Cold Start Link Prediction Using Temporal Graph Embeddings

Deepika Rama Subramanian
dramasubramanian@vt.edu
Virginia Tech

Deepthi Peri
pdeepthi@vt.edu
Virginia Tech

Dhruva Sahasrabudhe
dhruvas@vt.edu
Virginia Tech

ABSTRACT

In this project, we generate node embeddings for directed temporal graphs. We propose two major modifications to the sampling strategy and transition probability computation of the popular *node2vec* framework, suited for directed temporal graphs. Our approach is inspired by the task of achieving robust performance for 'cold start link prediction'. We come up with a node embedding method called the "Friend of Friend" embedding. We test the performance of our modified embeddings against the original (unmodified) embeddings on two temporal graph datasets, and find that our method consistently gives good results for link prediction both in the case of cold start nodes, and general nodes, as compared to node2vec.

KEYWORDS

Node2Vec, Graph embeddings, Cold start,Deep learning, Link prediction, Recommendation systems, Random walks

1 INTRODUCTION

1.1 Embeddings for Directed Temporal Graphs

Currently, the Node2Vec embedding does not capture the temporality of the graph. Given that real-world graphs are constantly growing in size, capturing the temporal aspect is extremely important to understand the evolution and changing structures of these systems. In order to achieve this, we temporally bias the random walk within Node2Vec. These embeddings play an important role in link prediction. In particular, we are interested in addressing the cold start link prediction problem through our temporal graph embeddings.

1.2 Cold Start Problem in Link Prediction

Link prediction is used to predict the future possible links or the missing links in a network. That is, given the snapshot of a network, we try to predict the next most likely links to form in the network.[4] Several models have been proposed to solve it. We are looking at a specific case in the link prediction scenario: The cold start problem. The term cold start is derived from cars, referring to the problem encountered when trying to start a car engine in the cold. However, once the optimal temperature is reached, the car runs smoothly.

With recommendation systems, the term cold start means that the circumstances are not yet ideal for the system to provide the best possible results. In the case of link prediction in a graph network, lack of information about new nodes give rise to the cold start problem. There aren't sufficient links from the new node to another to carry out accurate link prediction. [5] Attempting to predict the links between new nodes and existing nodes will result in inaccurate to bad link predictions.

For example, consider a movie recommendation system. A new user, who is interested in Sci-fi movies is added to the system. However, this user interacts with a Romantic movie first. This means that the recommendation system will predict links to other romantic movies, which is obviously inaccurate considering that the user likes Sci-fi movies.

In traditional recommendation systems, the cold start problem is dealt with either explicitly by querying the users to obtain node information or implicitly by using machine learning models to infer the information based on external information, like the demographics, location, age, sex etc. of the user, or their interactions with another system.

1.3 Why Graph Embeddings

It is desirable to perform classification or prediction tasks on graphs, like labeling nodes/edges, predicting links, inferring edge direction, etc. Classification or prediction can be done in a supervised or unsupervised manner. Supervised machine learning requires optimization of a large number of parameters, and hand engineering of features, based on domain knowledge. Handcrafting structure based features, and obtaining labeled data is not as easy for data represented as graphs as it is in other domains.

Unsupervised machine learning techniques define an optimization function independent of the downstream task. However, techniques like PCA and their extensions involve transforming the data matrix using expensive eigen decomposition operations, and do not always perform well on downstream classification tasks.

That is why graph embeddings, which are vectorized, distributed representations of nodes/edges of a graph over a continuous, low dimensional space are useful for representing graph data. These embeddings can be trained according to a specific optimization function for a specific task, yielding better performance than generic dimensionality reduction techniques.

Moreover, they compress graphs of arbitrary size into a dataset of fixed size, making it possible for them to be used as input in downstream machine learning tasks.

For these reasons, graph embedding based approaches have become very popular recently, and various graph embeddings have been developed and adopted widely in a variety of domains.

2 PROBLEM DEFINITION

Graph embeddings are the current state-of-the-art for tasks like link prediction, and are growing in popularity. Thus, it is important to make these embeddings more robust for special scenarios where the embedding might fail to accurately represent the node.

One such scenario is for nodes representing cold-starts, in temporally varying networks. That is, for a new node in the graph, which has not existed in the network for a long time, and has very few links to other nodes. The true neighborhood structure for this type of node might be very different in the future, when the node is "settled in". Here, the embedding, which uses only the current network connectivity, may not work well.

In this project, we propose a method to overcome the problem of cold starts in link prediction for graphs. We do so by creating a new temporal embedding over the nodes of the directed temporal graph, purely by using information about the current and past structure of the graph, without using external domain-specific information.

We consider the history of other nodes which were cold start nodes, and in a "similar situation" to the node in consideration. We use this history, along with the assumption that the cold start node for which we need to perform link prediction will want to form links in a manner similar to the way similar nodes did in the past. We do this for arguably the most popular graph embedding technique, node2vec [3].

Moreover, we also make use of the temporal nature of the graph by biasing our random walks to pass through links which are formed recently in the history of the node.

We modify the random walk sampling procedure defined in node2vec, to encode our modifications, and compare our approach to the default embedding method, specifically for the task of link-prediction in both cold-start and non cold-start nodes.

3 RELATED WORK AND SURVEY

In this section, we provide a succinct version of work that we surveyed. Liben-Nowell et. al.[7] formalize the link prediction problem and develop approaches based on proximity of nodes in the network. Taskar et. al. [11] apply link prediction to social networks in universities. They use machine learning techniques as well as querying to obtain user information to increase link prediction accuracy. Glenski et. al. [2] predict user interaction in Reddit networks by recording clicking, browsing and voting behaviour. Leroy et. al.[5] propose a two phase method based on bootstrap probabilistic graphs, wherein the first phase they generate an implicit network in the form of a probabilistic graph and in the second phase they apply probabilistic graph based methods to obtain the final link prediction. Wang et. al. [12] suggest a type of Graph Embedding for predicting sign of sentiment links in social network graphs. This is effective in cold start predictions as well. The Signed Heterogeneous Information Network Embedding (SHINE) framework to extract user's latent representations from heterogeneous networks and predict the sign of the unobserved links. SHINE adopts multiple deep autoencoders, a type of deep learning based embedding technique, to extract user's highly nonlinear representation from the sentiment network, social network and profile network, respectively. The learned three types of user representations are subsequently fused together by specific aggregation function for further sentiment prediction. Cai et al [1] provide a comprehensive analysis of Graph embedding techniques. They also provide a taxonomy of Graph embeddings based on various problem settings and systematically categorize the applications that graph embeddings enable. Singer et al. [10] propose an algorithm to temporal node embedding of each node using historical node embeddings, tuning it to learn the evolution of each node and edge in the graph over time. They incorporate this algorithm into a traditional node embedding framework to carry out two tasks, namely Link Prediction and Multi-Label Classification. They come up with a joint loss function that creates temporal embedding of a node by learning to combine the historical temporal embeddings. Their method, called the tNodeEmbed, mimics sentence embedding for feature learning in temporal graphs. Each node v, is associated with a matrix $X(v)$ $\subseteq R_T$ $..d$ of its historical T embeddings over time, each of size d. The graph G is associated with |V | matrices, one for each node: $G_X = X(v_1), ..., X(v|V|)$. Given G_X ,they perform graph prediction tasks. Rossi et al [9] propose a framework to learn time-respecting embeddings from continuous-time dynamic networks, i.e temporal networks. They address the problem of learning an appropriate network representation from continuous-time dynamic networks for improving the accuracy of predictive models. They describe a general framework for incorporating temporal dependencies into network embedding methods, resulting in a more appropriate time-dependent network representation that captures the important temporal properties of the continuous-time dynamic network.

4 PROPOSED METHOD

4.1 Node2vec

Node2vec [3] is an approach proposed by Grover et. al. in 2016, to create node embeddings in a directed/undirected, weighted/unweighted graph. They modify the approach used by Mikolov et. al. to generate word embeddings in a landmark paper, where they introduced the *word2vec* technique [8]. Mikolov et. al. developed a skip-gram model to generate the word embeddings, based on the principle, "a word is defined by the company it keeps". That is, the embeddings for words which often occur in similar contexts will be nearby in the vector space. The context of a word for a particular occurrence is defined by a bag-of-words model taking into account a fixed size window around the word.

Node2vec modifies this for graphs, by treating a node in a graph as a "word", and by treating a random walk in the graph as the context of this "word". The embeddings learned in this way optimize for preserving some combination of the local and global network structure in the graph for each node, causing nodes which occur in similar contexts to cluster together in the vector space just like similar words do in the case of word2vec.

The major contribution of the paper is a clever sampling strategy over the nodes, so as to generate robust embeddings which capture both the local and global network structure in the graph. The sampling strategy combines a BFS (Breadth First Search) and DFS (Depth First Search) to achieve this. This is done using two parameters, the **return parameter** p and the **inout parameter** q. p controls the probability of returning to the previous node in the walk, while q controls the probability of exploring further nodes. Other parameters include the walk length, and the number of walks to be generated for each node.

4.2 Intuitions and Methods

Intuition #1: We define *friends of friends* in the following way: Given a node u in a directed temporal graph $G = (V, E)$, the set of friends of friends of the node u:

$fof(u) = \{u'|\forall u'\forall v : (u, v) \in E \land (u', v) \in E\}$.

Our intuition is that in cases where a node has not formed too many links, a random walk from that node will not be able to appropriately capture the neigborhood of that node. In this case, the friends of friends of the node can provide information about the links that the node may form in the future.

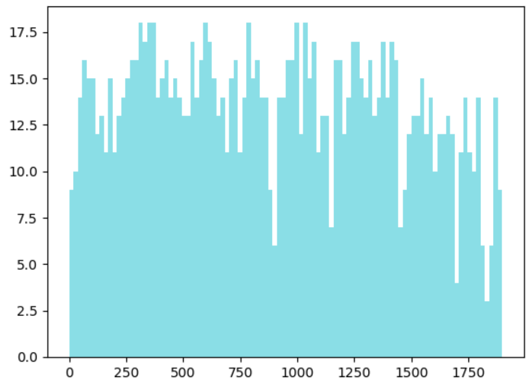

Figure 1: Distribution of time for which users have been on the platform for CollegeMSG dataset

We feel that a person may mirror other people who joined the circle by connecting to the same friend. In other words, we think that a node (X) that cold starts with a particular node (Y) in the graph is likely to behave similar to node (Z) that also cold started with (Y).

We substantiated this intuition by doing a simple prediction task using our 'Friend of Friend' method. We used Jaccard's similarity between our prediction and the actual connections made by cold start nodes to validate its efficacy over a random case. We performed this twice:
1. By using the most recent node that cold started with a particular node X (Last FoF). The comparison of this case vs random is presented in the Table 1
2. By obtaining a random subset of nodes that cold started with a particular node X (FoF Random). The comparison of this case vs random is presented in the Table 2.
Both of these cases performed substantially better than a complete random selection. Note that we used the sum of the Jaccard's similarity over all the cold start nodes.

For our embeddings, we pick the friend of friend nodes with a probability proportional to their degree.
Intuition #2: Our next intuition was that temporal connections tend to happen in spurts and not uniformly. For example, on the Twitter network, we follow several users in a short interval of time and then not at all until another 'follow' spurt comes along. We confirmed our intuition by performing 1-D K Means Clustering on our datasets' timestamps of link-formation. We found that the results of the clustering showed a small number of clusters and connections huddled together. Some samples of our clustering exercise are shown in Figure 2.

Moreover, it seems natural that the most recently formed neighbors would be the most important in giving information about the node, and the further back in past the formed links are, the less important they should be.

DataSet	Random	FoF Random
CollegeMsg	5463.0807	6993.9447
Reddit Hyperlinks	285520.9318	876212.9821

Table 1: FoF Random vs Random results

DataSet	Random	Last FoF
CollegeMsg	1.6316	6.9939
Reddit Hyperlinks	25.9117	162.3975

Table 2: Last FoF vs Random results

In our embeddings, we biased the random walk in favour of the edges that were created in the most recent spurt.

Intuition #3: Our third and final intuition is that once a node has created enough links, the *friend of friends* approach is less desirable. Here, the friend of friend nodes would now contribute a lot of noise thereby making this approach unreliable. Moreover, since the nodes "place" within the network structure gets more set as it forms more links, it becomes less necessary to depend on the friend of friend nodes of the node, since a temporally biased node2vec style walk from the node itself will be better.

In order to tackle this issue while generating our embeddings, we perform multiple random walks and probabilistically choose either the friend of friend method or the temporally biased random walk that begins at the node itself. This probability depends on the number of connections the node has formed.

The mathematical formulations of our intuitions are given in the next section.

4.3 Algorithm

Let $G = (V, E, T)$ where V is the set of nodes, $E \subseteq V \times V$ is the set of edges, and $T : E \longrightarrow \mathbb{R}$ is a function over the edges giving the time of edge formation in the graph.

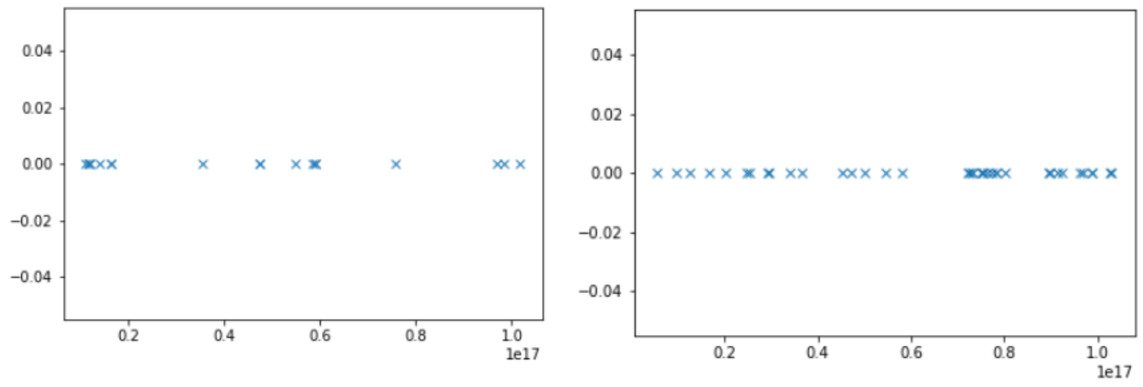

Figure 2: Spurts of link-formation

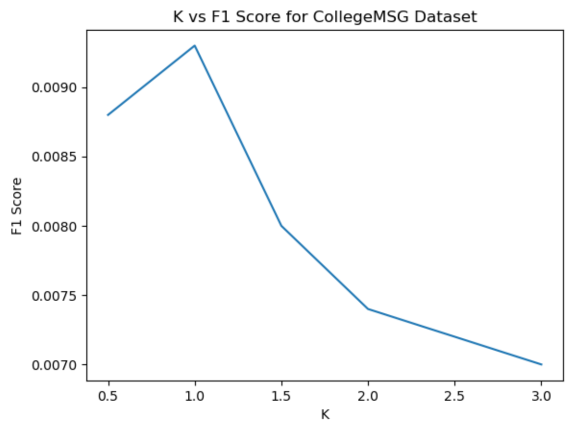

Figure 3: K vs F1 Score for CollegeMSG Dataset

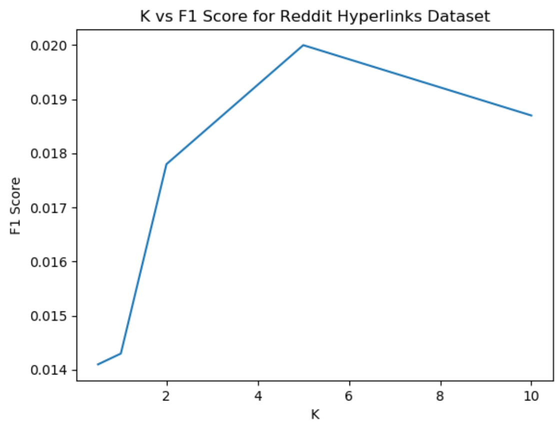

Figure 4: K vs F1 Score for Reddit Hyperlinks Dataset

For each node u, we find the timestamp of the last link it makes, denoted by ts_{final}. Now, for each edge $(u, v) \in E$, let the timestamp of the edge be ts. The weight of edge (u, v) is given by:

$w(u, v) = e^{-k \cdot \frac{|ts - ts_{final}|}{ts_{final}}}$, where k is a parameter, dependent on the dataset.

Now, we define two sampling strategies to generate the embedding for u, $fof\_walk(u)$ and $temporal\_walk(u)$.

- $temporal\_walk(u)$ performs a node2vec style directed random walk over the graph, starting at u, where at each step, if the walk is on node n, the transition probability to each node n' is given by:
  $P(n') = \frac{w(n,n')}{\sum_{(n,m)\in E} w(n,m)}$
  The other node2vec specific parameters of the walk are either the default values, or tuned for our algorithm. This is discussed in the Results section.
- $fof\_walk(u)$ performs a node2vec style directed random walk over the graph, starting at a random node n, where $n \in fof(u)$, where the probability of selecting n:
  $P(n) = \frac{out\_degree(n)}{\sum_{n' \in fof(u)} out\_degree(n')}$
  where $out\_degree(n)$ is the number of outgoing edges from n.

Sampling Strategy The sampling strategy for node n is given as:
$S(n) = fof\_walk(n)$ with probability α, and
$S(n) = temporal\_walk(n)$ with probability $1 - \alpha$.

Where $\alpha = \frac{1}{1+out\_degree(n)}$

4.4 Datasets

Our datasets are going to be sourced from SNAP: Stanford Large Network Dataset Collection[6]. As node2vec only addresses unsigned and homogeneous networks, we choose the Social Network: Reddit Hyperlink Network and the CollegeMsg temporal network datasets. These are both temporal datasets which offers the time stamps we require for cold-start link predictions.

CollegeMsg Temporal Network: This dataset consists of private messages sent between on an online social network students at UC Irvine. An edge (u, v, t) represents a message sent from user u to user v at time t. It consists of 1899 nodes, 59835 temporal edges, 20296 edges in the static graph collected over 193 days.

Social Network: Reddit Hyperlink Network: This dataset consists of subreddits that are hyperlinked to each other through their posts. This dataset contains both hyperlinks present in the body and the title of posts. For the purpose of this project, we will only be considering the hyperlinks in the body. An edge (u, v, t) represents a subreddit u whose post hyperlinks to another subreddit v at time t. It consists of 55863 nodes, 858490 temporal edges, 20296 edges in the static graph collected over 3.5 days.

5 EXPERIMENTS

5.1 Node2Vec Hyper-Parameter Tuning

The random walks generated by Node2Vec are highly dependent on the user-defined hyper-parameters namely the return parameter p, the in-out parameter q, the number of walks num_walks and, the random walk length $walk\_length$. After several runs with varied parameters, we observed the best results with $walk\_length$=8, num_walks = 10 and default values for p and q.

5.2 Temporal Bias Factor Tuning

We incorporate the temporal bias factor K to determine probabilistically whether to choose the friend of friend method or to use the temporal window. We tried several values for K and found that when K = 5, our Mean Average Precision (MAP) for prediction is at its highest for the Reddit dataset, and the value of K = 1 is highest for the College MSG dataset. Figures 3 and 4 present the change of the F1-Score with respect to change of our temporal bias factor, K for the CollegeMSG and Reddit Hyperlinks datasets respectively.

6 TESTS AND RESULTS

We predict links in the graph by simply measuring the Euclidean distance between the embeddings of every pair of nodes in the embedding space, and returning a list of the top results, ranked.

We run our tests in two phases:

- First, we predict links for cold-start nodes. Here,
  - We use the graph in a static fashion.
  - We partition our dataset into two parts, the first 75% is the training set and the rest is the test set.
  - We determine all the cold-start nodes in our training set, i.e. the nodes which have made only one link and attempt to predict the links that are to appear in the test set.
  - We run this for 50 iterations on both datasets using Node2Vec and FriendOfFriend method.
- Second, we attempt to only predict the top link for all nodes in the graph.
  - For this case, we try 4 different splits of the graph into train and test, 20/80,40/60,60/40,80/20.
  - After this, we attempt to predict the links that appear in the test set. We perform this for all nodes, not only cold-starts.
  - We run this for 50 iterations on both datasets using Node2Vec and FriendOfFriend method and then compare our results.

We are using two ways to measure success for cold start link prediction: Average over the iterations for **(i) Number of successful link Predictions and (ii) Mean Average Precision (MAP)** We present our findings for the cold-start link prediction (1) in table 3 and 4.

For cold starts, we used MAP@10, and the top 10 links as the ranked list for MAP@10, while for all nodes, we used precision, recall, and f-1 score.

We also have the values of precision, recall and f1 score for link prediction (2) in Table 5 and 6 for both the datasets.

From our results, it is apparent that our embeddings are performing consistently better than the Node2Vec embeddings for cold-start and normal link predictions.

From the above results, we can infer that when the dataset is small and there is not enough temporal information, Node2vec outperforms our model. For example, in the Reddit Hyperlinks dataset, when the train-test split is 20-80

Number of Succesful Predictions	FoF	Node2Vec
Reddit Hyperlink Dataset	**309**	283
CollegeMSG Dataset	**17**	14

Table 3: Number of Succesful Predictions for FoF vs Node2Vec for Cold-Start Nodes with 75-25 train-test split

Average MAP at 10	FoF	Node2Vec
Reddit Hyperlink Dataset	0.00149	**0.00155**
CollegeMSG Dataset	**0.00499**	0.000328

Table 4: Average MAP at 10 for FoF vs Node2Vec for Cold-Start Nodes with 75-25 train-test split

Dataset	Train/Test Split	Precision	Recall	F1 Score
CollegeMSG	20/80	0.08080	0.00379	0.00725
CollegeMSG	40/60	0.05465	0.00470	0.00865
CollegeMSG	60/40	0.05360	0.00837	**0.01448**
CollegeMSG	80/20	0.02981	0.01148	**0.01658**
Reddit Hyperlinks	20/80	0.063108	0.00646	**0.01172**
Reddit Hyperlinks	40/60	0.04640	0.00964	**0.01597**
Reddit Hyperlinks	60/40	0.03503	0.01361	**0.01960**
Reddit Hyperlinks	80/20	0.02185	0.01911	**0.02039**

Table 5: Precision, Recall, and F-1 score FoF embeddings for different partitions of the graph into train, test

Dataset	Train/Test Split	Precision	Recall	F1 Score
CollegeMSG	20/80	0.083333	0.003918	**0.00748**
CollegeMSG	40/60	0.06003	0.00516	**0.00950**
CollegeMSG	60/40	0.03431	0.00535	0.00926
CollegeMSG	80/20	0.02683	0.01033	0.01492
Reddit Hyperlinks	20/80	0.04769	0.00488	0.00886
Reddit Hyperlinks	40/60	0.031150	0.00647	0.01072
Reddit Hyperlinks	60/40	0.02452	0.00953	0.01372
Reddit Hyperlinks	80/20	0.01516	0.01326	0.01415

Table 6: Precision, Recall, and F-1 score node2vec for different partitions of the graph into train, test

7 DRAWBACKS

- Computing and Modifying the random walk for each node in the graph adds to the time complexity of the algorithm. Additionally, implementing the friend of friend method for all the nodes adds to the time complexity as well.
- We have tuned the parameters for the model manually and have not accounted for every possibility.
- Keeping in mind the scope of the project, we have considered only two datasets for the time being.Since the Temporal bias factor k is heavily dataset dependent, testing our algorithm on more datasets would be beneficial.

8 CONCLUSION AND FUTURE WORK

We have shown that our technique is better than vanilla node2vec both for cold start link prediction and for link prediction in general nodes. We have also shown that our intuition has both statistical and empirical validity on the task of link prediction, as our methods outperform node2vec.

For future work, we can add parallelism to the running of the code, to make our algorithm faster, and we can try the friend of friend modification for other graph embedding techniques like convolutional graph embeddings, or other temporal graph embeddings.

We can also evaluate the running of our method on new datasets, and find optimal hyperparameter values.

REFERENCES

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

[2] M. Glenski and T. Weninger. Predicting user-interactions on reddit. *CoRR*, abs/1707.00195, 2017.

[3] 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*, pages 855–864. ACM, 2016.

[4] M. A. Hasan and M. J. Zaki. *A Survey of Link Prediction in Social Networks*, pages 243–275. Springer US, Boston, MA, 2011.

[5] V. Leroy, B. B. Cambazoglu, and F. Bonchi. Cold Start Link Prediction. In *The 16th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, page 12 p, Washington DC, United States, July 2010.

[6] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

[7] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *J. Am. Soc. Inf. Sci. Technol.*, 58(7):1019–1031, May 2007.

[8] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[9] G. H. Nguyen, J. B. Lee, R. A. Rossi, N. K. Ahmed, E. Koh, and S. Kim. Continuous-time dynamic network embeddings. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, pages 969–976, Republic and Canton of Geneva, Switzerland, 2018. International World Wide Web Conferences Steering Committee.

[10] U. Singer, I. Guy, and K. Radinsky. Node embedding over temporal graphs. *CoRR*, abs/1903.08889, 2019.

[11] B. Taskar, M.-F. Wong, P. Abbeel, and D. Koller. Link prediction in relational data. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, NIPS'03, pages 659–666, Cambridge, MA, USA, 2003. MIT Press.

[12] H. Wang, F. Zhang, M. Hou, X. Xie, M. Guo, and Q. Liu. Shine: Signed heterogeneous information network embedding for sentiment link prediction. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 592–600. ACM, 2018.