Diffusion based temporal network embedding for link prediction

Ziyu Li

TU Delft

Diffusion based temporal network embedding for link prediction

by

Ziyu Li

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday August 30, 2019 at 10:00 AM.

Student number:	4710126	
Project duration:	November 1st, 2018 – August 30th, 2019	
Thesis committee:	dr. P. Cesar,	TU Delft
	Prof. dr. A. Bozzon,	TU Delft
	dr. H. Wang,	TU Delft, Daily supervisor
	X. Zhan,	TU Delft, Daily supervisor

This thesis is confidential and cannot be made public until December 31, 2019.

TU Delft

Contents

Preface

Three years ago, I visited Netherlands while I was an exchange student in Sweden. I was drawn by the unique street view, bridges connecting canals with pedestrians and cyclists passing through it. I posted on my timeline that this is a country of freedom. One year later by then, I stepped on this land again. While this time, I am about to be a master student. Though one year has passed, the train station and the canals brought me back to the day I visited last time, as if it was yesterday. What was different is that I came to spend the next two years for further study. I was determined and dedicated to extend my professional path. I hoped that this free country can provide me with a free academic environment. As it turns out, studying in TU Delft is a treasured gift in my life, paving the way for my professional passion and bringing cherish to my life. I have great thanks to a lot of people.

Dr. Huijuan Wang, my supervisor, guides me through the way of the thesis project. Thank you for your professional support, which has all the time inspired me and provided with new inspirations while I was in a dead end. Your patience while listening to me reporting and answering my questions has encouraged me to think further on my work. I thank you sincerely for showing me the mindset to be a responsible researcher with rigorous scholarly working attitude.

Xiuxiu, thank you for being a supportive supervisor and a true friend. It wouldn't be that smooth for me to get on track without your progressive and critical guidance. Whenever I felt lack of confidence about my work, you are the one who dragged me from the gloom and encouraged me, for which I am sincerely grateful. We had so much fun together while dancing Zumba and playing badminton with Manel and Malek. You are never too shy to introduce me to your friends. I had made so many friends in the last year of my master, from whom I learned a lot.

I would like to thank Pablo Cesar and Alessandro Bozzon being my thesis committee members. Also, I would like to thank all the members in MMC group. Manel, Roger, Omar, Alberto, Julian, Cynthia, Odette, Alan, Elvin, Jay, Sandy, Harlley, Andrew, thank you for making my 'research year' so much fun. I am grateful for the snacks and drinks, also birthday cakes, from time to time. :)

Wanning, Jiahui, Zequn, Felix, Kun, thank you all being the best buddies during my master study. I become a better self because of you. We shared happiness in good times, and you were always there when I was down. Your company made my master a unforgettable experience! Chitra, thank you for being a 'research partner' during the second year. We have had so many common courses and now we are doing thesis under the same supervisor. Wish you all the best.

The last two years has filled with ups and downs. I am grateful for the moments no matter they are bitter or sweet, since those experiences have made me who I am today. Finishing my master study is not the destination. Tomorrow is another day.

Ziyu Li
Delft, August 2019

List of Figures

List of Tables

List of Symbols

T	Temporal window size, $T \in R^+$
$\mathscr{G}$	Static network
$\mathscr{G}_{\mathscr{W}}$	Weighted network
$\mathscr{G}_T$	Temporal network
G_S	Integrated network formed by trajectories extracted from SI(S) spreading process.
$l_{i,j,t}$	Contact between node i and node j at time t
$\mathscr{N}$	Node set
$\mathscr{L}_T$	Contact Set
N	Number of nodes
$\lvert\mathscr{L}_T\rvert$	Number of contacts
$\mathscr{B}$	Context size, number of nodes in walk set corpus
ws	Skip-Gram model window size
d	Dimension of embedding vectors ($d \ll N$).
β	Infection rate
γ	Recovery rate
$\mathscr{T}_i(\beta,\gamma)$	The spreading tree records union of contacts starting from node u_i under the combination of parameters, β and γ.
$\overrightarrow{h_i}$	Embedding vector of node u_i.
f	Dot product of the embedding vector between two nodes.
$N_S(u_i)$	The neighboring set of node u_i sampled from the SIS spreading process.
CC	Clustering coefficient
$<d>$	Mean degree
$<wl>$	Mean trajectory length
τ_w	Edge weight correlation
H	$H \in R^{N\times d}$ is the node embedding matrix

Abstract

Link prediction in complex networks has attracted increasing attention. The link prediction algorithms can be used to retrieve missing information, identify spurious interactions, capturing network evolution, and so on. Recently, network embedding has been proposed as a new strategy to embed network into low-dimensional vector space. By embedding nodes into vectors, the link prediction problem can be converted into a similarity comparison task. Nodes with similar vectors are more likely to connect. Some traditional network embedding methods include matrix factorization, random walk paradigm and deep neural network models.

In this thesis, we propose *SISNE*, a diffusion based paradigm for node embedding, applying Susceptible-Infected-Susceptible (SIS) model to extract node neighborhood structure. Both random walk based algorithms and our proposed method sample node sequences as input and feed them into a Skip-gram model, a representative language model that embeds words into vectors. Specially, our proposed model provides flexibility to explore the network topology by operating information spreading on networks. Another contribution of the proposed model is that *SISNE* takes into the account of the evolving nature of complex networks. To verify the efficacy, we conduct experiments on missing link prediction task and show that our SIS diffusion based model outperforms other state-of-the-art network embedding algorithms across all four empirical datasets, reaching a maximal 7% improvement. Importantly, even when the input size is small, the performance remains stable whereas other baseline models drop dramatically, which indicates that our proposed model is less sensitive to input size and suggests that the model is applicable to large-scale networks. Moreover, we further show that as long as the infection probability β is larger than the threshold value of the diffusion model, we can obtain a relatively high performance for link prediction task. Taken together, our work has shown great effectiveness and efficiency in learning embeddings in temporal networks.

1 Introduction

Networks are becoming ubiquitous across a large spectrum of fields. The critical challenge is to represent networks effectively and efficiently so as to tackle advanced analytic tasks such as pattern recognition[57, 67], community detection[81], prediction[37, 55] and visualization[78]. Recently, network embedding has been proposed as a new strategy to embed network into low-dimensional vector space, in which way, network structure is preserved[16, 45, 70]. In this thesis, I strive to investigate further to predict links in a network.

1.1. Motivation

Most of the real-world complex systems can be represented as complex networks, with nodes representing the components and links representing the connections between these individuals [50, 90]. Therefore, the study of complex networks pervades in different fields[15]. For example, with biological or chemical networks, scientists study interactions between proteins or chemicals to facilitate new treatments or components[20, 58]. With social networks, researchers tend to classify or cluster users into groups or communities, which is useful for many tasks, such as advertising, search and recommendation[28, 74]. With communication networks, learning the network structure can help understand how the information spreads over the networks[90]. These are only a few examples of the important role of analyzing networks.

Link prediction, one of the most fundamental problems among all the network analysis tasks, has multiple applications, such as social recommendation on social platforms (e.g., Facebook or Twitter) or recommendation applications on e-commerce systems [39, 40, 45]. Generally, the link prediction problem can be classified into two classes, i.e., predicting the missing links by using the observed networks or predicting future links by using the historical network structure [37]. Link prediction has been attracting considerable amount of attention and has been successfully applied in multiple fields. Email or communication networks, usually regarded as highly dynamic systems, are served as popular observation targets for link prediction[46]. In addition, social media have been working on the task to suggest unknown friends who may have strong connection with us[74, 80]. E-commerce websites recommend products that we could be interested in[30]. The prediction of unknown interactions between proteins in biological networks is also in line with the link prediction research [44, 58].

The aim of link prediction problem is to estimate the likelihood that two nodes connecting to each other based on observed network structure[19]. To solve the advanced analytic tasks, e.g., link prediction, node classification and etc, a concise learning of a network is fundamental. A typical solution is to extract hand-engineered features from network based on expert knowledge[16]. Despite the tedious effort for feature engineering, the lack of applicability across different prediction tasks

is also a concern[23]. Alternatively, network embedding, as a promising way to represent networks, has attracted great interest recently[16, 81]. Network embedding, as the name suggested, embeds nodes in a network into a low-dimensional vector space . Traditionally, a discrete adjacency matrix is used to represent a network[88]. The elements of the matrix indicate the existence of edges in the network: pairs of nodes connected are presented as positive values (1 in an unweighted network or the weight of the link in a weighted network) in the corresponding positions (row or column index indicates an unique node). Simply, a row vector or a column vector in the adjacency matrix can be regarded as the representation for a specific node. In such a way, the representation space is N-dimensional, where N is the number of nodes in the network. However, this kind of representation contains a lot of zero values and the dimension in some large-scale networks can reach up to millions, which could cause high computation complexity. Thus works in the early 2000s, attempted to achieve lower-dimensional embedding using dimension reduction techniques. Representative works, such as Isomap [73], Locally Linear Embedding(LLE) [65] and Laplacian Eigenmap [4], are conducted to embed graph into lower dimensions. Nonetheless, time complexity usually reaches quadratic level with respect to node number. Extensive researches apply matrix factorization methods to learn a low-rank space for the adjacency matrices. Among a series of matrix factorization models, Singular Value Decomposition (SVD) [21] is commonly used. However, network embedding from adjacency matrix can only reveal a simple neighboring relationship between nodes. Complex relationships such as high-order proximity are neglected. Let alone the adjacency matrix contains noise or abundant information. In addition, these traditional methods have many limitations when processing on large-scale networks.

In the field of Natural Language Processing (NLP), similar issues also occur when learning representation of words, embedding words into lower-dimension. The development of language models significantly improve the effectiveness of word embedding. Word2Vec, an up-to-date word embedding model, has paved the way for learning dense, continuous and low-dimensional representations for words [48]. Node embedding methods using random walk paradigms borrowed the idea from the language model and established an analogy for networks by regarding a network as a "document" and a node as a "word"[57, 70]. The same way as a document is an ordered sequence of words, one could sample sequences of nodes from the underlying network to capture the neighborhood of nodes. Models, *e.g.*, DeepWalk[57], Node2Vec[23] are representative works to learn low-dimensional vectors based on Word2Vec model, which will be further illustrated in the future section.

Even though a lot of works have been introduced about network embedding, they are mainly designed for static networks. Few work has considered the evolving nature of the network [52]. In addition, current network sampling methods are mainly based on random walk, which is a specific type of diffusion processes on the network. In this work, we propose to apply information spreading, e.g., SIS model, on networks to simulate node sequences, defined as *trajectories*. This brings us to the first research question: **Q1 – How to utilize the diffusion based model and temporal information to simulate trajectories on networks for node embedding?** Since we are solving a link prediction task, we aim to predict missing links on temporal networks. We concern about the efficacy, which leads to **Q2 – How is the performance of the proposed model in contrast with the other node embedding algorithms on the link prediction task in temporal networks?** For a more fine-grained research, we would also like to know the 'why': **Q3 – How to explain the difference in performance of different algorithms and how is the properties of the sampled information related to the performance?** In the following chapters, we will answers the questions in details.

1.2. Research Questions

Recently, there has been renewed interest in network embedding. Many attempts have been made to solve the link prediction task, including the random walk paradigm. This research strives to solve the task by proposing a diffusion based paradigm. We aim to explore which sampling strategy can

extract more informative corpus and achieve better performance. This give rise to our main research questions:

1. **How to utilize the diffusion based model and temporal information to simulate trajectories on networks for node embedding?**
2. **How is the performance of the proposed model in contrast with the other node embedding algorithms on the link prediction task in temporal networks?**
3. **How to explain the difference in performance of different algorithms and how is the properties of the sampled information related to the performance?**

1.3. Contribution

The contribution of this project is threefold:

1. We propose *SISNE*, which is an efficient scalable network embedding methods on temporal networks.
2. We evaluate *SISNE* on link prediction task. We compare our algorithms with the state-of-art static network embedding methods and show that considering temporal information of the network can help significantly improve the accuracy of link prediction. In addition, we compare *SISNE* with a random-walk based temporal network embedding method, i.e.,*CTDNE*, and find that *SISNE* is able to obtain much higher accuracy for link prediction even with a small amount of input data.
3. We explore the relationship of the extracted 'corpus' and the performance of the embedding method.

1.4. Thesis Outline

The rest of the report is structured as follows. In Chapter **2** Background, definitions and background knowledge pertaining the concepts used in this thesis will be explained. Baseline models, including their main concepts and privileges are introduced in the following chapter, Chapter **3** Baseline Models. In Chapter **4** Methododology, we present technical details for the proposed model, *SISNE*. Following in Chapter **5** Experimental Setup, we introduce the networks applied and data processing. In Chapter **6** Results and Analysis, *SISNE* is evaluated on link prediction task on various real-world datasets compared with baselines. Parameter sensitivity is also involved. In the last chapter Conclusion, we conclude with a discussion of the *SISNE* framework and propose some directions for future work.

2

Background

This chapter includes background information about network embedding as well as its categories. The current development on network embedding methods will also be introduced. These will lay the foundation for the following chapters.

2.1. Definitions

This section includes definitions and meaning of a network and a temporal network. The main notations that will be used in this thesis are given in Table 2.1.

2.1.1. Network

A network, represented as $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, is defined as a set of $\mathcal{N}$ nodes connected by a set of $\mathcal{L}$ links[68, 88]. The number of nodes is denoted by N, in which $N = |\mathcal{N}|$. $\mathcal{L} = \{l_{i,j}, i, j \in \mathcal{N}\}$ is the link set, where the element $l_{i,j}$ indicates a static link between node i and node j. An weighted network $\mathcal{G}_W = (\mathcal{N}, \mathcal{L}_W)$ is consisted with links assigned with weights. Each link $l_{i,j}$ in $\mathcal{L}_W$ is associated with a weight w_{ij}, the strength of the tie between node i and node j.

2.1.2. Temporal Network

A temporal network can be denoted as $\mathcal{G}_T = (\mathcal{N}, \mathcal{L}_T)$, where $\mathcal{N}$ denotes a set of nodes and N also represents number of nodes. Here, $\mathcal{L}_T = \{l_{i,j,t}, t \in T\}$, where T is a temporal window size with $T \in R^+$. $l_{i,j,t}$ indicates that node i and node j have a contact at time step t. A weighted network can be derived from temporal network $\mathcal{G}_T$ by aggregating all contacts over time, in which the weight of each link represents the occurrence of contacts between the two end nodes of the link[87].

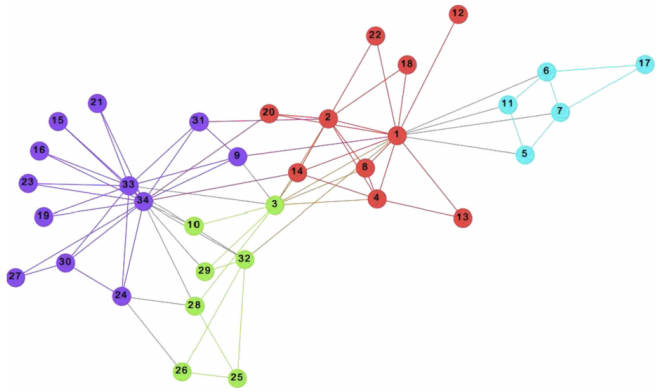
2.2. Network Embedding

The conventional approach to represent nodes and edges involves hand-crafted features, e.g., betweenness centrality, degree centrality, clustering coefficient and etc. Properties of network structure are extracted by computing betweenness centrality, triangle count, and modularity[59]. These features require extensive domain knowledge and expensive computation to obtain. To tackle this challenge, learning latent representations for networks, a.k.a., network embedding, has been extensively studied to automatically project the structural properties of a network into a latent space[16, 59, 88]. This background section focuses on the definition of network embedding and its motivation.

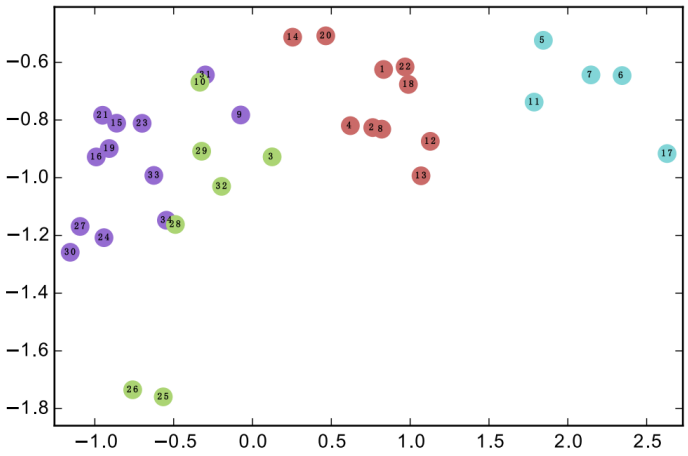
Network embedding, or Network Representation Learning(NRL), strives to embed the network into a low-dimensional space by preserving the network structure. When nodes or edges are embedded as points into low-dimensional space, the relationships among nodes and edges would be

Table (2.1) Notations

T	Temporal window size, $T \in R^+$
$\mathcal{G}$	Static network
$\mathcal{G}_T$	Temporal network
$l_{i,j,t}$	Contact between node i and node j at time t
$\mathcal{N}$	Node set
$\mathcal{L}_T$	Contact Set
N	Number of nodes
$\lvert\mathcal{L}_T\rvert$	Number of contacts
ws	Skip-Gram model window size
d	Dimension of embedding vectors ($d \ll N$).
H	$H \in R^{N \times d}$ is the node embedding matrix



(a) Input: karate network



(b) Output: representations

Figure (2.1) Network embedding example [57]

represented by distances between points in the space. The goal is to minimize the distance between similar nodes, either close in the network topology or share similar roles [57, 89]. Figure 2.1 shows an example of network embedding. In Figure 2.1(a), the karate network with colors representing different communities is shown. The karate network is embedded into a two-dimensional space in Figure 2.1(b). It can be observed that nodes with same color (i.e., within the same community)

in the original karate network have shorter distance in the two-dimensional space. After obtaining the embedding vector of each node, we can further use them as input for tasks like link prediction, community detection, classification, etc.

2.3. Network Embedding Methods

Generally, network embedding methods include matrix factorization, random walk based method and deep neural networks[16]. In this section, we will give an overview study of these three categories. We will also introduce graph neural networks, which has close relation to node embedding.

2.3.1. Matrix Factorization

Unsupervised feature learning approaches typically exploit various matrix representations of networks. Laplacian and the adjacency matrices are commonly used to represent the topology structure of a network. A row or column vector can simply represent a node, but the vector is formed in N-dimensional representation space. The goal of network embedding is to learn a low-dimensional vector space for a network. Thus dimensionality reduction techniques can naturally be applied to solve the problem. SVD is one commonly-used matrix factorization model, due to its ability for low-rank approximation[43, 55]. The idea of SVD is to decompose a matrix into the product of $U\Sigma V^T$, as shown in Figure 2.2. In Figure 2.2, the complex matrix M can be factorized as the production of orthonormal matrices U, V^T and a diagonal matrix Σ. *GraRep*[7], as one of the representatives model, learns low-dimensional representations of nodes by performing SVD to reformulate the loss function. The privilege of *GraRep* is its ability to preserve higher-order proximities (k-step proximities $k \leq 2$) when constructing the global representations of nodes. Given A is the adjacency matrix, the k-step probability transition matrix can be computed by $A^k = \prod^k A$, where A^k_{ij} refers to the transition probability between node i and node j consisting of k steps[16]. They extracted a positive k-step log probabilistic matrix X^k:

$$X^k_{i,j} = max\left(\log\left(\frac{A^k_{i,j}}{\Gamma^k_j}\right) - \log(\beta), 0\right) \tag{2.1}$$

,where $\Gamma^k_j = \sum_p A^k_{p,j}$ and β is a log shifted factor. The representations of nodes can be inferred by applying SVD on $X^k_{i,j}$.

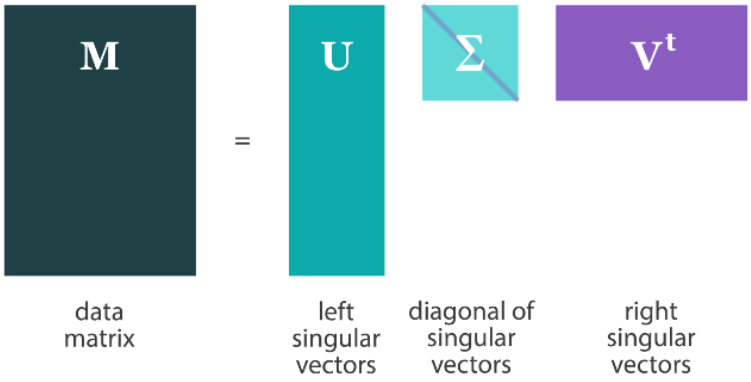


Figure (2.2) Visualization of SVD [75]

Similar to *SVD*, non-negative matrix factorization[65] is another form of linear dimensionality reduction. The difference is that *NMF* guarantees that the approximately reconstructed distances are nonnegative. Other linear (e.g., *PCA*) and non-linear (e.g., *IsoMap*) dimensionality reduction techniques have also been proposed[73, 84].

2.3.2. Random Walk Based Methods

The above-mentioned methods depend on matrices to capture the neighborhood of a node. Although the local neighborhood of a node is encoded, usually a sparse and discrete vector is learned in large-scale networks. In NLP field, same concerns also occur when learning word representation. Nevertheless, *Word2Vec*[48], a word embedding model, significantly improves the effectiveness of word representation by learning dense, continuous and low-dimensional vectors from sparse, discrete and high-dimensional space. Inspired by the model, researches established an analogy between a network and a document. Figure 2.3 can well present the network embedding process based on random walk paradigm. One can sample random walks (sequences of nodes) from the underlying network, as the same way a document is formed with sequences of words. Thus walks are generated from the original network and later can be fed into a skip-gram model. The skip-gram model aims to predict the words based on a target word within context. Details will be given in Chapter **3** Methododology.

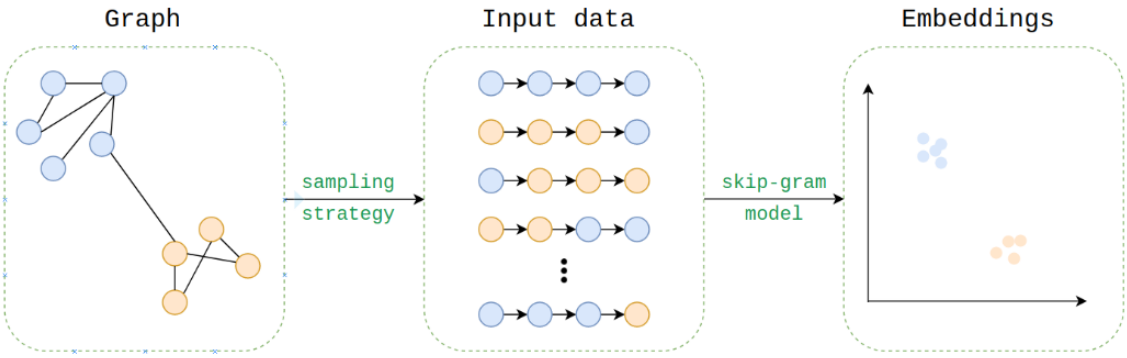


Figure (2.3) Network embedding process, from sampling node sequences from a network, to learning embedding based on the input data. [14]

However, there is no fixed sample strategy that is determined to be superior. Different sample strategies lead to various learned feature representations. Random walk based methods lie in one of the different sampling strategies on network. *DeepWalk* simply samples node sequences under a uniform distribution on static network. While, *Node2Vec*[23] defines flexible notions and capable to capture both homophily and structural equivalence attributes of networks by manipulating parameters. In this work, we tend to explore different sampling strategies based on information spreading process and their effect on preserving network information. Details would be introduced in future chapters.

2.3.3. Deep Neural Networks Based Methods

Deep Neural Networks have been used as nonlinear function learning models in many cases and have received huge successes in other fields, e.g., computer vision, NLP, information retrieval[17]. Unlike the aforementioned methods, deep neural networks based network embedding methods adopt deep models to learn representatives. Specifically, *SDNE*[78] uses deep autoencoder to preserve neighborhood of nodes. The input adjacency nodes x_i of node i are first fed into multiple non-linear layers in the encoder part. Then the output representation $\hat{x}_i$ is obtained by reversing the calculation process of encoder, which is the decoder part. The loss function takes into the account of first-order and second-order proximity simultaneously. Some other representative methods, such as *SDAE*[8] and *SiNe*[80] also propose deep learning models for network embedding. These models strive to work on high non-linearity, structure-preserving and sparsity issues.

2.3.4. Graph Neural Network

The research of graph neural networks is closely related to node embedding, whose relation is shown in Figure 2.4. As shown in the figure, both node embedding and graph neural networks involve deep learning approaches. The overlap models include graph autoencoder-based algorithms (e.g., *SDNE*[78] and *DNGR*[8] and graph convolution neural networks with unsupervised learning (e.g., *GraphSage*[24]). The learning paradigms of graph neural network include graph convolution networks[31], graph attention networks[76], graph generative networks[35, 85] and graph spatial-temporal networks[86].

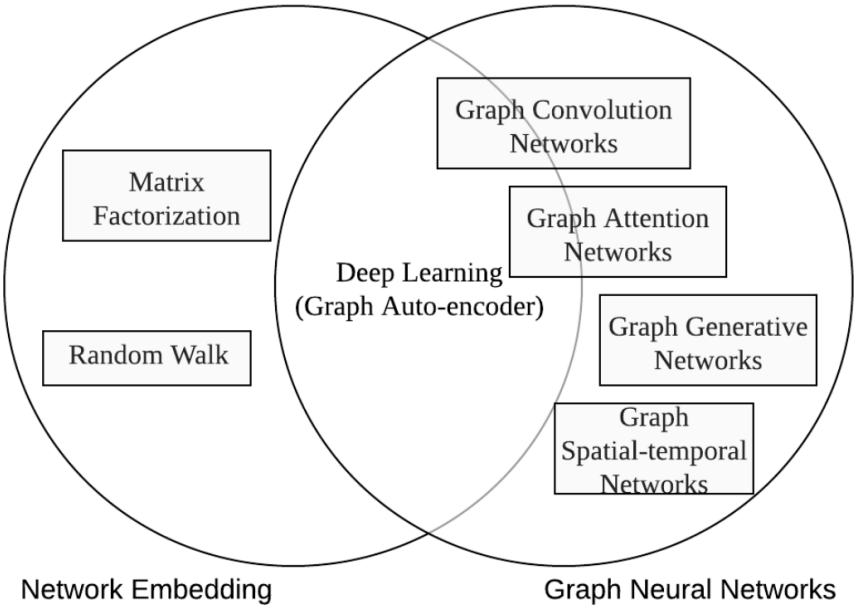


Figure (2.4) Network Embedding v.s. Graph Neural Network [82]

2.4. Dynamic Network Embedding

Most of the models introduced above lie in this category, in which the learned networks are static. However, in real world, networks are dynamic and evolve continuously over time. Many researches in network representation learning has focused on static network while neglecting the rich information on network development throughout time. Recently, a surge of works have been done focusing on these dynamic information. And this will also be our main focus in this research.

CTDNE[52] proposed a general framework for learning temporal representations based on walks that respect time. Temporally valid sequences of node connections are represented and thus issues and information loss is avoided when time is ignored[51]. The process is two-fold, selecting initial node and proceeding random walk. The neighbor selection process is under different distribution, either uniform or linear. Walks follow a chronological patterns, aligning with ascending order in time on each step, representing the order of events. Zhou et al. proposed *DynamicTriad*[91]. They applied the concept of triad, a group of three vertices, and employed a triadic closure process, from open triad to closed triad, to model the formation and evolution of networks. They also employed social homophily and temporal smoothness in the model. The hypothesis is that two vertices tend to have closer embedding if they are connected to each other and embedding vectors tend to be close in adjacent time steps. Compared to *LANE*, *DANE*[34] takes dynamic information into consideration. On every time step, changes in topology structure and attribute values are recorded. Embeddings of network structure and attribute information are then updated by leveraging matrix

perturbation theory. This online model is scalable to large-scale networks.

2.5. Network Embedding Based on Learned Information

Based on the information preserved in the network representation, Cui et al.[16] has categorized network embedding models into three types: 1. preserving network structure and properties 2. preserving side information 3. preserving advanced information. Most of the embedding models lie in the first category and strive to preserve the structure of network, which is also the case in this work. We will give some overview on network embedding methods capturing versatile information. Given the fact that a lot of information are aligned with nodes and edges in the network, additional approaches dealing with these information is necessary.

2.5.1. Preserving Structure and Property

The first category preserve structure and property information such as neighborhood structure[57], high-order proximity of vertices[7, 78] and network communities[81]. *DeepWalk*[57] was proposed to preserve network structure. It embraces the idea that node can be represented in a network as word in a document. By extracting random walks from a network and feeding it to Skip-gram, a commonly used word embedding model, *DeepWalk* learns node representations by maximizing the similarity between neighbors in the walk sequences. *Node2Vec*[23] was later on inspired by the idea and broadened the connectivity patterns in a network. By tuning the hyperparameter, *Node2Vec* is able to manipulate the random walk sampling strategy on a network, between breadth-first search(BFS) or depth-first search(DFS). It is also able to capture second order proximity of nodes. *Line*[70] was proposed for analyzing large-scale network, and able to preserve first and second order of proximity. The first order indicates the directly connected nodes and the second order preserves the contexts(neighbors) of nodes.

SDNE[78] applied auto-encoder to capture first-order and second-order of proximity jointly to preserve the network structure. The first order proximity is regarded as a supervised learning process, since the edges in the network is provided. The second order proximity is captured in the reconstruction process and is nonetheless unsupervised.

2.5.2. Integrating Additional Information

Side information includes node content or labels, as well as node types. Researches on information network[70], attributed network[36] and heterogeneous network[12] are within the range of this category. *LANE*[27], proposed by Huang et al., incorporated label information into embedding while maintaining their correlations. They first constructed two latent matrices preserving network structure and attribute information, which were then incorporated and mapped into another latent representation and finally unified. To tackle the challenges in scalability problem in network embedding, *AANE*[26] was proposed, which decomposed the modeling and optimization during joint learning process.

As for deep neural models, *HNE*[12] aims at heterogeneous network embedding, utilizing rich content and linkage information. The difference of *HNE* compared to the above-mentioned model is that it aggregates different deep architecture, Deep Neural Networks(*DNN*) and Convolutional Neural Networks(*CNN*), to handle additional information, such as texts or multimedia objects.

2.6. Application Tasks of Network Embedding

Network embedding is capable of supporting subsequent network processing and analysis tasks such as link prediction[9, 13, 79], node classification[5, 31, 49], as well as node clustering (community detection)[10, 41, 81]. Though the goal is to preserve as much information in the network, slight preference on the type of information extracted is different when dealing with different tasks. Table 2.2 shows the application tasks that have been analyzed in different models.

Table (2.2) A summary of applications of network analysis

Model	Missing Link Prediction	Future Link Prediction	Node Classification	Community Detection	Network Visualization
DeepWalk[57]			✓		
Node2Vec[23]	✓		✓		✓
LINE[70]			✓		✓
LANE[27]			✓		
AANE[26]			✓		
ComE[10]			✓	✓	✓
HNE[12]	✓		✓	✓	
SDNE[78]	✓	✓	✓		✓
CTDNE[52]		✓			
DynamicTriad[91]	✓	✓	✓		✓
Sajjad et al.[66]			✓		
DANE[34]			✓	✓	

2.6.1. Link Prediction

Among all the tasks in network analysis, link prediction is the most fundamental problems and has received a great amount of attention[9, 13, 18, 37, 39, 79]. The goal is to predict the likelihood of connection between two nodes given the existing topology structure and the attributes of nodes[39] in a network. Since the network embedding tends to learn the vector based feature of each node, similarity between nodes can be easily measured, by computing inner product or cosine similarity of two node vectors. A higher value of similarity indicates a higher probability that two nodes may be linked. The evaluation metrics include precision@k, Mean Average Percision (*MAP*)[78]. *Node2Vec*[23] tends to learn edge representations to predict missing links by comparing link prediction heuristic scores for node pair with immediate neighbor sets respectively.

In real-world scenarios, link prediction techniques are widely applied to predict unknown connections among people in social networks , which can be used to recommend friends to users. In biological networks, link prediction methods have been developed to predict relationships and interactions between proteins, which greatly reduces the expenses of empirical approaches.

2.6.2. Node Classification

Another application of network embedding is node classification. Given that nodes are assigned with one or more labels in some networks. The challenge is to predict the labels of a node and classify them to different classes. Many works have been proposed to deal with this application[7, 27, 57, 70]. The classification process is similar to the common machine learning pipeline. First a network embedding algorithm is applied to learn representations of each node, thus every node is represented by a low-dimensional vector. Then these learned features are fed into a classifier and the classifier is trained on the training set where the labels of nodes are known. With the trained classifier, we are able to infer missing or unknown labels. Usually, Micro-F1 and Macro-F1 are used as evaluation metrics for multi-label classification problem[16, 72].

Node classification task have been testing on a variety of networks. Social networks, usually extracted from communication networks among users on online platforms, are commonly used as datasets, e.g. BLOGCATALOG, FLICRK[57, 70, 72, 78]. A citation network is a type of network that represents the citation relationships between papers and authors. [56, 70] evaluate the classification performance on DBLP network[71].

2.6.3. Community Detection

As regards to community detection, the goal is to identify which community a node belonging to. Communities consist of dense connections within themselves while sparser connections between them[20]. It is well recognized that community structure reveals the organizational structures and functional components of networks and is one of the most fundamental features of networks[81]. The idea is that nodes within the same cluster are more similar[16] to each other. Community detection requires exploitation of rich node interactions, such as nodes with content and attributes[10]. Cavallari et al.[10] constructed a framework, where community embedding, community detection and node embedding mutually improved in a closed loop. Community detection techniques can be categorized based on the network applied, either static or temporal. In the static network, the topology structure do not change over time, while some communities will evolve over time in temporal networks. The most intuitive method to detect community is to cluster nodes based on their learned representations using typical clustering methods, such as Kmeans[42]. Other algorithms, such as graph partitioning[29, 77], diffusion-based algorithms[61] and modularity optimization[22] are also applied to detect communities on static networks. When taking into the account of time effect, different aspects of techniques should be incorporated. There are several strategies to detect temporal communities, instant-optimal based[3], temporal trade-off based[63] and cross-time based[2]. More information can be referred in [62]

Community detection algorithms have been widely applied to uncover the underlying community structure of various networks. For instance, in biological networks, community detection methods are used to detect functionally homogeneous proteins[33]. Lin et al.[38] identified location-concerned patterns of traffic accidents using community detection algorithms.

2.6.4. Other Tasks

There are many other tasks researched apart from the ones introduced above. Another important application of network embedding is network visualization[56, 83]. Networks are usually presented on a two dimensional space, such that users can easily get an overview on the community structure or node centrality of a sophisticated network, as shown in Figure 2.5. Anomaly detection is also widely investigated. Anomaly detection in networks attempts to infer the structural inconsistencies, aiming to find out anomalous nodes that connect to various influential communities[6, 25]. Figure 2.6 shows the anomalous node in a network, in which the red nodes A,B,C connect to a set of different communities.

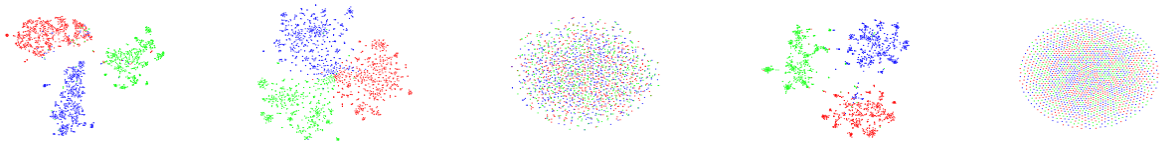


Figure (2.5) Network visualization of 20-NewsGroup by different network embedding algorithms[78]

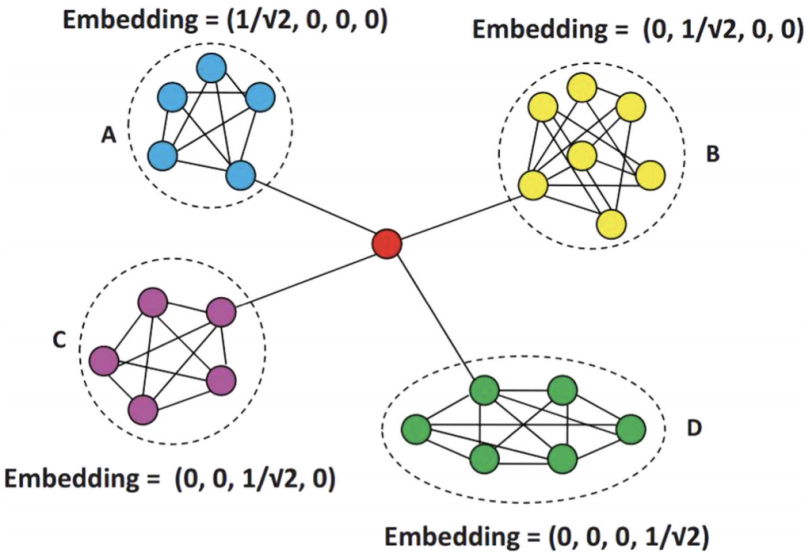


Figure (2.6) Anomalous nodes in embedding[25]

2.7. Evaluation Metrics

In the experiment, I adopt AUC as evaluation metrics.

AUC : AUC is the area under the Receiver Operating Characteristic(ROC) curve, which is one of the most fundamental evaluation metrics on link prediction task. ROC curve is a fundamental tool for diagnostic test evaluation. Before illustrating the ROC curve, I will first introduce the confusion matrix of binary classification, shown in Table 2.3.

Table (2.3) Confusion Matrix

	Positive Prediction	**Negative Prediction**
Positive Label	True Positive(TP)	False Negative(FN)
Negative Label	False Positive(FP)	True Negative(TN)

The goal of classification is to discriminate populations, as shown in Figure 2.7. If the populations overlap, the criterion value selected to divide the populations will result in some false classifications, hence false positive and false negative would exist. The goal is to find the optimal criterion value so as to achieve the best classification result.

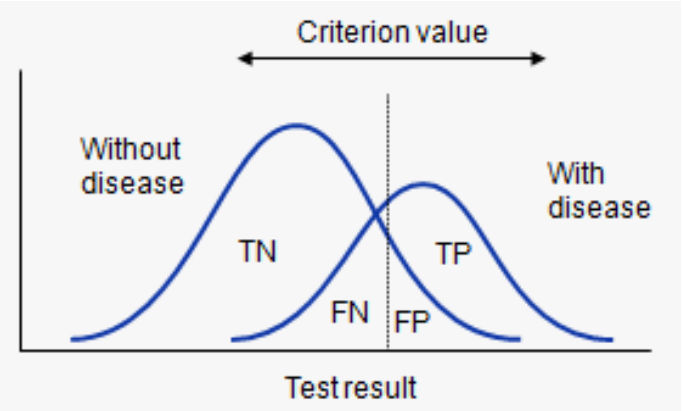


Figure (2.7) Classification of test result[1]

In ROC curve is intuitive, different criterion values available are taken into account. It keeps track on the true positive and true negative rate, illustration graph is shown in Figure 2.8. The extreme cases are either predicting all the labels correctly or wrongly. The first case will results in the curve passing through the upper left corner. While the latter case would be the populations overlap completely, thus the line would be a diagonal line.

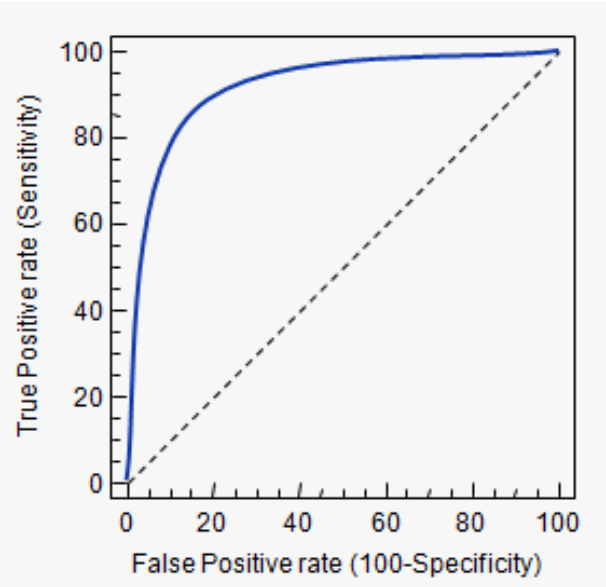


Figure (2.8) ROC curve[1]

3

Methododology

In this chapter, detailed illustration of the proposed model, SISNE, is presented. We will introduce the model by parts, each achieving one of the specific functions.

3.1. Methodology Framework

Figure 3.1 shows the methodology framework of the proposed model, *SISNE*. We aim to solve a link prediction problem by modeling our model on temporal networks. To verify our model, we first split the dataset into trainset and testset. We tend to train our model on trainset and verify the performance on the other. After we obtain the trainset, we simulate a SIS spreading process on the network and generate spreading trees. We then extract trajectories from the trees and feed the input into skip-gram model to learn embeddings of nodes. The following sections will explain each procedure in details.

3.2. Information Spreading

Information spreading (also known as information spreading) has yielded wide attention throughout years and abundant empirical analysis has been obtained. The application field is various[90], from different network platforms and applications, such as instant messaging, social network interactions, email and mobile communication, to information spreading phenomenon, like broadcasting, information sharing, crowdsourcing and etc. Epidemic models are the most widely used among all mathematical models for information spreading, which is also the focus of this paper.

Consider the information spread in a group of people. Based on whether information has reached a person and whether the information is spread through by an 'awared' person, models can be resulted into *SI*, *SIS*, and *SIR*. *SI* and *SIS* are applied in this paper and the model is shown in Figure 3.2. As referred from the name of the models, people can be labeled as different compartments: 1. *S*: the susceptible state, indicating people unaware of the information yet; 2. *I*: the infected state, indicating information has reached this person; 3. *R*: the recovered state, people who aware of the information make no response to it and thus will not transmit it to others. Among the three models, *SI* is the simplest case, in which only state *S* and *I* are considered. *SI* model can be illustrated by differential equations:

$$\begin{cases} \frac{ds(t)}{dt} = -\beta s(t) i(t) \\ \\ \frac{di(t)}{dt} = \beta s(t) i(t) \end{cases} \tag{3.1}$$

, where β indicates the infection rate(the probability an infected person infect others), $s(t)$ and $i(t)$ denotes the proportion of susceptible and infected people at time t.

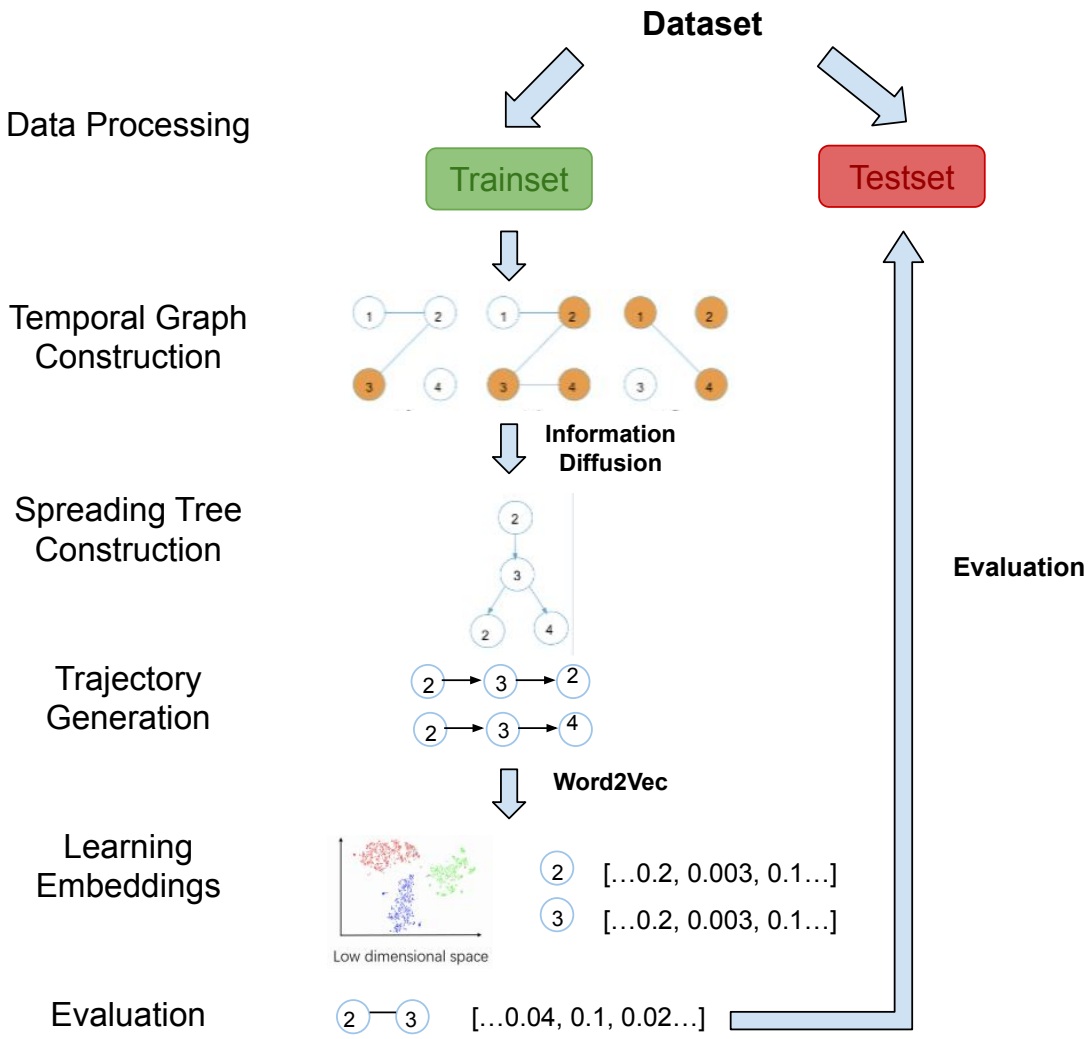


Figure (3.1) SISNE Framework

SIS model involves a recover process. Individuals who have already known about the information will ignore it and become susceptible again. The differential equations for *SIS* model is :

$$\begin{cases} \frac{ds(t)}{dt} = -\beta s(t)i(t) + \gamma i(t) \\ \frac{di(t)}{dt} = \beta s(t)i(t) - \gamma i(t) \end{cases} \tag{3.2}$$

, where γ is recover rate (the probability an individual in I state recover and become S state).

3.3. SIS Sampling Strategy

Many network embedding models focusing on random walk has been proposed in recent years. In this thesis report, information diffusion strategy is applied on networks to generate node pair corpus. Under the diffusion spreading process, nodes can be in either susceptible or infectious state. Every susceptible node would have a certain probability, infectious rate β, to be infected and thus convert to an infectious state. In SI model, once infected, nodes would remain as infectious permanently. While in the case of SIS, nodes could return to susceptible state after infection and thus may have repeated infections.

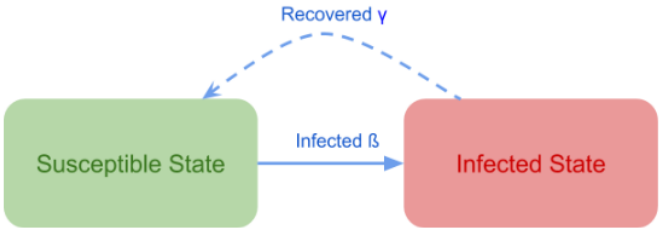


Figure (3.2) SI(S) model

The process can be described into two steps: 1. **spreading tree (ST) construction**; 2. **trajectory generation**.

3.3.1. Spreading Tree (ST) Construction

First, we randomly select a node u_i as the seed and a time step $t_i \in [0, T]$ as the starting time of the spreading process. Node u_i is infected and the other nodes are in the susceptible state at time step t_i. The spreading follows the time step of the temporal network. At every time step, the infected node can infect its susceptible neighbor with probability β and can recover with probability γ. Therefore,we can construct the spreading tree starts from node u_i as $\mathcal{T}_i(\beta, \gamma)$, which records the union of contacts that the spreading passes. As the infected node can recover to the susceptible state after a period, therefore a node can appear in the spreading tree $\mathcal{T}_i(\beta, \gamma)$ more than once. This process continues till the last time step or all the nodes in the network are in suspected state, indicating no infected node exists in the network. Noted that, these temporal interactions can only be traversed in ascending time slots.

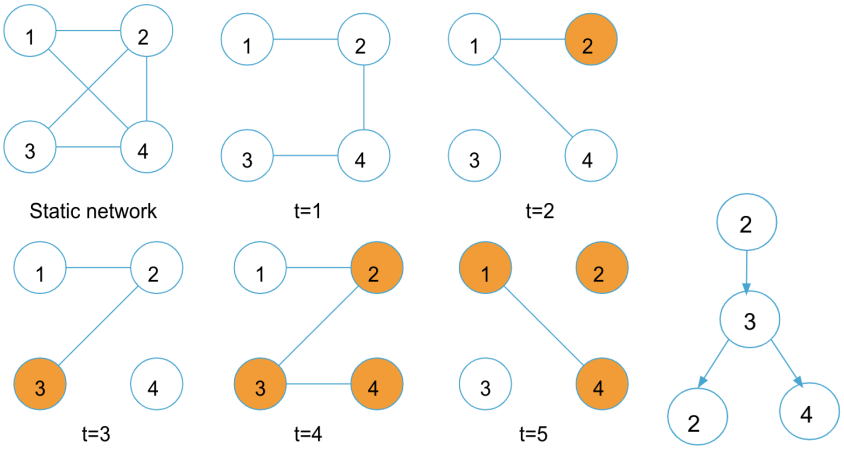


Figure (3.3) Temporal network snapshots and extracted spreading tree

Figure 3.3 shows an example of temporal network and the node state along time steps. Noted that the infection or the susceptible sates are all presumed and may not the the same in real-time processing. In t_2, node u_2 is randomly selected. We can assign u_2 as the root node in the spreading tree $T_2(\beta, \gamma)$. In the following time step t_3, node u_3 is infected by u_2 through $l_{2,3}$ and afterward u_2 recovered and became susceptible again. Thus u_3 is added into the spreading tree as the child node of u_2. Similar process continues along the time order. When a susceptible node u_j is infected by infected node u_i, a corresponding contact is generated in the spreading tree. It is noticeable that u_2 appears twice in the hypothetical spreading tree. That is due to the fact that u_2 has altered the

infection state twice, from infectious to susceptible and vice versa. As long as there is an change in infection state, the update will be recorded in the spreading tree. In order to generate enough corpus, a certain number of spreading trees are extracted from the dynamic networks. Some nodes may serve as root node in different spreading trees. Due to the randomness of SIS process, two trees with the same root node may result in various structures. In this stage, length and depth of the spreading tree is not restricted. It only depends on the time interval, from the moment initialing the starting time step till the last recorded event.

Algorithm 1 ST_Construction

Input: $\mathscr{G} = (\mathscr{N}, \mathscr{L})$,$[0, T]$, β, γ
Output: $\mathscr{T}_i(\beta, \gamma)$

1: Randomly choose node u_i, t_i as the seed and the starting time
2: u_i is set as root of ST $\mathscr{T}_i(\beta, \gamma)$
3: $t = t_i$
4: **while** no infected node **or** $t = T$ **do**
5: Operate the SIS spreading process following time stamp $t \in [t_i + 1, T]$
6: When temporal neighbors are infected, add nodes to ST
7: **end while**
8: **return** $\mathscr{T}_i(\beta, \gamma)$

3.3.2. Generating Trajectories

For each spreading tree $\mathscr{T}_i(\beta, \gamma)$, we randomly select m_i different trajectories from $\mathscr{T}_i(\beta, \gamma)$, where $m_i \propto d(i)$ with the assumption that the higher the degree of a node, more trajectories will be extracted from a spreading tree. All the trajectories will be gathered and serve as input data and we define the corpus set as D. We define the size of input data as $\mathscr{B}$, which defines the total number of node in the extracted corpus set D. $\mathscr{B}$ is the multiplication of $|N|$ and X, where X determines how large is the input size.

In a spreading tree, the root node is regarded as the start of the trajectory. The following node step is sampled under a uniform distribution among the children. And the selection strategy goes on until it reaches the leaf node or exceed the length of the trajectory. Also in the same example in Figure 3.3, two trajectories are possible to be extracted, $\{2,3,2\}$ and $\{2,3,4\}$. Later on, these trajectories will be gathered and fed into the Skip-gram model, which is illustrated in the following session.

Algorithm 2 Generating Trajectories

Input: $\mathscr{G} = (\mathscr{N}, \mathscr{L})$,$[0, T]$, $\mathscr{B}$, L_{min}, L_{max}, β, γ, m_i
Output: a set of node corpus D

1: Initialize number of context windows $C = 0$
2: Initialize node corpora set $D = \varnothing$
3: **while** $\mathscr{B} - C > 0$ **do**
4: $\mathscr{T}_i(\beta, \gamma)$ = ST_Construction
5: Randomly choose trajectory D_g $(g = 1, \ldots, m_i)$ from $\mathscr{T}_i(\beta, \gamma)$, $s.t.$ $L_{min} < |D_g| < L_{max}$
6: Add the temporal trajectory D_g to D, $g = 1, \ldots, m_i$
7: $C = C + (|D_g| - \omega + 1)$, $g = 1, \ldots, m_i$
8: **end while**
9: **return** D

3.4. Skip-gram Model

As mentioned, Skip-gram aims to predict many context words based on a target word. The goal is to maximize the probability of context words based on target word. Since it is a deep learning architecture, the neural structure contains hidden layers, as well as output layer. The weight matrix contains in hidden layer is the embedding we would like to train, the embedding for words. Similarly, if our input are nodes then we can train embedding for nodes. In such a way, we achieve node representation.

3.4.1. An Analogy Model – Word2Vec

In order to obtain embedding for nodes, a language model, *Word2Vec*[47], is applied. Word embedding has been widely analyzed in the field of NLP(Natural Language Processing) and many up-to-date models have achieved great performance. The reason why scientist link word representation and node representation together is that word and node share similar context. Consider a sentence is a sequence of words and a random walk is a sequence of nodes. In order to train a language model, we may need a large amount of texts so as to obtain enough contexts, which is also the case in network embedding. We may also need to extract a good number of random walks from the network so as to capture the network structure. CBOW and Skip-gram are two model architectures within *Word2Vec* model. CBOW aims to predict a word based on context while Skip-gram works in the opposite way, predicting context given a target word. Skip-gram is the model applied in node embedding, as shown in Figure 3.4.

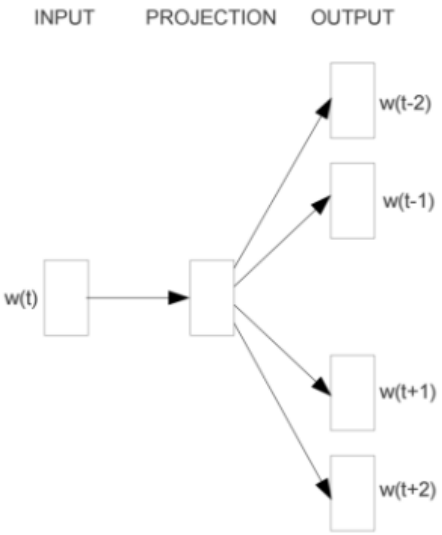


Figure (3.4) Skip-gram model

3.4.2. Node Pair Generation

After we extract walks from the temporal networks, one step is needed before applying Word2Vec model to generate embedding vectors, which is node pair generation. The goal of Word2Vec is to minimizes the distance of embedding vectors of nodes in a node pair. In this session, how to obtain node pairs is illustrated. The parameter, window size ws makes an influence on retrieving node pair corpus set. Below in Figure 3.5 shows the example of the noe pair retrieved from a walk with window size 2. Each node in the walk is regarded as target word once. Then we will look at its neighbors within the window size. The nodes within two hops are considered as the neighbors. If n_1 is target node, then its neighbors are n_2 and n_3. Then node pairs are generated with every combination of the target node and its neighbors. The node pair corpus is determined by parameter ws. With a

larger ws, a larger node pair corpus is generated and perhaps more diverse, due to more neighbors are assigned to the target node.

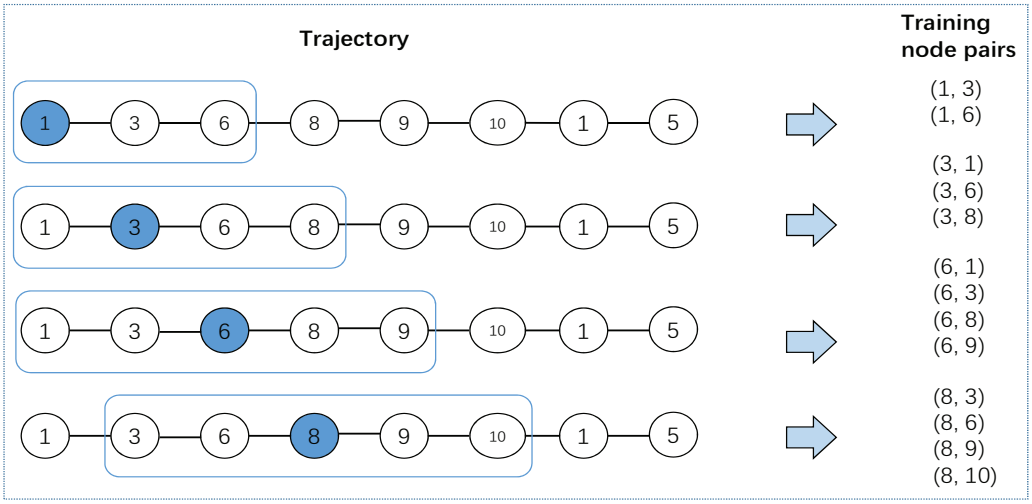


Figure (3.5) Extracting node pair from a walk

The parameter settings used for generating embedding vectors are in line with typical values used for *DeepWalk* and *Node2Vec*. Specifically, $d = 128$, $ws = 10$.

3.4.3. Temporal Network Structure Modeling

In this subsection, we give mathematical illustration on the modeling of temporal network structure. We model the local pairwise proximity between node u_i and node u_j by a conditional probability

$$p(u_j|u_i) = \frac{exp(f(u_i,u_j))}{\sum_{u_k \in \mathcal{N}} exp(f(u_i,u_k))} \tag{3.3}$$

We set f as the dot product of the embedding vector of u_i, u_j [23, 57], which can be viewed as shallow model for modeling node pairs proximity, i.e.,

$$f(u_i,u_j) \quad = \quad \overrightarrow{h_i} \cdot \overrightarrow{h_j} \tag{3.4}$$

We define $N_S(u_i) \in \mathcal{N}$ as the neighboring set of node u_i sampled from the SIS spreading process. We use the conditional independence assumption that the proximity between node u_i and any one neighbor is independent of all the other neighbors. Given a center node u_i and its neighboring node $u_j \in N_S(u_i)$, the proximity between node u_i and all its neighbors is defined as

$$p(N_S(u_i)|u_i) = \prod_{u_j \in N_S(u_i)} p(u_j|u_i) \tag{3.5}$$

To model the parameters of the framework, the objective function is given as follows:

$$
\begin{aligned}
maximize \quad \mathcal{O} &= \prod_{u_i \in \mathcal{N}} \prod_{u_j \in N_S(u_i)} p(u_j|u_i) \\
&= \sum_{u_i \in \mathcal{N}} \sum_{u_j \in N_S(u_i)} log\, p(u_j|u_i) \\
&= \sum_{u_i \in \mathcal{N}} \sum_{u_j \in N_S(u_i)} log \frac{exp(f(u_i,u_j))}{\sum_{k=1}^{N} exp(f(u_i,u_k))} \\
&= \sum_{u_i \in \mathcal{N}} \sum_{u_j \in N_S(u_i)} log \frac{exp(\overrightarrow{h_i} \cdot \overrightarrow{h_j})}{\sum_{u_k \in \mathcal{N}} exp(\overrightarrow{h_i} \cdot \overrightarrow{h_k})} \\
&= \sum_{u_i \in \mathcal{N}} [-log Z_{u_i} + \sum_{u_j \in N_S(u_i)} \overrightarrow{h_i} \cdot \overrightarrow{h_j})]
\end{aligned} \tag{3.6}
$$

where $Z_{u_i} = \sum_{u_k \in \mathcal{N}} exp(\overrightarrow{h_i} \cdot \overrightarrow{h_k})$. To compute Z_{u_i} for each node u_i, we need to traverse the entire node set $\mathcal{N}$, which is of high computational complexity. To solve this problem, we approximate Z_{u_i} by using negative sampling [48]. The idea is for node pair (u_i, u_j), we sample multiple negative edges according to some noisy distribution. Therefore, the objective function Eq. (3.6) can be approximated by

$$
maximize \quad \mathcal{O} = \sum_{u_i \in \mathcal{N}} [\sum_{u_j \in N_S(u_i)} log\sigma(\overrightarrow{h_i} \cdot \overrightarrow{h_j}) + \sum_{u_k \in N_S^{ns}(u_i)} log\sigma(-\overrightarrow{h_i} \cdot \overrightarrow{h_k})] \tag{3.7}
$$

The number of negative samples is ns for each node and the sampling distribution is $F(v) \propto d(v)^{3/4}$, in which node with high degree has higher probability to be sampled as negative nodes. Therefore, the objective function $\mathcal{O}$ is maximized to produce neighboring nodes u_i, node u_j to be close and node u_i, node u_k to be distant in the embedding space as node k is a negative sample.

4

Baseline Models

In this chapter, we will introduce the details of the baseline models, including static network embedding methods, as well as temporal embedding models.

4.1. DeepWalk

DeepWalk[57] is one of the first network embedding models that applies random walk to learn network structure. It is able to preserve the neighboring structures of nodes. The concept is inspired by a language model, *Word2Vec*, that learns word embedding. *DeepWalk* proposed an analogy between nodes in a network and words in a document. Motivated by this analogy, *skip-gram* model is adopted by *DeepWalk* to learn node representations. The goal is to maximize the probability of the neighbors of a target node u_i in a walk sequence. The formula is as follows:

$$max \Pr(\{u_{i-ws}, \cdots, u_{i+ws}\} \setminus u_i | \Phi(u_i)) = \prod_{\substack{j=i-ws \\ j \neq i}}^{i+ws} \Pr\left(u_j | \Phi(u_i)\right) \tag{4.1}$$

, where ws is the window size, $\phi(u_i)$ is the representation of node u_i and $\{u_{i-w}, ..., u_{i+w}\}$ is the local context nodes of node u_i. Hierarchical soft-max[57] is used to efficiently retrieve the embeddings.

DeepWalk consists of two components: random walk generator, and second, an update procedure. When generating random walks, a random node u_i is uniformly sampled as the root of the random walk W_{v_i}. A walk extends by sampling uniformly from the neighbors of the last node visited until the maximum length is reached. Here the walk length (wl) is fixed, while there is no restriction for the random walks to be of the same length. In practice, the implementation specifies a number of random walks of a fixed length to start at each node.

4.2. Node2Vec

Node2Vec[23] is served as a static baseline. It follows the idea of extracting random walks from the network and at the same time smoothly interpolates between BFS(Breath First Search) and DFS(Depth First Search). This is achieved by a biased neighbor selection process, compared to *DeepWalk*[57], where unbiased sampling strategy is applied. Let n_i denote the ith node in a walk, starting from n_0. The sampling probability of n_i follows the distribution:

$$P(n_i = x | n_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} & \text{if } (v, x) \in E \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

, where π_{vx} is the transition probability from v to x, and Z is the normalizing constant.

Two parameters **Return parameter** p and **In-out parameter** q are designed to define a second order random walk, where q is responsible for the interpolation between BFS and DFS. A second order proximity indicates the neighborhood information of nodes and first order proximity reveals the direct connection between nodes. Figure 4.1 shows the case when a random walk just traversed edge(t,v) and node v was deciding on its the next step.

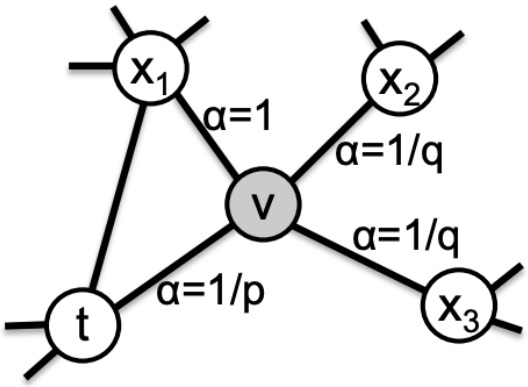


Figure (4.1) Figure from[23]

The transition probability between node v and its neighbors follows $\pi_{vx} = \alpha_{pq}(t,x) * w_{vx}$, where:

$$\alpha_{pq}(t,x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases} \tag{4.3}$$

d_{tx} indicates the shortest distance between node t and node x. These relations reveal the ability of p and q in deciding on the next step to take. p controls the probability of the recurrence of a node in a random walk. q interpolates between BFS and DFS. By tuning these parameters, node2vec is able to capture the neighbor structure of nodes with both first-order proximity and second-order proximity.

4.3. CTDNE

CTDNE[52] takes the time information into consideration when taking the next step during random walk sampling. A contact indicates an edge in the network along with a unique timestamp. What is worthwhile to mention is that the random walk extracted from the temporal network follows a time-ascending order. Example is revealed in Figure 4.2. A random walk is possible in the sequence of $v1, v2, v3$ instead of $v4, v1, v2$ due to the fact that events between $v1$, $v4$ appears later compared to $v1$, $v2$ node pair link.

CTDNE consists of two steps: 1. **Initial Temporal Edge Selection**: an edge is first selected and an initial node is chosen to be the start of a walk. 2. **Temporal Random Walk Generation**: a walk is generated by sampling a sequence of neighbors starting from the node, following the time step. The idea is similar to *Node2vec*. *Node2vec* considers the shortest distance between nodes while *CTDNE* only samples the contacts with a larger timestamp (contacts exists later in time) compared to the initial status and apply either unbiased or biased sampling strategy in both stages. Unbiased sampling strategy indicates that all the contacts are sampled under uniform distribution. A biased selection process tends to select the contacts that happened later. The biased sampling strategy is under a softmax function. Below presents the sampling strategies by listing the formula equations.

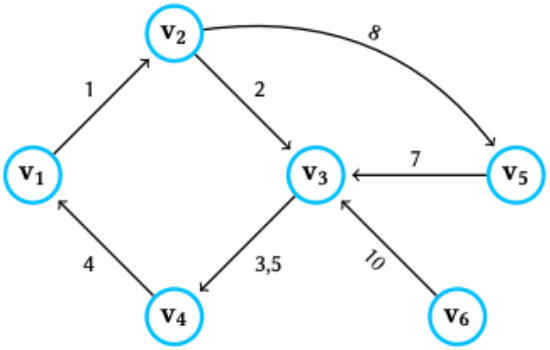


Figure (4.2) Dynamic network[52]. Edges are labeled by time.

4.3.1. Initial Temporal Edge Selection

In this stage, an initial contact along with its associated time is sampled from an arbitrary distribution[52]. The distribution used to select the initial temporal contact can either be uniform in which case there is no bias or the selection can be temporally biased using an arbitrary weighted (non-uniform) distribution. For biased sampling, more temporal walks from contacts closer to the current time point are chosen, as the contacts in the distant past may be less predictive or indicative of the state of the system now.

1. **Unbiased Sampling**: In this case, uniform distribution is applied. Each contact $l_{i,j,t} = (u_i, u_j, t) \in \mathscr{L}_T$ has the same probability of being chosen as an initial edge.

$$\Pr(l) = 1/|\mathscr{L}_T| \tag{4.4}$$

2. **Biased Sampling**: Two techniques are introduced in the original paper, weighted distribution on exponential or linear functions. In this project, only linear function is applied.

$$\Pr(l) = \frac{\eta(l)}{\sum_{l' \in \mathscr{L}_T} \eta(l')} \tag{4.5}$$

, where η maps each contact to an index with $\eta(l) = 1$ for the earliest contact.

4.3.2. Temporal Random Walk

The next node in a temporal random walk can be chosen from the set $N_t(u_i)$. We define $N_t(u_i)$ as the temporal neighbor set of u_i at time t, given contact $l_{i,j,t} = (u_i, u_j, t) \in \mathscr{L}_T$. Again, either uniform or biased distribution is applied in the neighbor selection.

1. **Unbiased Sampling**: Node u_k has the following probability of being selected:

$$\Pr(u_k) = 1/|N_t(u_i)| \tag{4.6}$$

2. **Biased Sampling**: Biased distribution on linear function is applied. Different from the previous contact biased sampling strategy, neighbors are chosen with time closer to the current node.

$$\Pr(u_k) = \frac{\delta(u_k)}{\sum_{u_k' \in N_t(u_i)} \delta\left(u_k'\right)} \tag{4.7}$$

,where δ sorts temporal neighbors in descending order time-wise.

5

Experimental Setup

The proposed model is evaluated on five real-world dataset. In this chapter, details and attributes of data is presented as well as the experimental results.

5.1. Empirical Networks

In order to evaluate the effectiveness of the proposed model, five real-world networks are included, x social networks, y rating networks. A link prediction task is performed on all these data.

Details about each dataset are presented below:

- **ia-contacts_hypertext2009**[64, 69]: It is a dynamic contact social network, collected during ACM Hypertext 2009 conference. Attendees were asked to wear radio badges that captured the face-to-face proximity. The data last for 2.5 days. Users in contact were recorded with the form <$user_i$, $user_j$, t>

- **ia-radoslaw-email**[46, 64]: It's a social network extracted from an internal email communication. The activities are sending and receiving emails between employees from a mid-sized company. The network is directed, with the first node indicating the sender and the other the receiver.

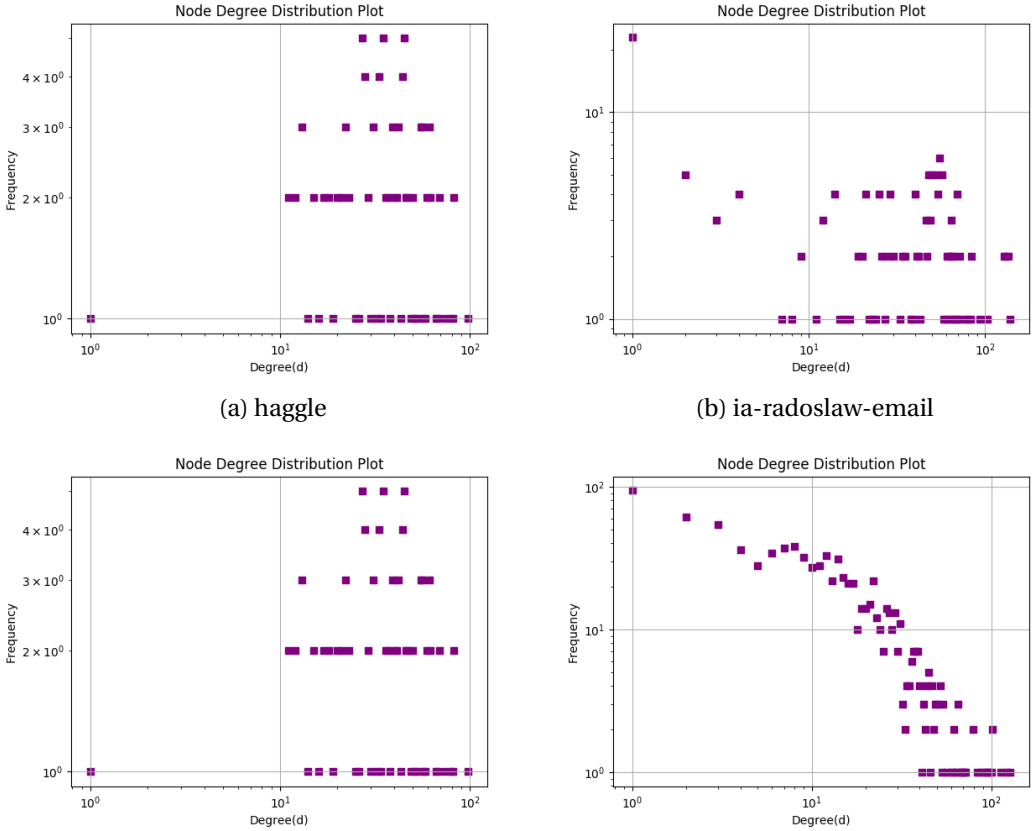
- **haggle**[11, 32]: The undirected network capture interactions between people measured by wearable wireless device. Contacts were recorded along with time. A node represents a person; an edge indicates a contact between two people.

- **fb-forum**[53, 64]: This dataset is collected from Facebook-like online community of students. In the community, users could create groups and post messages to groups where they are a member. The dataset record more than 33k user activities in the forum in 2004. The original purpose of generating this dataset is to redefine the clustering coefficients for two-mode networks. It was later applied in [52] for link prediction. That is also the task I strive to solve in this project.

Summary of datasets are shown in Table 5.1, with number of edges, number of nodes, maximum degree and average degree presented. In addition, degree distribution is presented in Figure 5.1. Experiments are conducted on these five directed or undirected networks. All the networks will be considered as undirected in the experiments.

Dataset	N	T	$\|\mathscr{C}\|$	$\|E\|$	E den.	Deg. avg	W avg	CC	wCC
ia-hypertext2009	113	5246	20818	2196	0.3470	38.87	9.48	0.5348	0.0059
ia-radoslaw-email	167	57842	82927	3251	0.2345	38.93	25.51	0.5919	0.0023
haggle	274	15662	28244	2124	0.568	15.5	13.298	0.6327	0.0312
fb-forum	899	33515	33720	7046	0.0175	15.68	4.79	0.0637	0.0012

Table (5.1) A summary of attributes of networks



(a) haggle (b) ia-radoslaw-email

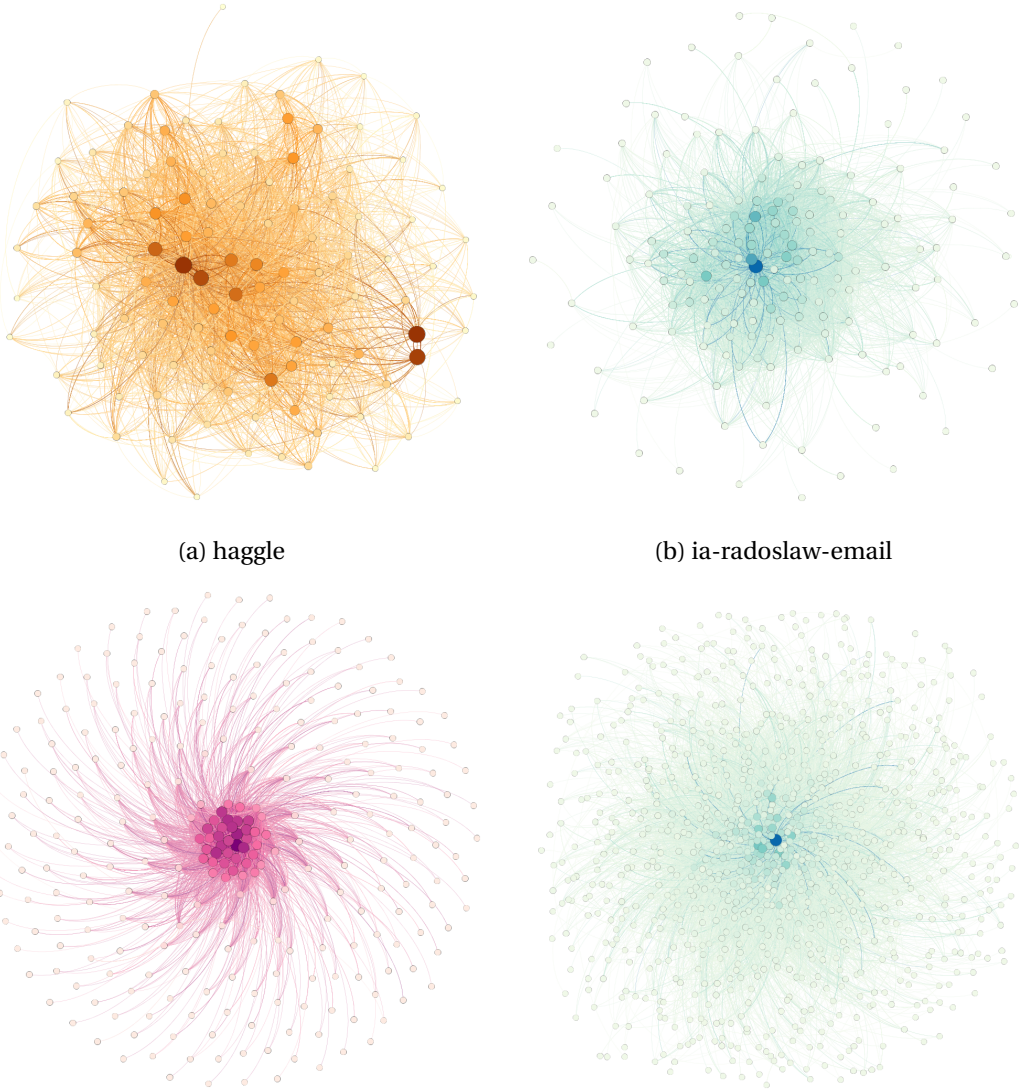
(c) ia-contacts_hypertext2009 (d) fb-forum

Figure (5.1) Degree Distribution.

5.2. Network Processing

In this project, I conduct link prediction on five datasets. Link prediction is to predict links or edges that have never existed in the train dataset. Thus, a certain proportion of edges should be removed from the original dataset when data is processed. First, convert the original dataset as a static network. 75% of edges are randomly chosen. Alongside with the corresponding timestamps from the original network, trainset is constructed. The remaining 25% of edges remain to be testset. I have also considered negative sampling, which was also applied in [23, 52]. The idea of negative sampling is to generate links between nodes where links do not exist in original dataset. The number of negative sampling links remains the same as positive links in the above-mentioned test samples. The real links are labeled positive while the manufactural ones are labeled negative.

In order to make the results solid, the original data is split five times. The trainset is randomly selected in every split. In order to increase efficiency, all the spreading trees retrieved are stored in json files. Walks are extracted from the stored tree files, in which way it is not necessary to rerun the code when some hyper-parameters change, for example, context size **B**. In addition, it is also possible to extract different walks under the same parameters. Under every split files, walks would



(a) haggle (b) ia-radoslaw-email

(c) ia-contacts_hypertext2009 (d) fb-forum

Figure (5.2) Network Visualization

be extracted from the spreading tree file for 10 times. Thus every combination of hyper-parameters could generate 50 results. The reason to repeat these procedure is to verify whether different split of dataset would results in various performance and to compensate the randomness of the extraction of walks.

6

Results and Analysis

This chapter presents the experiment results, including performance against baseline models, as well as hyper-parameter sensitivity analysis. In the end, we will show how the properties of retrieved networks would change by applying different spreading processes.

6.1. Parameters Overview

We first introduce the parameters considered in this model, some of which adopt the value in line with default setting used for other baselines. The detailed information of the hyper-parameters are shown below:

- $\mathscr{B}$: input context size, which is the multiplication of **N** and **X**, $\mathscr{B} = N * X$. **X** is a tunable parameter. The values tested in the experiment are $\{1, 2, 5, 10, 25, 50, 100, 150, 200, 250, 300, 350\}$.
- β: infection rate, probability that an infected node influence an susceptible node in information diffusion process. The values tested in the experiments are $\{0.001, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$. (For further fine-grained analysis below, we also tried out values such as 0.005 and 0.05)
- γ: recovery rate, probability that an infected node recoveries to susceptible state again.
- $\frac{\beta}{\gamma}$: the ratio between β and γ. Combined with β, it manipulates the SI(S) process. The selection range lies in $\{2, 5, 10, 25, 50, 100, 200, \infty\}$. When the ratio is infinite ($\gamma = 0$), the SIS spreading model degenerates to SI model.
- wl: trajectory length, which defines the length of each extracted trajectories from networks. In this work, the default value is 20, indicating the maximum length of a trajectory is 20. The average trajectory length is denoted as $< wl >$
- ws: window size, which determines the hop distance range from center node to its context nodes when generating node pairs from trajectories. Default value is 10.
- d: embedding dimension size, which indicates the dimension of the learned embedding of nodes. Default value is set as 128.

The most important parameters taken into consideration are the manipulation factors of SI(S) model, β and $\frac{\beta}{\gamma}$. We conducted the experiments using 5-fold cross-validation and each fold repeating 10 times in order to even the randomness from the spreading process. We tested on 25% labeled data from the network combined with negative samples with the same amount. Hyper-parameters were tested with a grid search over $\beta, \frac{\beta}{\gamma}, X$.

6.2. Model Comparison

For link prediction task, we use AUC to evaluate our model as well as the baseline models, which is shown in Table 6.2. The proposed embedding method is named as *SISNE*. We consider both the static network embedding methods, i.e., *DeepWalk* and *Node2Vec*, as well as temporal network embedding method, i.e., *CTDNE*, as the baseline models. The detailed description of the models have been given in Chapter 3. In all cases, all the parameters are kept consistent among models, including X, d, ws and wl. The parameter, X, is set to be the optimal value when *SISNE* achieves the best results (haggle: 150, ia-radoslaw-email:200, ia-contacts_hypertext2009: 250, fb-forum: 50).

Table (6.1) Performance evaluation of different network embedding methods

Dataset	*DeepWalk*	*Node2vec*	*CTDNE*	*SISNE*
haggle	0.3823	0.7807	0.7796	**0.8051**
ia-radoslaw-email	0.6439	0.6619	0.6575	**0.7329**
ia-contacts_hypertext2009	0.5209	0.5572	0.6038	**0.6740**
fb-forum	0.5392	0.6882	0.6942	**0.7125**

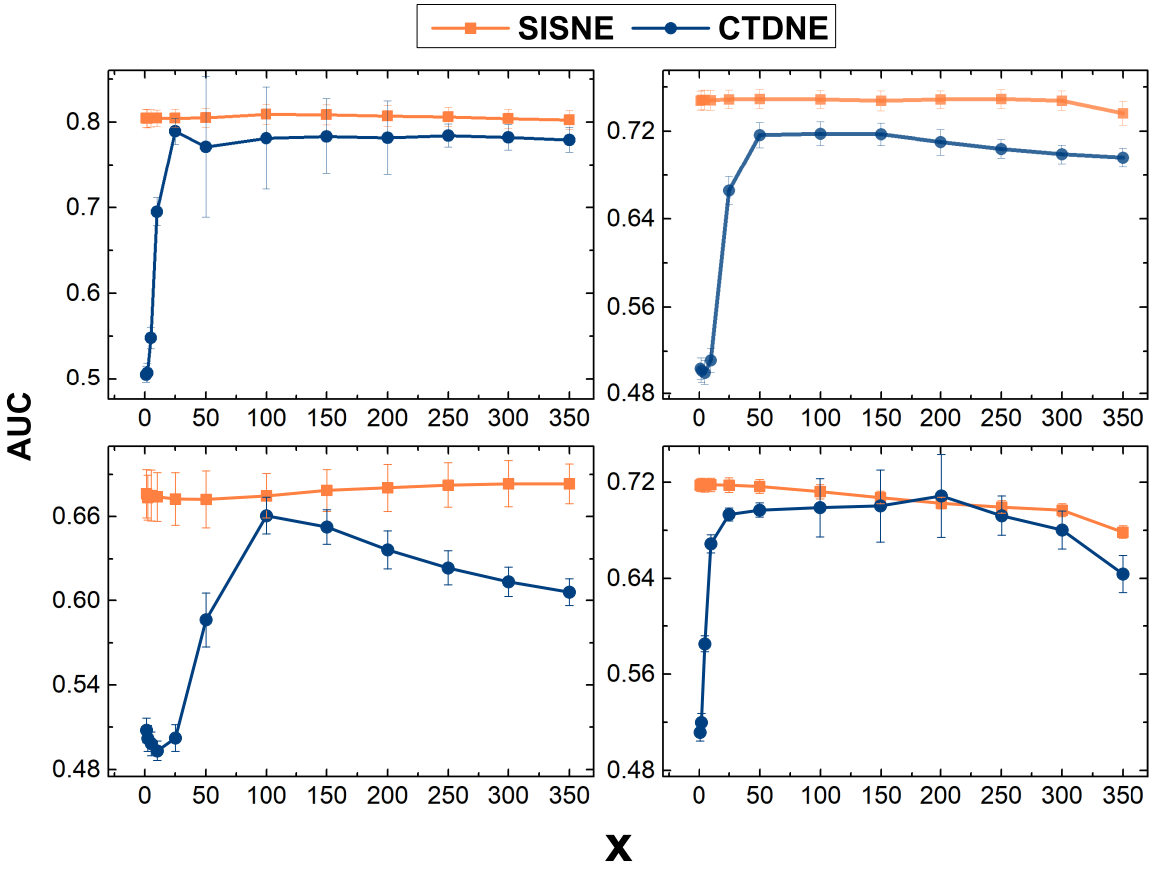


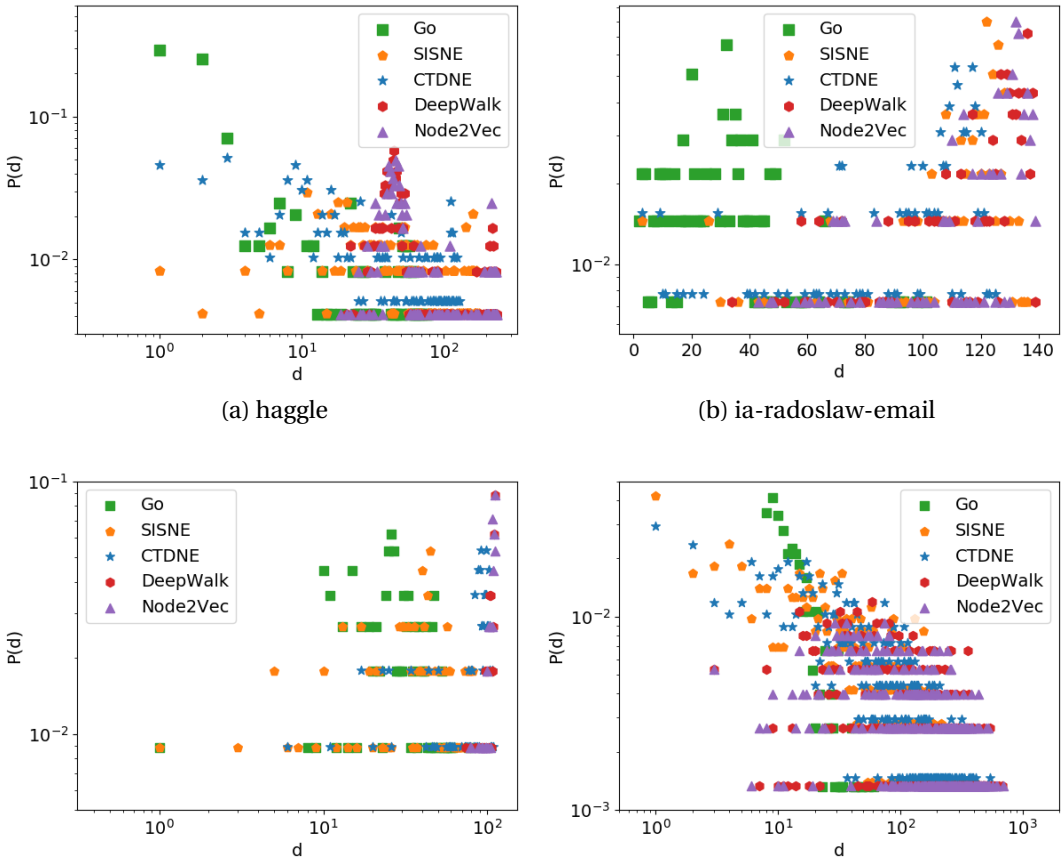
Figure (6.1) Performance evaluation with the change of X, where X determines the number of trajectories generated from the SIS model.

As shown in Table 6.2, applying information spreading model on network allows *SISNE* to outperform the other baseline algorithms on all four datasets under the same parameter settings. Compared to the most competitive baseline model, *CTDNE*, the gain can be up to 7.54% in ia-radoslaw-email and 7.01% in ia-contacts_hypertext2009. The convincing performance of *SISNE* and *CTDNE*

strongly show that we need to consider the temporal aspect of the evolving networks. As expected, *Node2Vec* outperformed *DeepWalk* in all four datasets, due to its more flexible strategy to learn the attributes of network, balancing between homophily and structural equivalence [23].

The advantage of *SISNE* also lies in its consistent performance varying the amount of input data, which is determined by X. The result is shown in Figure 6.1. Here we compare the result of the proposed model with the most competitive baseline, *CTDNE*. *SISNE* achieves large improvements over the *CTDNE* model across different amount of input data. Even with small amount of data, for example, the first points given in the figure $X = 2$, *SISNE* shows very high performance. In addition, the performance stays relatively stable within the change of X. However, for the *CTDNE* model, there is an optimal value of X, which varies across different networks. When X is small, the performance drops dramatically.

6.2.1. Degree Distribution Analysis



(a) haggle (b) ia-radoslaw-email

(c) ia-contacts_hypertext2009 (d) fb-forum

Figure (6.2) Degree distribution of G_O, SISNE and baselines on four datasets.

To further investigate the attributes of the learned information using different models, we compare the degree distribution of different models compared with G_O, which is shown in Figure 6.2. We observe that static network embedding tend to have nodes with higher degree, points showing in the right panel of the plot, indicating that *DeepWalk* and *Node2Vec* prefer to capture nodes with high degree. On contrary, temporal embedding models, e.g., *SISNE* and *CTDNE*, are possible to capture nodes with all possible degree, showing a great ability to capture the structure of a network.

Table (6.2) Average Degree of different models

Dataset	G_O	*SISNE*	*CTDNE*	*DeepWalk*	*Node2Vec*
haggle	13.17±19.83	62.62±49.84	41.71±40.49	79.60±65.64	81.01±66.03
ia-radoslaw-email	31.55±20.16	109.46±25.57	90.25±31.76	115.22±24.40	115.20±24.11
ia-contacts_hypertext2009	29.15±13.77	42.60±22.56	80.20±25.84	108.96±7.42	109.63±6.77
fb-forum	7.08±7.43	66.32±64.42	70.28±74.62	160.29±126.76	160.36±126.45

6.3. Hyper-parameter Analysis

The *SISNE* involves a variety of hyper-parameters as listed earlier in this Chapter. The following part investigates how different choices of parameters affect the performance on link prediction task across different datasets. Specifically, we examine what kind of parameters we should choose in the information diffusion process in order to get better performance.

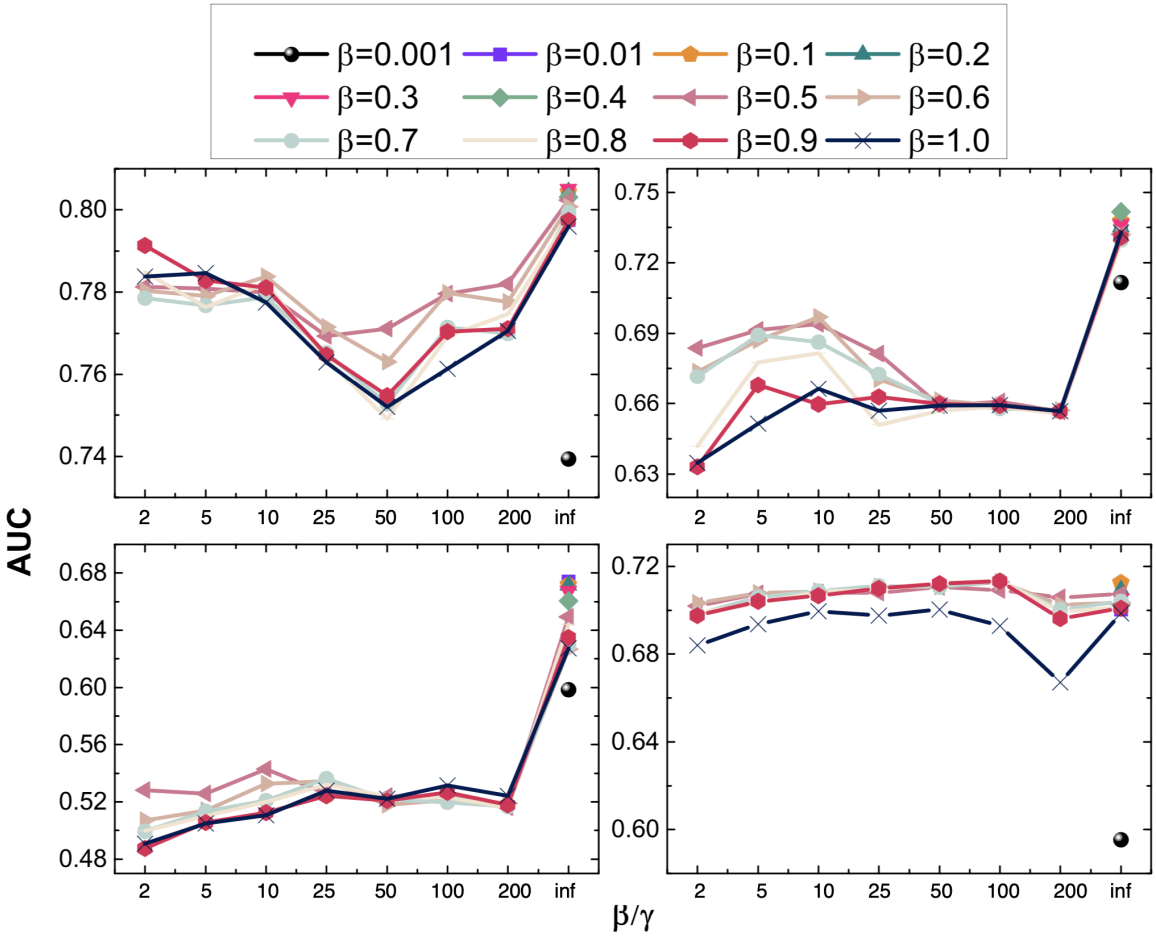


Figure (6.3) The change of AUC with $\frac{\beta}{\gamma}$, different curves correspond to different value of β.

Figure 6.3 shows the performance when choosing different combinations of $\frac{\beta}{\gamma}$ and β. When $\beta < 0.5$ ($\beta = 0.001, 0.01, 0.1, 0.2, 0.3, 0.4$), only SI spreading process (e.g., $\frac{\beta}{\gamma} = \infty$) has been conducted, shown as dots in the right part of figures. When $\beta \geq 0.5$, experiments with grid search over β and $\frac{\beta}{\gamma}$ are conducted. When $\frac{\beta}{\gamma} = \infty$ (e.g., an SI model), the best results are obtained almost in all cases (expect when β is extremely small, such as $\beta = 0.001$). The best AUC value of the first three

datasets (Figure 6.3(a-c)) is given by $\beta = 0.01, 0.4, 0.3$ when we use the SI model respectively. For the dataset fb-forum, the best result is obtained when applying either SIS model ($\beta = 0.7$, $\frac{\beta}{\gamma} = 100$) or SI model($\beta = 0.1$). Compared to the other three datasets, the performance of SIS or SI model stays relatively stable with the change of $\frac{\beta}{\gamma}$ or β in fb-forum. It can be concluded from the above observations that applying SI model on networks is a better option considering its good performance and less parameter consumption. Therefore, we will focus on the analysis of the SI model in the following sections.

6.4. Performance analysis of the SI model

According to Figure 6.3, SI model can trigger better performance compared to the SIS model. The subsequent analysis will mainly focus on SI model. We aim to probe what sampling strategy can yield a better performance and explore what kind of attributes are extracted using different sampling strategies. In addition, we also study the properties of the information generated by the SI spreading process.

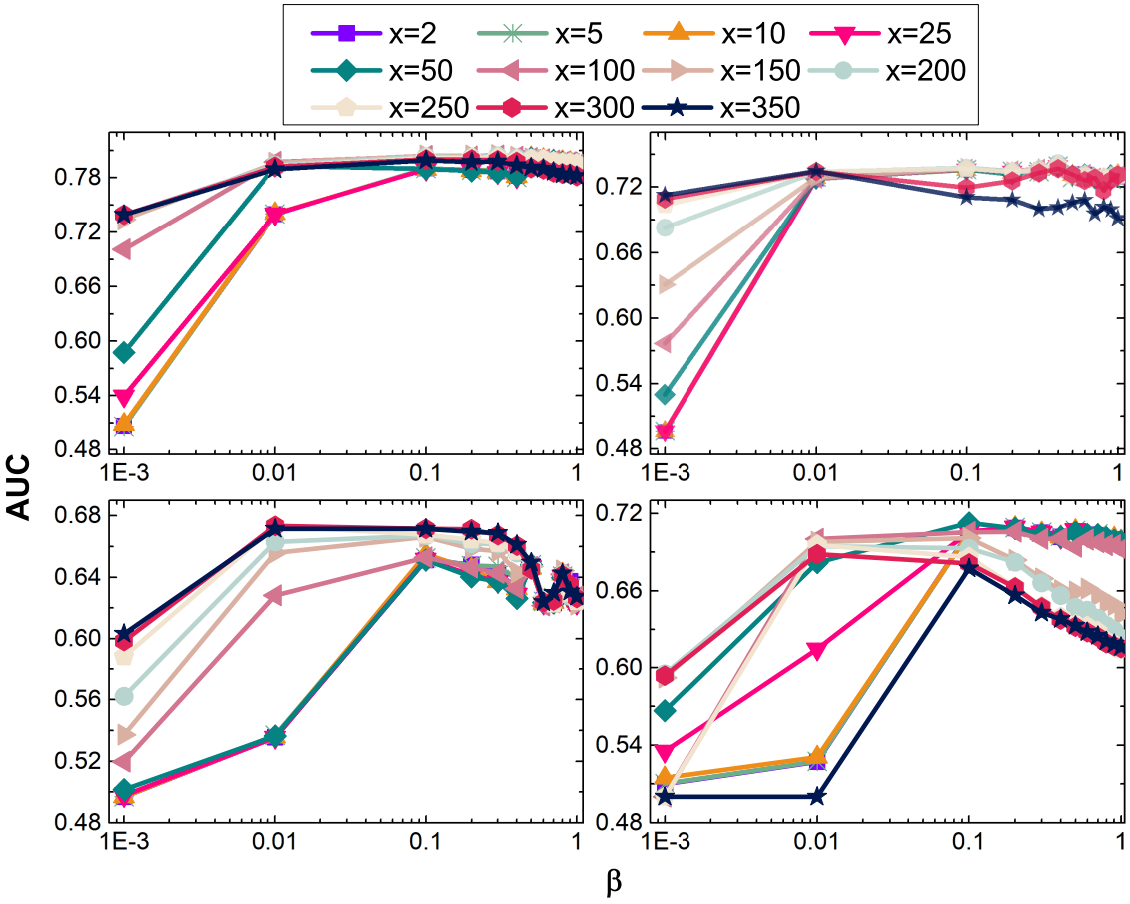


Figure (6.4) The change of AUC with β for different amount of input data X for the SI model.

6.4.1. Parameter analysis

To further investigate the influence of the amount of input data, performances are compared under different combinations between β and X in Figure 6.4. In the figure, each curve represents the performance with the same amount of input, X, with the change of the infection rate β. We observe that more input data cannot necessarily guarantee good performance. The best AUC scores are

obtained or the performance gets stable when β is small ($\beta \leq 0.5$). But there is a big increase of AUC scores when β is in $[0.001, 0.01]$ for datasets haggle and ia-radoslaw-email or in $[0.01, 0.1]$ for the datasets ia-contacts_hypertext2009 and fb-forum. Therefore, we can conclude from this figure that, we can get relatively good performance for all the datasets tested in this work when choosing $\beta \geq 0.1$ and a small value of X.

6.4.2. Spreading Tree Analysis of SI Model

Our method applies a diffusion-based paradigm by applying SIS spreading on networks to simulate trajectories. We construct *spreading tree* $\mathcal{T}$, which represents the spreading of the SIS model starting from each node on the networks[87]. In another word, a $\mathcal{T}_i$ records a spreading tree started from node i via information diffusion. Trajectories can then be extracted from the trees by uniformly sampling nodes along the branches. In this section, we try to analyze the properties of the spreading trees retrieved by setting different β. The analysis of the trajectory can provide a more general understanding of the input data for the learning algorithm considering that the input data we generated is extracted from the trees.

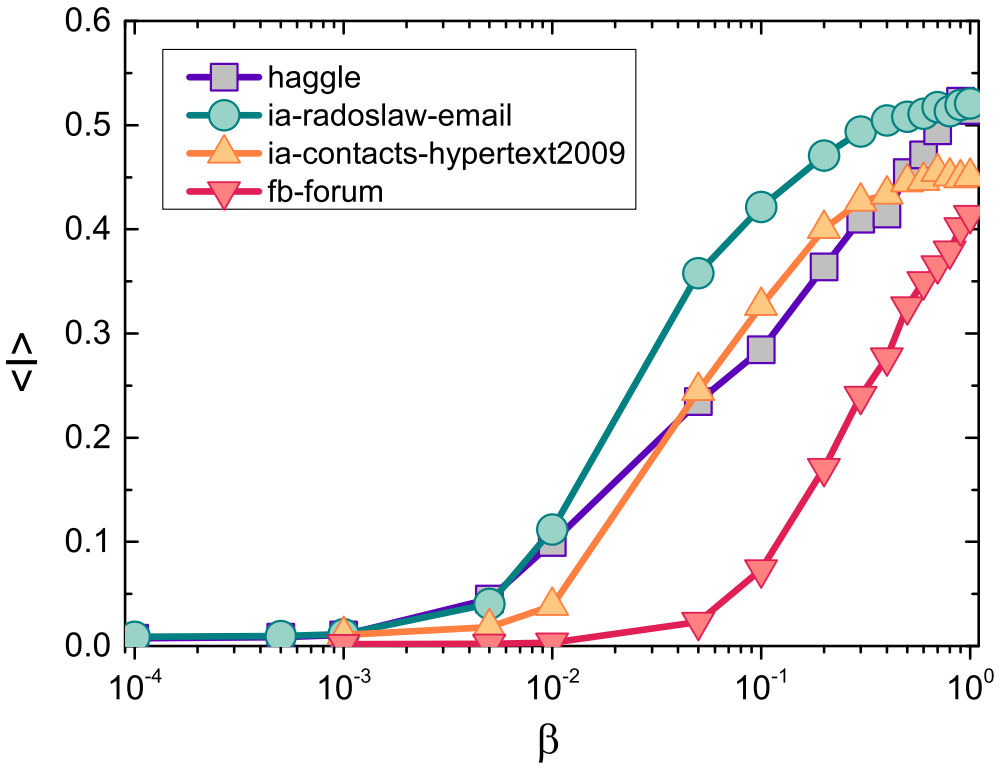
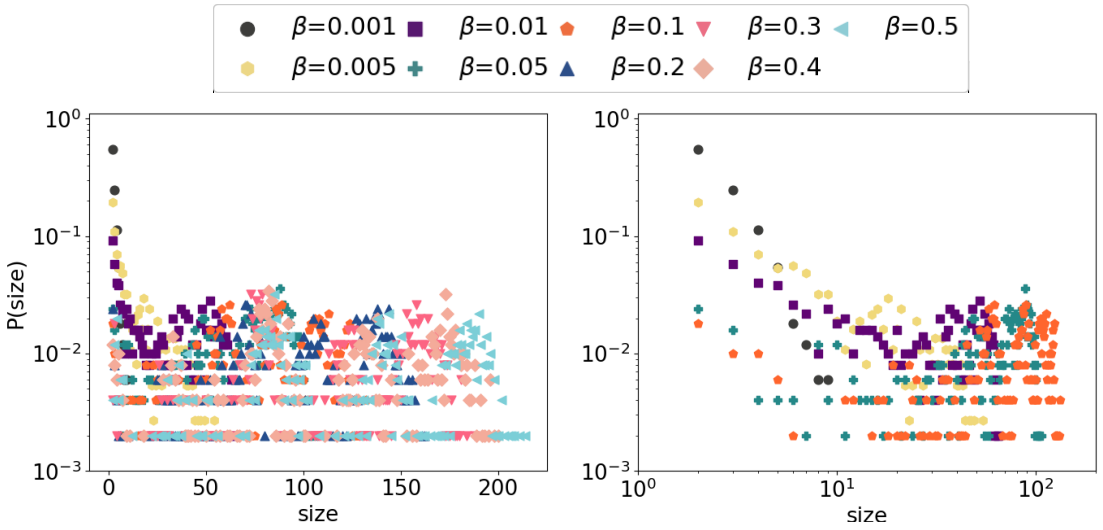
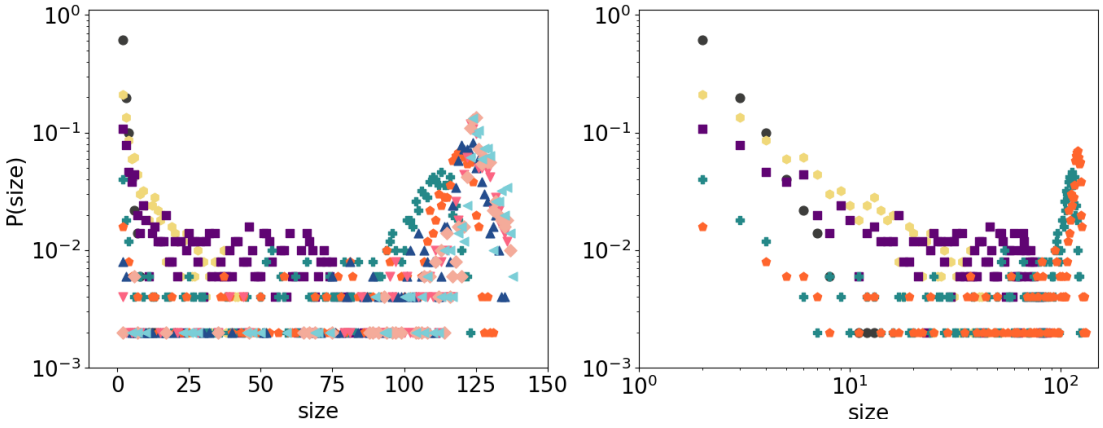


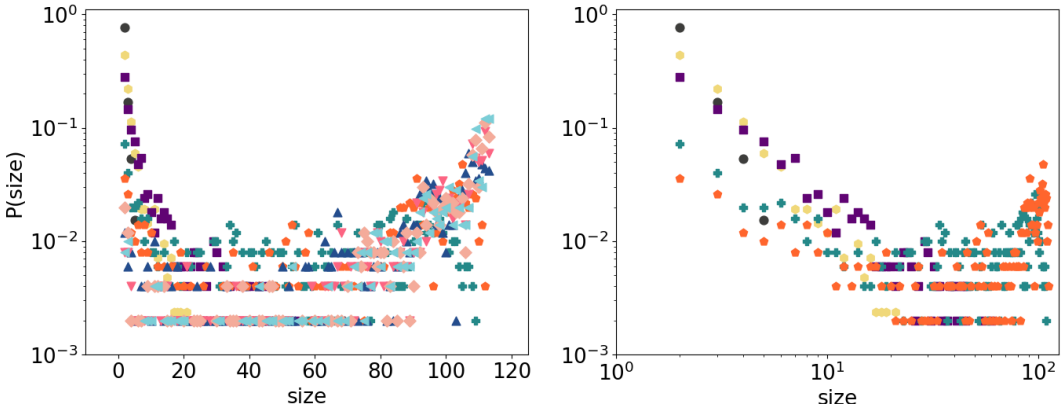
Figure (6.5) Normalized average tree size $< I >$ of the SI spreading process with the change β on temporal networks.



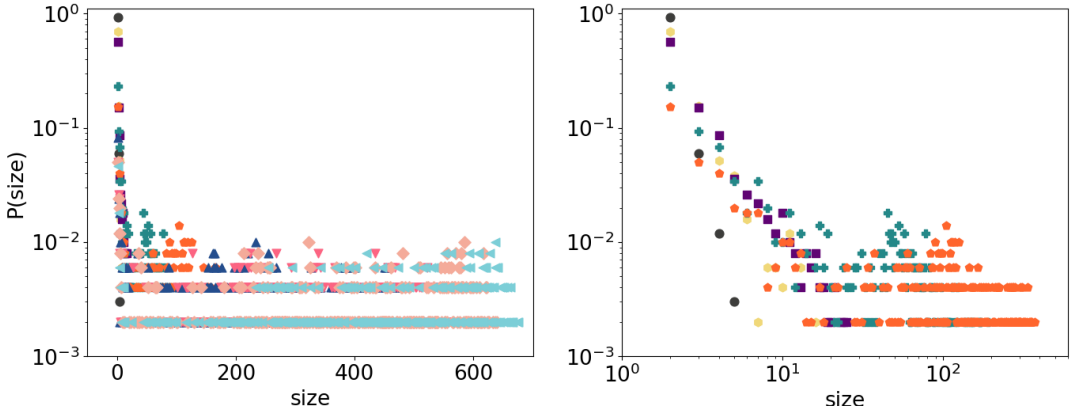
(a) haggle



(b) ia-radoslaw-email



(c) ia-contacts_hypertext2009



(d) fb-forum

Figure (6.6) Spreading tree size distribution with different β for SI model. When $\beta \geq 0.01$, the size follows bimodal distribution.

In Figure 6.5, we plot the normalized average size of the spreading trees (denoted as $< I >$) started from every node in the network. We show that the normalized average tree size $< I >$ is quite small when β is small and we can get a large infected population when $\beta > 0.1$ for all the datasets. This corresponds to the good performance we obtained in Figure 6.4 when $\beta > 0.1$ for all the datasets. Another observation that can be linked to Figure 6.4 is that the big increase of the infection population from almost zero to a non-zero value stays when $\beta \in [0.001, 0.01]$ for datasets haggle and ia-radoslaw-email and $\beta \in [0.01, 0.1]$ for datasets ia-contacts_hypertext2009 and fb-forum. This phenomenon corresponds to the big increase of the performance we claimed in the above section in the corresponding range of β.

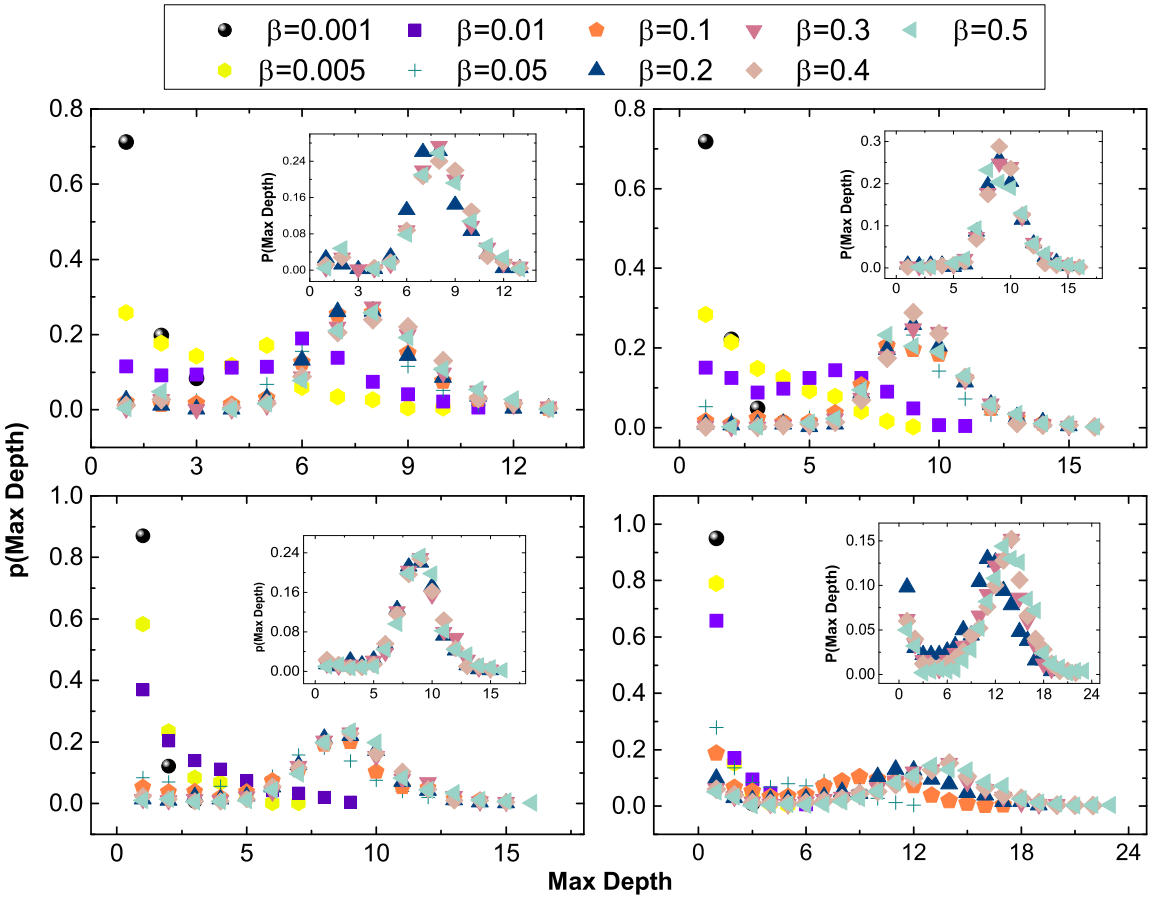


Figure (6.7) Spreading tree depth distribution with different β for SI model. When $\beta \leq 0.01$ in ia-contacts_hypertext2009 and fb-forum, the size follows long-tail distribution.

We further give the tree size distribution in Figure 6.6, with the left panel showing the results of semi-logarithmic plots for $\beta \in [0.001, 0.5]$ (only the ordinate is scaled logarithmically) for β and right panel showing results of log-log plots for small β in the range of [0.001, 0.1]. For small β, the tree size distribution stays only in the left side of figures in the left panel. If we check the log-log plots, we find that the tree size follows long-tail distribution as shown in the figures in the right panel. When $\beta > 0.01$, the tree size follows bimodal distribution with larger tree size compared to small β.

The depth of node i in a tree is defined as the length of the path from the root node to node i. Therefore, the maximum depth of the spreading tree is defined as the longest path from the root node to all the other nodes in the tree. We investigate the maximum depth distribution for different values of β, as it is related to the trajectory length of the trajectory set (also called corpus) we need

for the Skip-gram model. The maximum depth distribution for different β is given in Figure 6.7, with the inset of the figures showing the distribution of large β. The figures show that the maximum depth distribution is more heterogeneous with small β and homogeneous with large β. In addition, the max depth tends to be small with small value of β and vice versa.

We study the spreading tree structure over different β by looking at the the average degree of the nodes in the first three depths of the tree. For example, depth 0 corresponds to the root node, depth 1 corresponds to the nodes that are directly connected to the root node and so on. The results are shown in Figure 6.8. In the four datasets, the average degree is increasing with the depth when $\beta > 0.01$. This implies the spreading trees are growing and further explained why we get larger average tree size with larger β (shown in Figure 6.5). In Figure 6.8(a, b), the average degree starts to increase when $\beta = 0.01$. However, the average degree starts to increase when when $\beta = 0.05$ in Figure 6.8(c, d). The results from Figure 6.5 to 6.8 in total show that the SI spreading threshold value of haggle and ia-radoslaw-email stays approximately in the range of $\beta \in [0.005, 0.01]$. For ia-contacts_hypertext2009 and fb-forum, the threshold value stays in the range of $\beta \in [0.01, 0.05]$. Taking Figure 6.4 into account, we can conclude that the good performance (AUC score) can be obtained when β is larger than the threshold value.

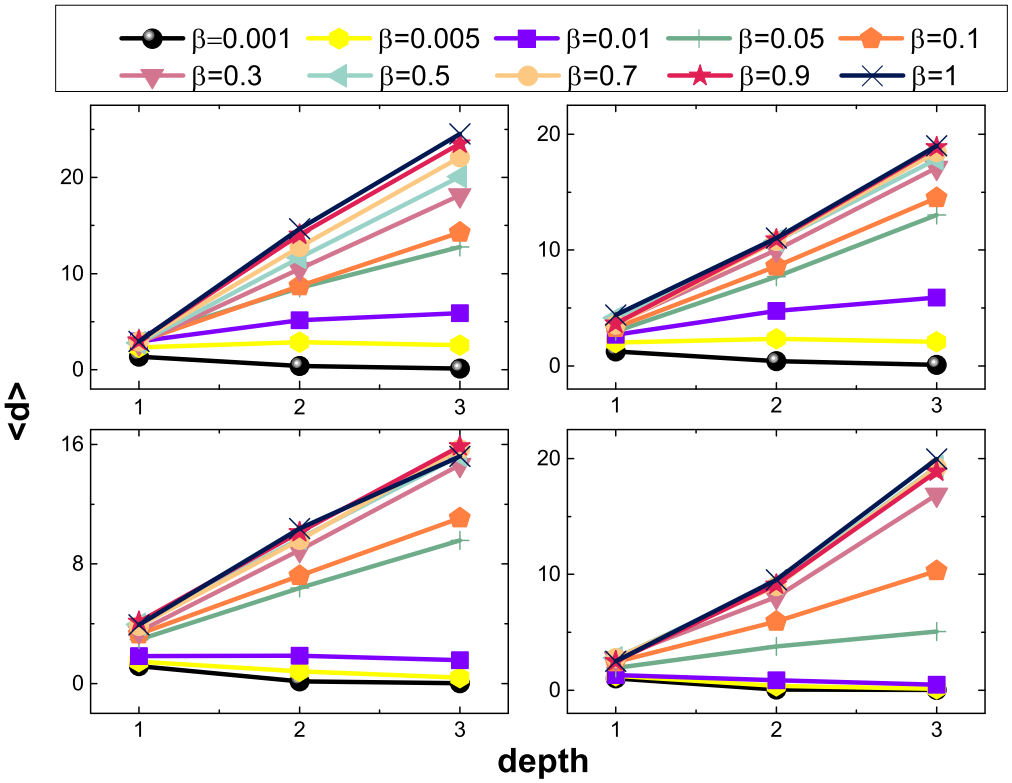


Figure (6.8) Average degree of the first three depths in the spreading trees. Different curves correspond to different value of β.

6.4.3. Summary

We analyze the performance of SI spreading model as well as the properties of the spreading trees. We find out that choosing a higher β tend to infect more nodes in networks, resulting in generating spreading trees with bigger size and larger depth. We observe that the shape of trees is expanding with a larger β while it remains thin when β is smaller than 0.01. Considering the results in link

prediction task, we can conclude that neither too much (large β) or too few information extracted can lead to good performance. A good balance is of significance. We also notice that there is a gap or threshold in performance, which is worthy of further investigation and figure out how it affects the result.

6.5. Properties of Trajectories and G_S

After obtaining the spreading trees, we generate the trajectories from the trees according to Section 3.4.2. Subsequently, G_S can be formed by the node pairs generated from the trajectories, where edges correspond to node pairs. In this section, we will focus on the properties of the extracted trajectories as well as basic properties of network G_S.

6.5.1. Trajectory Length Analysis

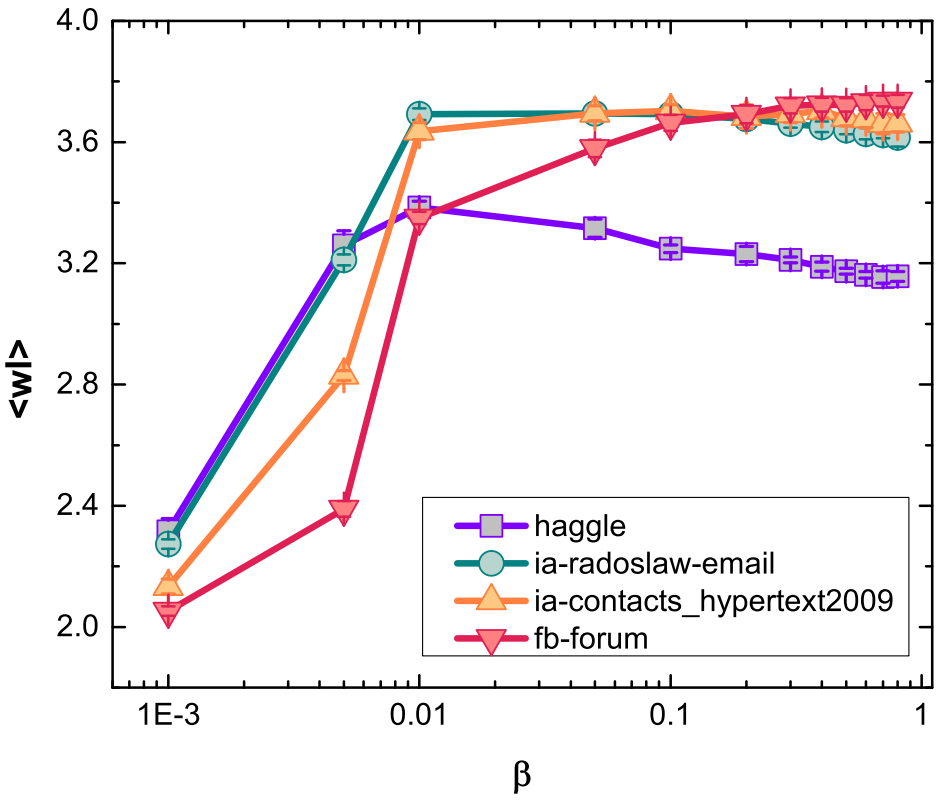


Figure (6.9) Average trajectory length $< wl >$ changes with β for networks: haggle, ia-radoslaw-email, ia-contacts_hypertext2009 and fb-forum.

The length of the trajectories actually can reflect the choice of neighboring nodes of the center node in the node pair generation procedure. For example, for a fix window size ws we choose, if the length of a trajectory is two, we can only generate node pairs with the first-order neighbors. The longer the length, the more hops we can consider as neighbors. We show how the average trajectory length changes with infection probability β in Figure 6.9. The larger β is, the larger the value of average trajectory length. This can be explained by the maximum depth distribution we give in Figure 6.7, which shows that larger β can result in larger maximum depth. In the static network embedding methods, like *Node2Vec* and *DeepWalk*, the trajectory length is set as a default value (i.e., 80). Since it is operated on static integrated network, the trajectory length is fixed in every extracted trajectory. However, the average trajectory length for different β here is less than 4. Taking the per-

formance we get in the previous sections (Table 6.2), we find that for temporal networks, the best performance of link prediction can be obtained by only considering short trajectories. We can conclude that a long trajectory extracted from networks may not necessary lead to good performance.

7 Conclusion

This chapter concludes the thesis work regarding the results and observations. Following up, the future work in this field will also be discussed.

7.1. Observations and Contributions

This section will be organized by answering the research question proposed in the first chapter, Introduction.

1. **How to utilize the diffusion based model and temporal information to simulate trajectories on networks for node embedding?**

   We introduce our framework in details in Chapter 3 Methododology. In short, we initialize the process by starting from a random node at random timestamp. In every time step, we sample newly infected nodes by applying SIS model on a temporal network. Those sampling trajectories follow the ascending time sequence and are later fed into a Skip-gram model.

2. **How is the performance of the proposed model in contrast with the other node embedding algorithms on the link prediction task in temporal networks?**

   Compared to the state-of-art techniques, *SISNE* has achieved outstanding performance, revealed by significant gains across all datasets. It has shown great privileges compared to models operating on static networks, e.g. *DeepWalk* and *Node2vec* (We aggregated the temporal network into static network when conducting the experiments). Specially, one of our contribution is that only a small amount of input is needed to achieve a relatively good result, given the same experimental setting compared to *CTDNE*. Experiments are conducted choosing different combinations of parameters. Based on the link prediction performance, we observe that using SI spreading process on networks can yield a better results. This can be explained by the fact that nodes can only be infected once throughout the whole process, which reduces the noise when preserving the structure of networks.

3. **How to explain the difference in performance of different algorithms and how is the properties of the sampled information related to the performance?**

   In chapter 6, Results and Analysis, performance of *SISNE* under different parameter settings are presented. Results from all steps in the model are gathered and recorded. Based on the node pairs retrieved in the model, we can generate simulated networks. We tend to explore what properties of the generated network can lead to the corresponding results.

   A well-rounded analysis is conducted to explore the intrinsic relationship, e.g., *walk length analysis, link density analysis, edge weight correlation analysis, trajectories analysis* and etc

(Properties analysis is presented in Appendix). We reveal some interesting insights. It is observed that there is a dramatic change in performance when β is small and the trend becomes stable when β is larger, indicating a threshold of β leading to the gap. We also observe that the threshold of β varies in different datasets, where the threshold of β is pair-wise similar, haggle & ia-radoslaw-emmail ($\beta \in [0.001, 0.01]$) and fb-forum & ia-contacts_hypertext2009 ($\beta \in [0.01, 0.1]$). These thresholds in results also accords with the observations when exploring the properties of the generated networks. These thresholds lie in the observed range in link prediction performance across datasets.

In addition, we observe some attributes of the retrieved networks as expected to the rules of different SIS spreading process. For example, a larger β leads to more nodes being infected during the information spreading process, resulting in trajectories with larger size and larger depth. However, a larger trajectories, indicating more information preserved, does not necessarily guarantee a good performance. Similar observation is obtained in *Walk length analysis*, where a larger β in SI spreading process retrieves longer walk length. A longer walk length is not a essential condition for good performance.

7.2. Future Work

We propose a novel node embedding model under the paradigm of random walk by operating SIS spreading process of networks. Results are analyzed and some interesting insights are discovered. In this section, future work will be discussed.

It is mentioned previously that there is a threshold of β in link prediction performance as well as the properties analysis of generated networks. However, these thresholds are not the same in different datasets. In addition, though the thresholds lie in the similar range observed in link prediction result and properties analysis, the threshold values vary in different experiments. Further investigation is necessary on the correlation of the threshold of β between the network properties and the task performance, which is not covered in this work.

Further more, networks with different attributes, e.g., density, domain, scale, can be involved. In this work, we tested on four different datasets and a pair-wise similarity is observed. However, we did not make the analysis on what attributes have led to the similar performance. More comprehensive analysis can be conducted to explore how the attributes of the network can influence the results.

Appendices

A

Properties Analysis of G_S

We give some analysis on the properties of G_S in this appendix, serving as a support to Chapter **6 Results**. Link density analysis, edge weight correlation analysis as well as degree analysis will be covered. We tend to provide a well-rounded observation on the properties of G_S.

A.1. Link Density Analysis

Link density is one of the fundamental network property, which reflects the density of a network. It follows the fomula:

$$\textbf{link density} = \frac{2 * |L_G|}{N * (N-1)} \tag{A.1}$$

We compute the link density of G_{Sample} by choosing different β, shown in Figure A.1. It can be observed that a larger β generate a denser network. All four datasets obtain a denser generated network than the original network. We also observe that when β is larger than 0.1, the generated network tends to have consistent density across datasets. Specially, the link density remains stable in haggle and fb-forum when β varies.

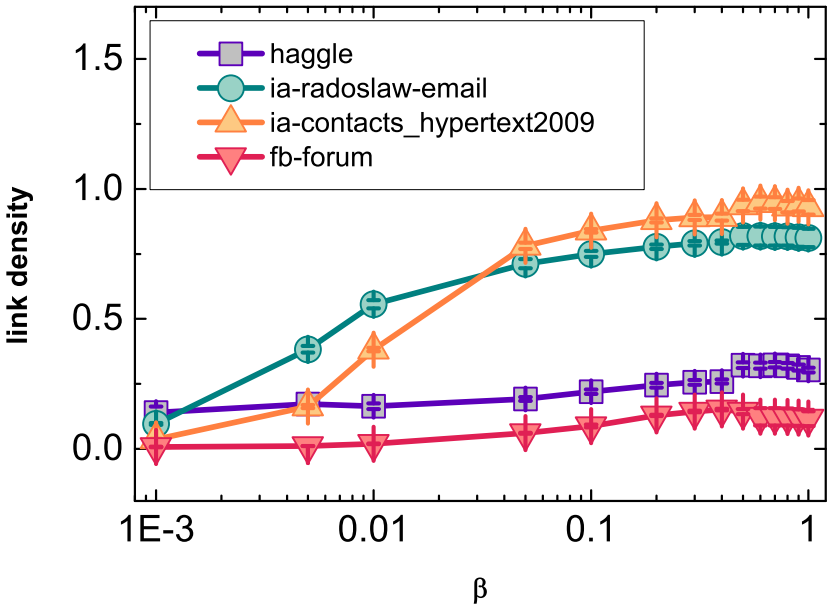


Figure (A.1) Link Density (**D**) vs. β. Link density in the original network: haggle(0.0527 ± 0.0015), ia-radoslaw-email(0.229 ± 0.0062), ia-contacts_hypertext2009(0.2603), fb-forum(0.0094 ± 0.0014

A.2. Edge Weight Correlation

In this subsection, we explore the correlation between the mean edge weight of different G_S and the mean edge weight of the original network. We compute the correlation using Kendall correlation coefficient, denoted as τ[60]. Given $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ as the observations of two joint random variables X and Y. Then, Kendall ranking correlation coefficient $\tau \in [1, 1]$ is computed as:

$$\tau = \frac{1}{n(n-1)} \sum_{i \neq j} \operatorname{sgn}\left(x_i - x_j\right) \operatorname{sgn}\left(y_i - y_j\right) \tag{A.2}$$

We compute the edge weight correlation between G_S and G_O, denoted as τ_w. We surprisingly find out that, as shown in Figure A.2, the peak of each line, denoting a dataset, lies around the threshold of β, which also accords with our observations above. We further conduct other experiments to analysis the properties of the extracted networks G_S.

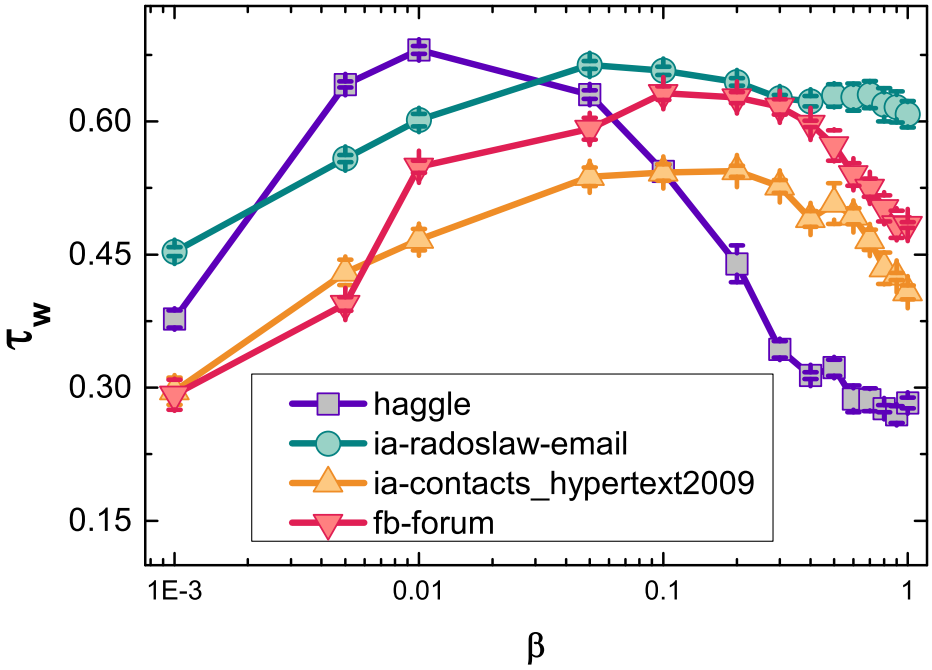


Figure (A.2) Edge Weight Correlation between G_S and G_O.

A.3. Clustering Coefficient Analysis

Clustering Coefficient (CC), concerned with the density of triplets of nodes in a network[54], measures the degree to which nodes in a network tend to cluster together. A higher value indicates a tightly knit groups among nodes. It can be observed from Figure A.3 that the unweighted Clustering Coefficient change dramatically when β is small and becomes steady when β becomes larger. When β is 0.001, all four datasets obtain the lowest value. The results on fb-forum are consistent and remain low given different β. Interestingly, our proposed model generate networks with a higher unweighted CC compared to the original network, indicating the proposed model can well capture the informative part of the network.

A.4. Degree Analysis

In the final analysis, we compute the average degree of nodes. The average degree increases when β is getting larger, which applies to all four datasets. When β is larger than 0.1, the average degree

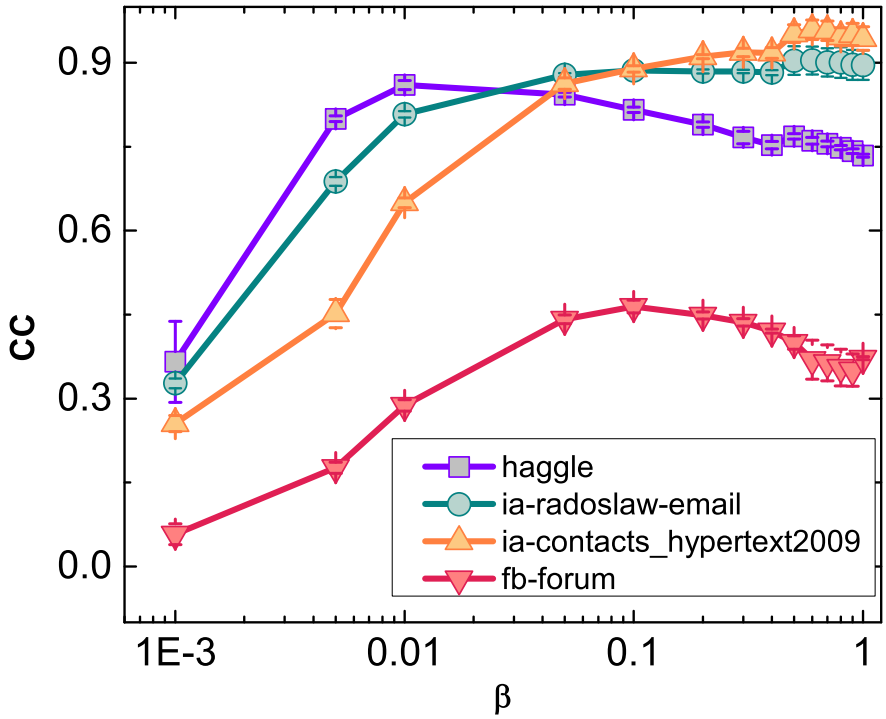


Figure (A.3) CC vs. β. CC in the original network: haggle(0.4501 ± 0.03236), ia-radoslaw-email(0.5041 ± 0.0062), ia-contacts_hypertext2009(0.3971 ± 0.0072), fb-forum(0.0231 ± 0.0021

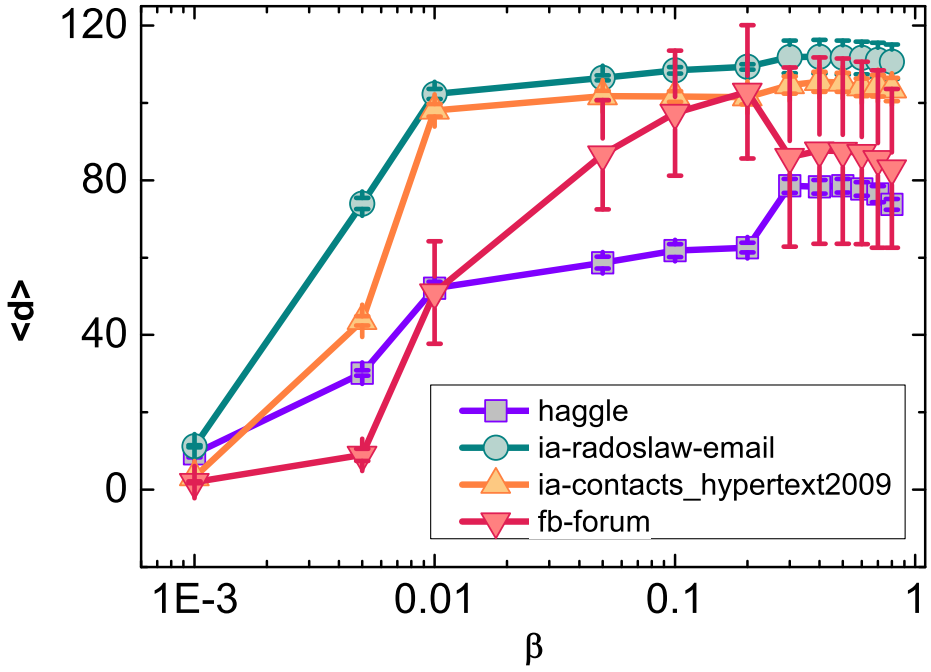


Figure (A.4) Mean degree vs. β. The average degree of the original network is: haggle(12.93 ± 0.1875), ia-radoslaw-email(31.49 ± 0.0734), ia-contacts_hypertext2009(29.15), fb-forum(7.09 ± 0.063)

becomes relatively stable. This show similar trend with the average trajectory length.

A.5. Summary

In this section, we analyze the property of the extracted trajectories from the networks and compute some basic centralities of the generated network based on the trajectories. We observe some similar trends in all these experiments, where the performance tends to be steady when β is getting larger ($\beta \geq 0.1$). This trend is in line with the performance in Figure 6.4, where the results for link prediction task remain relatively stable when $\beta \geq 0.1$, especially in the case of haggle and ia-radoslaw-email.

Bibliography

[1] Roc curve analysis.

[2] Thomas Aynaud and Jean-Loup Guillaume. Multi-step community detection and hierarchical time segmentation in evolving networks. In *Proceedings of the 5th SNA-KDD workshop*, 2011.

[3] Thomas Aynaud, Eric Fleury, Jean-Loup Guillaume, and Qinna Wang. Communities in evolving networks: definitions, detection, and analysis techniques. In *Dynamics On and Of Complex Networks, Volume 2*, pages 159–200. Springer, 2013.

[4] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, pages 585–591, 2002.

[5] Smriti Bhagat, Graham Cormode, and S Muthukrishnan. Node classification in social networks. In *Social network data analytics*, pages 115–148. Springer, 2011.

[6] Ronald S Burt. Structural holes and good ideas. *American journal of sociology*, 110(2):349–399, 2004.

[7] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pages 891–900. ACM, 2015.

[8] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[9] Zhu Cao, Linlin Wang, and Gerard de Melo. Link prediction via subgraph embedding-based convex matrix completion. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[10] Sandro Cavallari, Vincent W Zheng, Hongyun Cai, Kevin Chen-Chuan Chang, and Erik Cambria. Learning community embedding with community detection and node embedding on graphs. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 377–386. ACM, 2017.

[11] Augustin Chaintreau, Pan Hui, Jon Crowcroft, Christophe Diot, Richard Gass, and James Scott. Impact of human mobility on opportunistic forwarding algorithms. *IEEE Transactions on Mobile Computing*, (6):606–620, 2007.

[12] Shiyu Chang, Wei Han, Jiliang Tang, Guo-Jun Qi, Charu C Aggarwal, and Thomas S Huang. Heterogeneous network embedding via deep architectures. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 119–128. ACM, 2015.

[13] Hongxu Chen, Hongzhi Yin, Weiqing Wang, Hao Wang, Quoc Viet Hung Nguyen, and Xue Li. Pme: projected metric embedding on heterogeneous networks for link prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1177–1186. ACM, 2018.

[14] Elior Cohen. node2vec: Embeddings for graph data, 2018. URL https://towardsdatascience.com/node2vec-embeddings-for-graph-data-32a866340fef.

[15] Luciano da Fontoura Costa, Osvaldo N Oliveira Jr, Gonzalo Travieso, Francisco Aparecido Rodrigues, Paulino Ribeiro Villas Boas, Lucas Antiqueira, Matheus Palhares Viana, and Luis Enrique Correa Rocha. Analyzing and modeling real-world phenomena with complex networks: a survey of applications. *Advances in Physics*, 60(3):329–412, 2011.

[16] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. A survey on network embedding. *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[17] Li Deng, Dong Yu, et al. Deep learning: methods and applications. *Foundations and Trends® in Signal Processing*, 7(3–4):197–387, 2014.

[18] Sheng Gao, Ludovic Denoyer, and Patrick Gallinari. Temporal link prediction by integrating content and structure information. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1169–1174. ACM, 2011.

[19] Lise Getoor and Christopher P Diehl. Link mining: a survey. *Acm Sigkdd Explorations Newsletter*, 7(2):3–12, 2005.

[20] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002.

[21] Gene H Golub and Christian Reinsch. Singular value decomposition and least squares solutions. In *Linear Algebra*, pages 134–151. Springer, 1971.

[22] Benjamin H Good, Yves-Alexandre De Montjoye, and Aaron Clauset. Performance of modularity maximization in practical contexts. *Physical Review E*, 81(4):046106, 2010.

[23] Aditya Grover and Jure 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.

[24] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

[25] Renjun Hu, Charu C Aggarwal, Shuai Ma, and Jinpeng Huai. An embedding approach to anomaly detection. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 385–396. IEEE, 2016.

[26] Xiao Huang, Jundong Li, and Xia Hu. Accelerated attributed network embedding. In *Proceedings of the 2017 SIAM international conference on data mining*, pages 633–641. SIAM, 2017.

[27] Xiao Huang, Jundong Li, and Xia Hu. Label informed attributed network embedding. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 731–739. ACM, 2017.

[28] Yann Jacob, Ludovic Denoyer, and Patrick Gallinari. Learning latent representations of nodes for classifying in heterogeneous social networks. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 373–382. ACM, 2014.

[29] Brian W Kernighan and Shen Lin. An efficient heuristic procedure for partitioning graphs. *Bell system technical journal*, 49(2):291–307, 1970.

[30] Young Kim and Jaideep Srivastava. Impact of social influence in e-commerce decision making. In *Proceedings of the ninth international conference on Electronic commerce*, pages 293–302. ACM, 2007.

[31] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[32] KONECT. Haggle network dataset. http://konect.uni-koblenz.de/networks/contact. April 2017.

[33] Anna CF Lewis, Nick S Jones, Mason A Porter, and Charlotte M Deane. The function of communities in protein interaction networks at multiple scales. *BMC systems biology*, 4(1):100, 2010.

[34] Jundong Li, Harsh Dani, Xia Hu, Jiliang Tang, Yi Chang, and Huan Liu. Attributed network embedding for learning in a dynamic environment. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 387–396. ACM, 2017.

[35] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018.

[36] Lizi Liao, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. Attributed social network embedding. *IEEE Transactions on Knowledge and Data Engineering*, 30(12):2257–2270, 2018.

[37] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

[38] Lei Lin, Qian Wang, and Adel W Sadek. Data mining and complex network algorithms for traffic accident analysis. *Transportation Research Record*, 2460(1):128–136, 2014.

[39] Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications*, 390(6):1150–1170, 2011.

[40] Yuping Lu. Recommender systems. In *Physics Reports*. Citeseer, 2012.

[41] Guixiang Ma, Lifang He, Chun-Ta Lu, Weixiang Shao, Philip S Yu, Alex D Leow, and Ann B Ragin. Multi-view clustering with graph embedding for connectome analysis. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 127–136. ACM, 2017.

[42] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[43] Yun Mao and Lawrence K Saul. Modeling distances in large-scale networks by matrix factorization. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 278–287. ACM, 2004.

[44] Víctor Martínez, Carlos Cano, and Armando Blanco. Prophnet: a generic prioritization method through propagation of information. *BMC bioinformatics*, 15(1):S5, 2014.

[45] Víctor Martínez, Fernando Berzal, and Juan-Carlos Cubero. A survey of link prediction in complex networks. *ACM Computing Surveys (CSUR)*, 49(4):69, 2017.

[46] Radosław Michalski, Sebastian Palus, and Przemysław Kazienko. Matching organizational structure and social network extracted from email communication. In *International Conference on Business Information Systems*, pages 197–206. Springer, 2011.

[47] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[48] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[49] Sharad Nandanwar and M Narasimha Murty. Structural neighborhood based classification of nodes in a network. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1085–1094. ACM, 2016.

[50] Mark EJ Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003.

[51] Giang H Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. Dynamic network embeddings: From random walks to temporal random walks. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 1085–1092. IEEE, 2018.

[52] Giang Hoang Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. Continuous-time dynamic network embeddings. In *Companion of the The Web Conference 2018 on The Web Conference 2018*, pages 969–976. International World Wide Web Conferences Steering Committee, 2018.

[53] Tore Opsahl. Triadic closure in two-mode networks: Redefining the global and local clustering coefficients. *Social Networks*, 35(2):159–167, 2013.

[54] Tore Opsahl and Pietro Panzarasa. Clustering in weighted networks. *Social networks*, 31(2): 155–163, 2009.

[55] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1105–1114. ACM, 2016.

[56] Shirui Pan, Jia Wu, Xingquan Zhu, Chengqi Zhang, and Yang Wang. Tri-party deep network representation. *Network*, 11(9):12, 2016.

[57] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[58] Yanjun Qi, Ziv Bar-Joseph, and Judith Klein-Seetharaman. Evaluation of different biological data and computational classification methods for use in protein interaction prediction. *Proteins: Structure, Function, and Bioinformatics*, 63(3):490–500, 2006.

[59] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 459–467. ACM, 2018.

[60] Cunquan Qu, Xiuxiu Zhan, Guanghui Wang, Jianliang Wu, and Zi-ke Zhang. Temporal information gathering process for node ranking in time-varying networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(3):033116, 2019.

[61] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical review E*, 76(3):036106, 2007.

[62] Giulio Rossetti and Rémy Cazabet. Community discovery in dynamic networks: a survey. *ACM Computing Surveys (CSUR)*, 51(2):35, 2018.

[63] Giulio Rossetti, Luca Pappalardo, Dino Pedreschi, and Fosca Giannotti. Tiles: an online algorithm for community discovery in dynamic social networks. *Machine Learning*, 106(8):1213–1241, 2017.

[64] Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015. URL http://networkrepository.com.

[65] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

[66] Hooman Peiro Sajjad, Andrew Docherty, and Yuriy Tyshetskiy. Efficient representation learning using random walks for dynamic graphs. *arXiv preprint arXiv:1901.01346*, 2019.

[67] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[68] Uriel Singer, Ido Guy, and Kira Radinsky. Node embedding over temporal graphs. *arXiv preprint arXiv:1903.08889*, 2019.

[69] SocioPatterns. Infectious contact networks. http://www.sociopatterns.org/datasets/. Accessed 09/12/12.

[70] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077. International World Wide Web Conferences Steering Committee, 2015.

[71] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 990–998. ACM, 2008.

[72] Lei Tang and Huan Liu. Relational learning via latent social dimensions. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 817–826. ACM, 2009.

[73] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

[74] Amanda L Traud, Peter J Mucha, and Mason A Porter. Social structure of facebook networks. *Physica A: Statistical Mechanics and its Applications*, 391(16):4165–4180, 2012.

[75] Kerem Turgutlu. Tensor decomposition: Fast cnn in your pocket, 2018. URL https://medium.com/@keremturgutlu/tensor-decomposition-fast-cnn-in-your-pocket-f03e9b2a6788.

[76] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[77] Cuijuan Wang, Wenzhong Tang, Bo Sun, Jing Fang, and Yanyang Wang. Review on community detection algorithms in social networks. In *2015 IEEE International Conference on Progress in Informatics and Computing (PIC)*, pages 551–555. IEEE, 2015.

[78] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234. ACM, 2016.

[79] Hongwei Wang, Fuzheng Zhang, Min Hou, Xing Xie, Minyi Guo, and Qi 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.

[80] Suhang Wang, Jiliang Tang, Charu Aggarwal, Yi Chang, and Huan Liu. Signed network embedding in social media. In *Proceedings of the 2017 SIAM international conference on data mining*, pages 327–335. SIAM, 2017.

[81] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. Community preserving network embedding. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[82] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019.

[83] Linchuan Xu, Xiaokai Wei, Jiannong Cao, and Philip S Yu. Embedding of embedding (eoe): Joint embedding for coupled heterogeneous networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 741–749. ACM, 2017.

[84] Shuicheng Yan, Dong Xu, Benyu Zhang, Hong-Jiang Zhang, Qiang Yang, and Stephen Lin. Graph embedding and extensions: A general framework for dimensionality reduction. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (1):40–51, 2007.

[85] Jiaxuan You, Rex Ying, Xiang Ren, William L Hamilton, and Jure Leskovec. Graphrnn: A deep generative model for graphs. *arXiv preprint arXiv:1802.08773*, 2018.

[86] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875*, 2017.

[87] Xiu-Xiu Zhan, Alan Hanjalic, and Huijuan Wang. Information diffusion backbones in temporal networks. *Scientific reports*, 9(1):6798, 2019.

[88] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. Network representation learning: A survey. *IEEE transactions on Big Data*, 2018.

[89] Yunyi Zhang, Zhan Shi, Dan Feng, and Xiu-Xiu Zhan. Degree-biased random walk for large-scale network embedding. *Future Generation Computer Systems*, 100:198–209, 2019.

[90] Zi-Ke Zhang, Chuang Liu, Xiu-Xiu Zhan, Xin Lu, Chu-Xu Zhang, and Yi-Cheng Zhang. Dynamics of information diffusion and its applications on complex networks. *Physics Reports*, 651: 1–34, 2016.

[91] Lekui Zhou, Yang Yang, Xiang Ren, Fei Wu, and Yueting Zhuang. Dynamic network embedding by modeling triadic closure process. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.