals and proceedings. For our experiments, we selected articles published between the years of 2015 and 2019. We extracted year, number, title and authors from the data set and then constructed time stamps using the combination of year and number. We sorted the papers first by the time stamps, then by title and authors for articles with the same time stamp. We further filtered out papers with non-English titles and removed duplicate articles. We also removed the papers with less than three authors as we are studying groups and two authors are not considered to be a group. The resulting sequence contained $290,483$ articles.

Enron Email: The Enron email data set contains $\sim 500,000$ emails generated by employees of the Enron Corporation. The emails from 2000 to 2001 were adopted here. We extracted the email addresses and time stamp from each email across the selected time period. The emails were sorted by time in ascending order and the punctuation was removed from all of the email bodies. To construct a network, we used email addresses of Enron employees as nodes, and added edges between nodes with at least one email exchanged between them. The resulting sequence contained $13,011$ email exchanges.

SFHH Interactions: The SFHH conference data contains a set of spacial interactions between individuals at the 2009 SFHH conference [31]. To construct the network we considered groups of individuals in the same spacial area for some interval of time to be a group, and a timestamp was associated with that group based on the time that they all entered same spacial area . The resulting data is comprised of 3331 groups.

Hypertext 2009 Contact Network: The Hypertext data contains a set of spacial interactions between individuals the ACM Hypertext 2009 conference [32]. As with the SFHH data, to construct the network we considered groups of individuals in the same spacial area for some interval of time to be a group, and a timestamp was associated with that group based on the time that they all entered same spacial area . The resulting data is comprised of 2960 groups.

B. Competing Methods

For the following five models, we converted our data into a graph where edges are formed between nodes which have appeared together in a group (e.g. authors who have published a paper together, or email addresses that have been a part of an email chain together). For the embedding-based approaches, we calculated the centroid of the embedding vectors of a given group and measure the Euclidean distance between the embedding of a candidate node to the centroid. The nodes with the smallest distances are selected as most likely to form a link to the group.

Common Neighbor: In this approach, we first find out the set of nodes that link to at least one of the group members. We then rank these nodes by the number of group members connected to them. The top k nodes are selected from the set as the k most likely candidates to form a link to the given group. Using the DBLP dataset as an example, we collect a set of authors who have published with one of the group members before but currently not in the group. We then rank the authors by the number of "common neighbors" in the given group and select the top ranking authors.

Node2Vec: We used Node2Vec to embed the nodes from our graphs into a 20 dimensional vector representation. Then, using our training set, we removed a single author from each group and attempted to predict the missing author using the remainder of the group. This was accomplished using a look-up table for each of the authors in the group to find their vector representations, which we then used to compute the centroid of the group. We then sorted each node from the graph by distance to the centroid of the group and looked for the missing node within the closest k nodes. Then, with each of the groups in the testing set considered together, the hit rate at k is the percentage of times that the missing node showed up in the k closest nodes to the centroid. The return and in-out hyperparameters were tuned using a grid search over $p, q \in 0.25, 0.5, 1, 2, 4$ as recommended by the original Node2Vec publication.

LINE: We used the same approach as Node2Vec with LINE. However, while Node2Vec creates embeddings using random walks on the graph, LINE attempts to learn embeddings for the nodes using an optimizer function. In the case of LINE, nodes are expected to be close to each other in

TABLE II: Experimental Results.

		Hit@5	Hit@10	Hit@20
DBLP	Common Neighbor	**0.571**	**0.647**	**0.700**
	LSTM	0.516	0.551	0.580
	Node2Vec	0.350	0.405	0.432
	Graph2Gauss	0.242	0.298	0.342
	LINE	0.197	0.239	0.273
Enron	Common Neighbor	0.262	0.380	0.521
	LSTM	**0.416**	**0.502**	**0.581**
	Node2Vec	0.256	0.381	0.484
	Graph2Gauss	0.267	0.378	0.489
	LINE	0.129	0.203	0.287
SFHH	Common Neighbor	0.018	0.021	0.063
	LSTM	**0.076**	**0.121**	**0.163**
	Node2Vec	0.006	0.018	0.036
	Graph2Gauss	0.009	0.015	0.027
	LINE	0.015	0.021	0.045
HT09	Common Neighbor	0.133	0.251	0.422
	LSTM	**0.360**	**0.446**	**0.568**
	Node2Vec	0.118	0.190	0.312
	Graph2Gauss	0.148	0.224	0.297
	LINE	0.205	0.236	0.270

the vector embedding space if they are first or second degree neighbors.

Graph2Gauss: The same approach to Node2Vec was taken with Graph2Gauss as well, but Graph2Gauss had the added benefit of being able to create embeddings for an attributed graph. We ran two variants of the test using Graph2Gauss: one with and one without attributes. We did this so that we would be able to determine if the attributes we were using for these data sets would improve hit rates or if they would act as noise.

C. Hyper-parameter tuning

For the LSTM model, we tune the number of time-steps in $\{64, 128, 256\}$ and the embedding dimension in $\{128, 256, 512\}$. We used a batch size of 32, a learning rate of 1.0, and a 2 layer LSTM architecture. The number of cells in each LSTM layer is the same as the number of time-steps. We chose the hidden dimension to be the same as the embedding dimension. To prevent over-fitting, we used a dropout value of 0.5.

D. Results

Group link prediction. In Table II, we show the results of the group link prediction task for our LSTM model without keywords along with the competing methods over the four datasets. We used hit rates at 5, 10, and 20 as measures of the performance. Note that the results here are evaluated using the test set. We can see that the LSTM model outperforms the other methods except for the DBLP dataset. The reason why the Common Neighbor method performs better than the LSTM method in the DBLP dataset is likely because the co-authorship in the DBLP data set is very stable and the structural information plays the primary role in determining the link between an author and a group. For instance, the authors who have published together before are most likely to publish together again. In contrast, the participants of an email conversation (Enron) or a social contact (SFHH and HT09) are much more dynamic. For instance, employees often send out emails involving different groups of people given different topics. Therefore, for these three datasets, our LSTM method outperforms the other methods by a large margin.

TABLE III: Hit rates with and without keywords. The LSTM results here are obtained using time-step 64 and hidden dimension 128 on the Enron Email data set.

Method	Hit@5	Hit@10	Hit@20
LSTM w/o keywords	0.365	**0.472**	**0.589**
LSTM w/ keywords	**0.390**	0.466	0.554

TABLE IV: LSTM: time-step sensitivity. The results are Hit@5 rates with different time steps.

Data set	64	128	256
DBLP	0.558	0.541	**0.583**
Enron	0.501	0.499	**0.518**

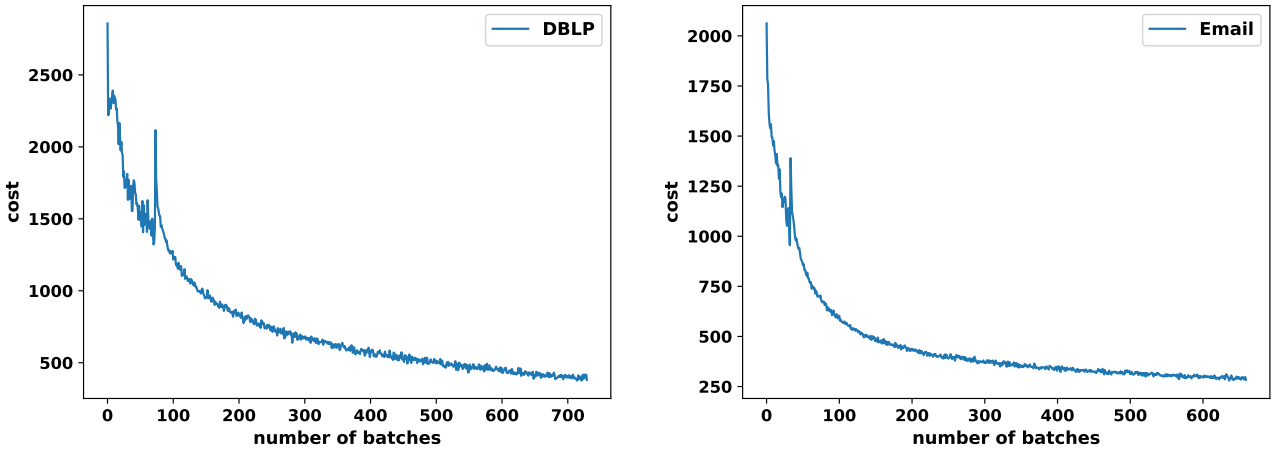
Performance with keywords. We compare the models with and without keywords in terms of hit rates in Table III. The experiments were performed using the Enron Email data set. The keywords were extracted from the email texts following the procedure described in Sec. IV-C. Both models used 64 time-steps and a hidden dimension of 128. We also kept the other hyper-parameters identical for both models to ensure a fair comparison. In Table III, we can see that the model without keywords outperforms the one with keywords, except for the Hit@5 case. This indicates that including keywords might introduce noise instead of providing useful information to help identify potential group members. Moreover, when we ran Graph2Gauss, the only competing model that accepted node attributes, we found that the attributed version performed far worse than the corresponding non-attributed model. This coincides with our results from the attributed and non-attributed versions of the LSTM model.

Model convergence. In Fig. 5, we plot the cost for the LSTM model without keywords for the DBLP and the Enron datasets. The cost is the cross-entropy loss defined in Eq. 3. The curves here indicate a convergence of the training data. A few spikes appear during the early part of the training data, which is likely the result of a small batch size (= 32).

Hyper-parameter sensitivity. We examine the impact on performance of changing some key hyper-parameters. We vary one hyper-parameter at a time, while keeping the other hyper-parameters as their optimal values. We use the Hit@5 rate to indicate the performance. In Table IV, we show the Hit@5 rates with different time-steps for each data set. Note that the results here are evaluated using the validation set. The results indicate that a larger time-step renders better performance. We also test the affect of tuning the embedding dimensions in Table V. We can see that the best performance is not

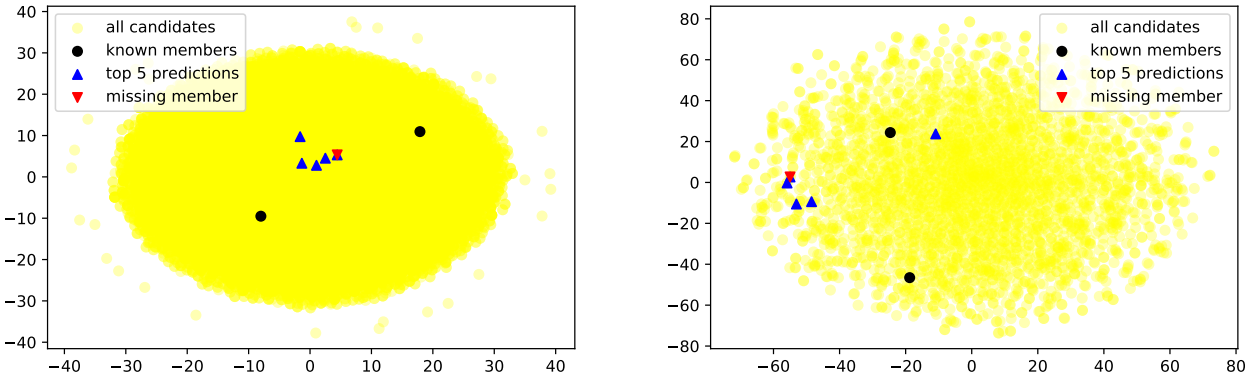
TABLE V: LSTM: embedding dimension sensitivity. The results are Hit@5 rates with different embedding dimensions.

Data set	128	256	512
DBLP	0.360	0.548	**0.583**
Enron	0.510	**0.518**	0.465



(a) DBLP (b) Enron

Fig. 5: Cost for LSTM with different data sets.



(a) DBLP (b) Enron

Fig. 6: Embedding vectors projected onto 2D space. The yellow circles represent the entire set of candidates, while the black circles represent the group members we already know. The blue triangles are the top 5 predicted members and the red triangle denotes the target (missing) member.

necessarily achieved at the highest dimension.

Quality of embedding vectors. In Fig. 6, we plot the 2D projection of the embedding vectors learned from the LSTM model for the DBLP and the Enron datasets. The figures are snapshots taken when a hit occurs. The yellow circles represent the entire set of candidates while the black circles are the members in a given group. The red triangles are target (missing) members and the blue triangles are the top 5 most likely candidates given by our model. We can see from the figures that the predicted nodes are in the vicinity of the target nodes, suggesting that the embeddings given by the LSTM model successfully capture the similarities among the nodes in the datasets.

VI. Conclusions

In this work, we proposed a new problem - group link prediction. Unlike the traditional link prediction which predicts the link between two individuals, our task was to predict the link between an individual and a group. To solve the problem, we proposed two LSTM based models: one with and one without attributes. The models learned a conditional probability distribution over the candidates given a group of known entities at a certain time step. We compared our model without keywords with a series of competing methods over four real-world data sets. Our model shows superior performance for the datasets where the group organizations are constantly changing. In addition, we found that the LSTM model without keywords is more effective than the one with keywords, as the keywords may introduce noise and reduce performance.

ACKNOWLEDGMENT

This work was supported in part by NSF grants ATD-1737996, REU-1343123, SCC-1737585, and IIS-1909916.

REFERENCES

[1] D. Liben-Nowell and J. Kleinberg, "The link prediction problem for social networks," in *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, ser. CIKM '03. ACM, 2003, pp. 556–559.

[2] D. S. Goldberg and F. P. Roth, "Assessing experimentally derived interactions in a small world," *Proceedings of the National Academy of Sciences*, vol. 100, no. 8, pp. 4372–4376, 2003.

[3] M. Hasan, V. Chaoji, S. Salem, and M. Zaki, "Link prediction using supervised learning," 01 2006.

[4] C. Lei and J. Ruan, "A novel link prediction algorithm for reconstructing proteinprotein interaction networks by topological similarity," *Bioinformatics*, vol. 29, no. 3, pp. 355–364, 12 2012.

[5] H. Chen, X. Li, and Z. Huang, "Link prediction approach to collaborative filtering," in *Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL '05)*, June 2005, pp. 141–142.

[6] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Neurocomputing: Foundations of research," J. A. Anderson and E. Rosenfeld, Eds. Cambridge, MA, USA: MIT Press, 1988, ch. Learning Representations by Back-propagating Errors, pp. 696–699.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, 1997.

[8] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[9] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: large-scale information network embedding," *CoRR*, vol. abs/1503.03578, 2015.

[10] A. Bojchevski and S. Gnnemann, "Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking," in *International Conference on Learning Representations*, 2018.

[11] M. Rahman and M. A. Hasan, "Link prediction in dynamic networks using graphlet," in *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part I*, 2016, pp. 394–409.

[12] L. Zhou, Y. Yang, X. Ren, F. Wu, and Y. Zhuang, "Dynamic Network Embedding by Modelling Triadic Closure Process," in *AAAI*, 2018.

[13] L. Zhu, G. V. Steeg, and A. Galstyan, "Scalable link prediction in dynamic networks via non-negative matrix factorization," *CoRR*, vol. abs/1411.3675, 2014.

[14] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, March 1994.

[15] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ser. ICML'13, 2013.

[16] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ser. ICML'14, 2014, pp. II–1764–II–1772.

[17] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *CoRR*, vol. abs/1409.3215, 2014.

[18] A. Graves, "Generating sequences with recurrent neural networks," *CoRR*, vol. abs/1308.0850, 2013.

[19] K. Gregor, I. Danihelka, A. Graves, and D. Wierstra, "DRAW: A recurrent neural network for image generation," *CoRR*, vol. abs/1502.04623, 2015.

[20] S. Abu-El-Haija, B. Perozzi, and R. Al-Rfou, "Learning edge representations via low-rank asymmetric projections," *CoRR*, vol. abs/1705.05615, 2017.

[21] P. Yanardag and S. Vishwanathan, "Deep graph kernels," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '15. ACM, 2015, pp. 1365–1374.

[22] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 2, no. 1, pp. 37 – 52, 1987, proceedings of the Multivariate Statistical Workshop for Geologists and Geochemists.

[23] J. B. Kruskal and M. Wish, *Multidimensional scaling*. Sage, 1978, vol. 11.

[24] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Advances in Neural Information Processing Systems 14*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds. MIT Press, 2002, pp. 585–591.

[25] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'13, 2013, pp. 3111–3119.

[26] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," *CoRR*, vol. abs/1403.6652, 2014.

[27] H. Cai, V. W. Zheng, and K. C. Chang, "A comprehensive survey of graph embedding: Problems, techniques and applications," *CoRR*, vol. abs/1709.07604, 2017.

[28] V. S. Dave, B. Zhang, P. Chen, and M. A. Hasan, "Neural-brane: Neural bayesian personalized ranking for attributed network embedding," *CoRR*, vol. abs/1804.08774, 2018.

[29] J. Li, H. Dani, X. Hu, J. Tang, Y. Chang, and H. Liu, "Attributed network embedding for learning in a dynamic environment," *CoRR*, vol. abs/1706.01860, 2017.

[30] M. Hasan, W. Spangler, T. Griffin, and A. Alba, "Coa: finding novel patents through text analysis." 01 2009, pp. 1175–1184.

[31] M. G'enois and A. Barrat, "Can co-location be used as a proxy for face-to-face contacts?" *EPJ Data Science*, vol. 7, no. 1, p. 11, May 2018.

[32] L. Isella, J. Stehl, A. Barrat, C. Cattuto, J. Pinton, and W. Van den Broeck, "What's in a crowd? analysis of face-to-face behavioral networks," *Journal of Theoretical Biology*, vol. 271, no. 1, pp. 166–180, 2011.