Graph Embedding aided Relationship Prediction in Heterogeneous Networks

CS 512 Project Report

Vipul Venkataraman
vvnktrm2@illinois.edu

Pramod Srinivasan
psrnvsn2@illinois.edu

ABSTRACT

We consider the problem of predicting relationships in large-scale heterogeneous networks. For example, one can try to predict if a researcher will publish at a conference (eg: VLDB) given her previous publications, or try to anticipate if two reputed researchers working in the same area will collaborate. The main challenge is to extract latent information from such real-world networks which are typically sparse. However, existing relationship prediction techniques do not take into account the sparsity of real-world networks.

Our contributions in this project are two-fold: 1) we develop methods to combat network sparsity by using node similarities obtained from state-of-the-art embedding techniques, and 2) we describe how to enrich the original sparse network with latent relationships to further improve the quality of embedding, and consequently the prediction performance of our method. By evaluating under different scenarios, we demonstrate that in practice, our methods are highly effective in predicting relationships in sparse networks, and improve the accuracy by up to 47% over existing techniques.

1. INTRODUCTION

The advent of social media has given rise to the creation of large, online social networks, such as the Facebook friend network, and the Twitter followers network. With the rapid development of the internet, communication between people have become more convenient. These networks are rich in information of various kinds, and are hence studied with great importance in recent years. Online social networks fall into one of the two categories:

1. Homogeneous information networks: These networks correspond to networks involving one type of nodes and links between them. A typical example is the Facebook friendship network.

2. Heterogeneous information networks: These networks contain different types of nodes and links, and contain more rich information in general [1].

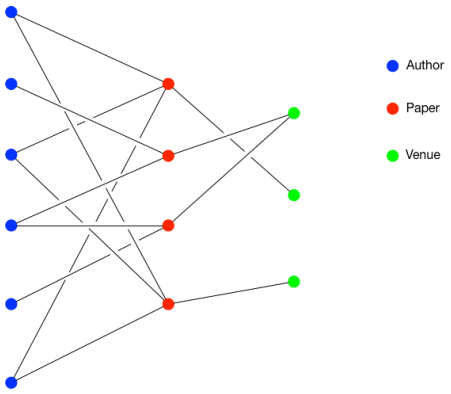

Figure 1: A toy instance of the DBLP network

For both the aforementioned categories, information entities are represented as *nodes* and relationships among the nodes are denoted as *links*. For instance, in the Facebook friendship network, nodes correspond to people and links correspond to the friendship between two people. Hence, a social network can be visualized as a graph.

Mining real world information network is non-trivial due to the following two challenges: incompletion and dynamism. In many cases, all the links in a static information network may not be observable. This can be because 1) these links are hidden to protect one's privacy or 2) these links are absent due to bugs while crawling, coding the network. Under such scenarios, link prediction becomes an important task. Link prediction becomes vital in dynamic networks as well, since these networks can evolve with time. Many links that are nonexistent in the network may appear in future [2, 3]. Therefore, predicting the missing links in social networks or future links in a dynamic network is an important problem.

Link prediction has applications in both the physics and computer science communities. For instance, link prediction can be used to: identify spurious interactions, analyze evolution of networks, extract unavailable information and is hence used extensively in bio-informatics, information retrieval and e-commerce. Link prediction also has ap-

plications in real-world social networks, e.g., recommending friends in a social network can be modeled as a link prediction task, holiday recommendation services (like Airbnb) can be modeled as a location link prediction task in the corresponding heterogeneous network.

Considerable efforts have been put to solve the link prediction problem by computer scientists, economists and physicists. Since networks are of fundamentally different kinds as mentioned above, the link prediction problem can be divided into two categories 1) Link prediction in homogeneous information networks [2] and 2) Link prediction in heterogeneous information networks [4]. For the many link prediction problems, different link prediction techniques have also been proposed: unsupervised and supervised link prediction methods, prediction methods based on random walk, meta-path based link prediction methods and methods based on matrix factorization.

Recent work on heterogeneous networks has focused primarily on latent meta-path based semantics for modeling node interactions. For instance, PathPredict, a meta path-based relationship prediction model, works on the problem of link prediction in the heterogeneous networks by extracting meta path based topological features from a network and using a supervised model to learn the best weights associated with different topological features in deciding future links.

1.1 Rest of the report

The rest of the project is organized as follows: in Section 2, we summarize the related work of the link prediction problem. We formally define the problem of various standard link prediction metrics in Section 3. In Section 4, we discuss the state-of-the-art link prediction methodologies developed for heterogeneous networks. We formally introduce the proposed methodology and the various techniques in Section 4. In Section 5, we compare and contrast the effectiveness of the proposed approaches through the experiments in the DBLP dataset. We conclude our survey in Section **??**.

2. RELATED WORK

Link prediction has been studied on various kinds of graphs including metabolic pathways, protein-protein interaction, social networks, etc. These studies leverage different measures such as node-wise similarity and topology based similarity to predict the existence of the links. In addition to these existing measures, different models have been investigated for the link prediction tasks including relational Bayesian networks and relational Markov networks.

Our work is related to classical methods of Link Prediction in Social Networks. These approaches typically first devise similarity metrics. The intuition is that more similar two nodes are in a network, the likelihood of a link between them increases, and vice versa. This can also be observed in real world networks: users who are more similar with respect to their interests, preferences, and mutual friends are more likely to be friends themselves.

There are several simple and intuitive similar metrics that are used to compute/predict similarities and the chances of a link being formed between a pair of nodes. These metrics are often used as features in learning based methods. In this section, we summarize standard link prediction metrics and methods.

2.1 Link Prediction

In this section, we briefly summarize the various link prediction approaches that are used to compute node similarities and compute the likelihood of a link being formed between a pair of nodes. Such similarity metrics are often used as features in learning based methods.

2.2 Metrics based on Random Walk

Social interactions between nodes in social networks can be modeled by random walk [5], which uses transition probabilities from a node to its neighbors to denote the destination of a random walker from current node. Figure 2 shows how link prediction is performed using supervised random walks. There exists some link prediction measures which calculate similarities between nodes based on random walk such as hitting time and commute time. The hitting time(HT) between two nodes n_1 and n_2 counts the average number of steps required to reach node n_2 from node n_1. Commute Time (CT) is an extension from HT, and corresponds to the number of steps in expectation, required to reach the second node from the first, and again back to the first node. We formally define these two measures before the analysis.

Hitting Time (HT): The hitting time between two nodes n_1 and n_2 counts the average number of steps required to reach node n_2 from node n_1. Therefore, this is an assymetric measure. In other words, $HT(n_1, n_2) \neq HT(n_2, n_1)$. Therefore, we have:

$$\mathrm{HT}(n_1, n_2) = 1 + \sum_{w \in \Gamma(n_1)} P_{n_1,w} HT(w, n_2)$$

Commute Time (CT): Commute time is an extension from HT, and corresponds to the number of steps in expectation, required to reach the second node from the first, and again back to the first node. Therefore,

$$\mathrm{CT}(n_1, n_2) = \mathrm{HT}(n_1, n_2) + \mathrm{HT}(n_2, n_1)$$

Cosine Similarity: Consider $L = D_A - A$. Now let L^* represent the pseudo-inverse of L. Consider a node n_1. Let $x_{n_1} = (L^*)^{\frac{1}{2}} e_{n_1}$. Now the cosine similarity between two nodes n_1 and n_2 is defined to be:

$$\mathrm{CS}(n_1, n_2) = \frac{x_{n_1}^T x_{n_2}}{\sqrt{(x_{n_1}^T x_{n_2})(x_{n_2}^T x_{n_1})}}$$

Random Walk with Restart (RWR): RWR [6] is based on Random Walk, where the worker can restart with a certain probability c.

$$x^{(\tau+1)}(i) = cP^T x^{(\tau)} + (1-c)e_i$$

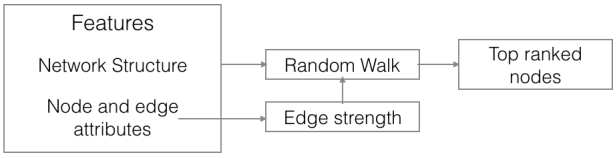

Figure 2: Link prediction based on random walks

We iterate until convergence.

One difficulty with random-walk-based approaches is their sensitive dependence to parts of the network far away from target nodes. Another demerit is that they do not provide information about the structure of the graph, and their use is discouraged in many machine learning applications.

2.3 Metrics based on Matrix Factorization

Menon et all [7] treated the link prediction problem as a matrix completion problem. Matrix completion has been studied extensively in linear algebra, and hence it makes sense to extend this problem to solve the link prediction problem. Their model combines explicit features for nodes and links with the latent features in the graph using a bilinear regression model. This model also takes care of the skewed imbalance ratio in a typical social network. The number of positive links is much lesser than the number of negative links. The latent features obtained via this model can be combined with the results of any other link prediction model.

The model combines explicit features for nodes and links with the latent features in the graph using a bilinear regression model. This model also takes care of the skewed imbalance ratio in a typical social network. Such collaborative filtering approaches suffer from graph sparsity resulting in unreliable and/or unavailable similar information.

2.4 Knowledge Embedding

Vectorized data representations frequently arise in many data mining applications. Knowledge embedding algorithms attempt to learn continuous low-dimensional feature representations of a graph the aim of preserving the inherent structure of the original network.

Among these models, the most representative ones are DeepWalk and PTE. DeepWalk[8] conducts a truncated random walk to obtain vector sequences thus limiting itself to local network information. By employing Skip-Gram, an efficient word representation learning model, the ensemble of such sequences are treated as word sentences. One of the main limitations of DeepWalk is that it does not leverage the first-order proximities among the nodes of the graph. Predictive Text Embedding(PTE) [9] addresses the problem by leveraging both *first-order* and *second-order* proximities.

While DeepWalk uses a random walk to expand the neighborhood of a vertex, which is similar to depth-first search, PTE uses a breadth-first strategy, a more reasonable approach to second-order proximity. The optimization algorithm in PTE is based on edge-sampling, which is very efficient, making the PTE able to scale to networks with millions of nodes and billions of edges. Experiments on both directed, undirected, weighted and unweighted networks have proved the effectiveness and efficiency of PTE.

The second-order information is captured in underlying bipartite graphs in order to train the network embeddings. As explained earlier, the notion is very similar to DeepWalk [8] whose underlying principle of expanding a node's neighborhood is a random walk is analogous to depth-first search. DeepWalk is empirically effective, it does not articulate the network properties which are preserved. DeepWalk is applicable to unweighted graphs is a false claim. In contrast, LINE uses a breadth-first search approach resulting in a carefully modeled objective function to preserve the *second-order* proximity.

The state-of-the-art network embedding models have demonstrated to preserve both local and global network structures, thus making it suitable for arbitrary types of information networks: undirected or directed, binary or weighted. Experimental results on various real-world networks prove the efficiency and effectiveness of PTE.

3. PROBLEM DEFINITION

In this section, we formally define the problem of graph embedding aided relationship prediction in heterogeneous networks. We first formally define the information network as follows:

3.1 Information Network

An *information network* is defined as $G = (V, E)$, where V is the set of vertices, each representing a data object and E is the set of edges between the vertices, each representing a relationship between two data objects. Each edge $e \in E$ is an ordered pair $e = (u, v)$ and is associated with a weight $w_{uv} > 0$, which indicates the strength of the relation.with an object type mapping function $\tau : V \rightarrow A$ and a link type mapping function $\varphi : E \rightarrow R$, where each object $v \in V$ belongs to one particular object type $\tau(v) \in A$, each link $e \in E$ belongs to a particular relation $\varphi(e) \in R$, and if two links belong to the same relation type, the two links share the same starting object type as well as the ending object type.

In contrast to the traditional network definition, when the types of objects $|A| > 1$ or the types of relations $|R| > 1$, the network is called **heterogeneous** information network; otherwise, it is a **homogeneous** information network.

3.2 Network Schema

Given a complex heterogeneous information network, it is necessary to provide its meta level(i.e., schema-level) description for better understanding the object types and link types in the network. Therefore, we propose the concept of network schema to describe the meta structure of a network as follows: A *network schema* is denoted as $T_G = (A, R)$, is a meta template for a heterogeneous network $G = (V, E)$ with the object type mapping $\tau : V \rightarrow A$ and the link mapping $\varphi : E \rightarrow R$, which is a directed graph defined over object types A, with edges as relations from R.

The network schema of a heterogeneous information network

has specified type constraints on the sets of objects and relationships between the objects. These constraints make a heterogeneous information network semi-structured, guiding the exploration of the semantics of the network.

Real-world information networks can be either directed (e.g., citation networks) or undirected (e.g., social network of users in Facebook). The weights of the edges can be either binary or can take any real value. Note that while negative edge weights are possible, in this study we only consider non-negative weights. If G is undirected, we have $(u, v) \equiv (v, u)$ and $w_{uv} \equiv w_{vu}$; if G is directed, we have $(u, v) \not\equiv (v, u)$ and $w_{uv} \not\equiv w_{vu}$.

3.3 Network Embedding

Embedding an information network into a low-dimensional space is useful in a variety of applications. To conduct the embedding, the network structures must be preserved. The first intuition is that the local network structure, i.e., the local pairwise(*first-order* proximity between the vertices, must be preserved.

However real world information network are *sparse*, the links observed are only a small proportion. A pair of nodes on a missing link has a zero *first-order* proximity, even though they are intrinsically very similar to each other. Therefore, *first-order* proximity alone is not sufficient for preserving the network structures, and it is primordial to seek an alternative notion of proximity that addresses the problem of sparsity.

A natural intuition is that vertices that share similar neighbors tend to be similar to each other. To generalize, given a pair of vertices, if their neighborhood network structures are similar, they are also likely to be similar to each other. We therefore define the *second-order* proximity, which complements the first-order proximity and preserves the network structure.

In the next section, we introduce a large-scale network embedding model that preserves both *first-* and *second-order* proximities.

Let $G = (V, E)$ be the given network, where $V = \bigcup_i V_i$ is the collection of all kinds of vertices in the network and $E = \bigcup_j E_j$ is the collection of all kinds of edges in the network. Note that in homogeneous networks, there exists just one type of vertex and one type of edge in the network. From the given network G, we can extract the set of existing links E. The goal is to predict the set of potential links E' that would occur in future.

Traditional methods in link prediction view this problem as a label prediction problem. Existing links are given positive labels and non-existing links are provided negative labels. This can also be viewed as probability estimates, where higher values represent higher likelihood of the existence of a link and vice-versa for lower values.

4. METHODOLOGY

In this Section, we introduce our proposed methods and techniques for efficient relationship prediction. We first introduce PathPredict [3], which forms the backbone of our methods in Section 4.1. We then introduce the LINE model [10], which is the off-the-shelf embedding technique we chose to use in our project in Section 4.2. Note that our framework allows us to use any embedding technique. We then describe how we extract more informative topological features in Section 4.3, and finally describe the learning algorithm we used to evaluate our predictions in Section 4.4. The workflow of our approach is summarized in Figure 3.

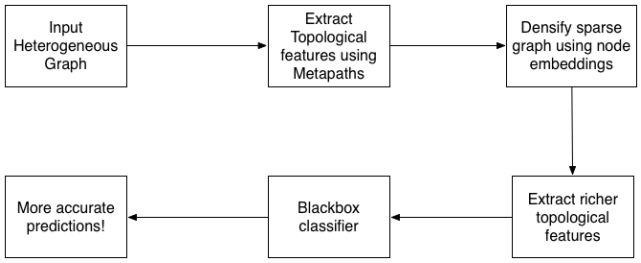

Figure 3: Our overall framework

4.1 The PathPredict framework

PathPredict [3] forms the backbone of our project, and our main contribution is to extend PathPredict to support richer metapaths. The PathPredict paper considered the task of co-authorship relationship prediction. However, we consider the task of predict if an author will publish a paper at a given venue. PathPredict can be viewed as an extension of PathSim [11].

4.1.1 Metapath

We now formally define a metapath as any set of edge connected vertices on the network schema of any arbitrary length, though for computational feasibility, the lengths are usually constructed to be in the range. A metapath usually contains some semantic meaning, which is crucial to the interpretability of its effectiveness.

4.1.2 Topological Features of Interest

In our project, we use the following three measures defined as topological features in PathPredict:

Path Count is written as $PC_R(a_i, a_j)$ where R is some metapath relation such as A — P — V — P — A. This measure simply counts the number of metapath instances starting at a_i and ending at a_j . This means all PC measures are integers.

Random Walk represents the probability of taking a given metapath from a_i to reach the destination node a_j . This is formalized as:

$$RW_R(a_i, a_j) = \frac{PC_R(a_i, a_j)}{\sum_{k=0}^{n} PC_R(a_i, a_k)}$$

Symmetric Random Walk considers both directions of a random walk and is defined as:

$$SRW_R(a_i, a_j) = RW_R(a_i, a_j) + RW_{R^{-1}}(a_j, a_i)$$

4.1.3 Network Sparsity

We now describe the issues with the PathPredict framework. PathPredict does not take into account the sparsity of real world information networks. Real world networks are often sparse, and this is the case with the DBLP network schema as well. A toy instance of the DBLP schema is provided in Figure 1 Hence, the key intuition behind our framework is the following:

Since real-world heterogeneous networks are typically sparse, we perform densification of the original network by adding links that capture similarity. The densified network will enable the extraction of richer and more informative topological features.

The features we eventually extracted from the DBLP schema after performing the densification is discussed in detail in Section 4.3. We now introduce the LINE model, which the off-the-shelf embedding technique we chose to use in our project. Note that our framework allows us to use any embedding technique.

4.2 Network Densification using LINE

LINE [10] performs embedding on large scale information networks. It projects every vertex in the network into a d-dimensional real space while capturing similarities in the network. More formally, graph embedding is a mapping from the set of nodes in the network to a vector space. The goal is to use a low-dimensional vector to represent a vertex in the graph, while preserving both local and global structures known as the *first-order* proximity and the *second-order proximity*. The *second-order* proximity assumes that vertices sharing several connections to other vertices are similar to one another.

LINE suits directed or undirected, weighted or unweighted information networks and scales well. Given an information network $G = (V, E)$, without loss of generality, the authors have assumed it to be always directed. An undirected network can be considered as a directed network by treating each undirected edge as two directed edges with opposite directions and equal weights.

4.2.1 Optimizing the first-order proximity

The first-order proximity refers to the local pairwise proximity between the vertices in the network. To model the first-order proximity, for each undirected edge (i, j), we define the joint probability between vertex v_i and v_j as follows:

$$p_1(v_i, v_j) = \frac{1}{1 + e^{-\vec{u}_i^T \cdot \vec{u}_j}}$$

where $\vec{u}_i \in R^d$ is the low-dimensional vector representation of vertex v_i.

The proximity measure defines a distribution $p(\cdot, \cdot)$ over the space $V \times V$, and its empirical probability can be defined as:

$$\hat{p}_1(i, j) = \frac{w_{ij}}{\sum_{(i,j)\in E} w_{ij}}$$

4.2.2 Optimizing the second-order proximity

The second-order proximity is also applicable for both directed and undirected graphs. Each vertex plays two roles:

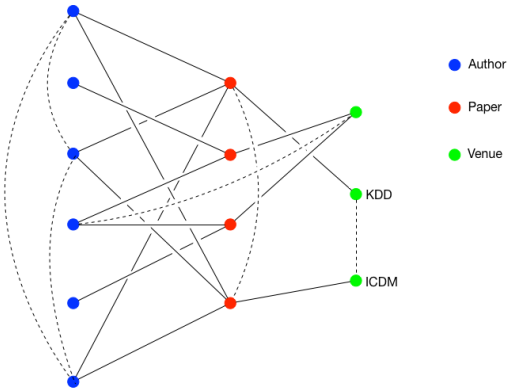

Figure 4: Densification of the original network from Figure 1

the vertex itself and a specific "context" of other vertices.

Let $\vec{u}_i$ be the representation of v_i, while $\vec{u}'_i$ is the representation of v_i when it is treated as a specific "context". For each directed edge (i, j), we first define the probability of "context" v_j generated by vertex v_i as:

$$p_2(v_j|v_i) = \frac{e^{\vec{u}_j'^T \cdot \vec{u}_i}}{\sum_{k=1}^{|V|} e^{\vec{u}_k'^T \cdot \vec{u}_i}},$$

where $|V|$ is the number of vertices or "contexts." Similar to the first-order case,the empirical distribution is defined as:

$$\hat{p}_2(\cdot|v_i) = \hat{p}_2(v_j|v_i) = \frac{w_{ij}}{d_i},$$

where w_{ij} is the weight of the edge (i, j) and d_i is the out-degree of vertex i, i.e. $d_i = \sum_{k\in N(i)} w_{ik}$, where $N(i)$ is the set of out-neighbors of v_i. For each vertex v_i, the conditional distribution $p_2(\cdot|v_i)$ is defined over the contexts, i.e., the entire set of vertices in the network.

4.2.3 Combining first-order and second-order proximities

The network embedding process is done by preserving both the first-order and second-order proximity. The authors of LINE had chosen to train the LINE model by preserving the first-order proximity and second-order proximity after which the trained embedding outputs from each method are concatenated for each vertex. We employ the same technique in our project.

4.3 Extracting richer topological features

In this section, we describe the topological features we can now extract in our densified network, which wouldn't have been possible in the orignal network. Since we embedded the original network using edge weights obtained using LINE, our densified network has edges between all possible pairs of heterogeneous nodes, unlike earlier. A densified version of the original DBLP schema from Figure 1 is shown in Figure 4. Note that among the meta-paths provided, only the first two can be extracted from the original sparse network.

Meta Path	Semantic Meaning of the Relation
$A - P - V$	Number of papers by the author at the venue
$A - P - A - P - V$	Number of papers written by the collaborators of the author of interest at the venue
$A - V$	Embedding similarity obtained between the author and the venue
$A - A - V$	Captures the similarity of the collaborators with the target venue
$A - V - V$	Captures the similarity of the author's favorite venues with the target venue

Table 1: Richer MetaPaths using Densification

4.3.1 *New topological features upon Densification*

Since we densify the orignal network via edges between authors, between papers, between venues and between nodes of different heteregeneous types, several new metapaths are now feasible, which wasn't the case in the sparse, original network. Once we mine such interesting metapaths, we can feed them into our classifier to perform accurate predictions. We summarize the metapaths originating between an author and a venue in Table 1.

4.4 The learning algorithm

The classification task is to predict whether a author who has not published at a particular venue in one time period (before 2006) will publish at that venue in a future time period (2006 onwards).

Various metapaths are defined, and different "measures", or methods of quantification are aggregated. These aggregations are used as features for a logistic regression classifier that is trained to predict whether a pair of authors will become co-authors in the future.

For each training pair of authors (a_i, v_j), let x_m be the $(d+1)$-dimensional vector including constant 1 and d topological features between them, and y_m be the label of whether a_i will publish at v_j in the time period ($y_m = 1$, if a_i does eventually publish at v_j, and otherwise 0), which follows binomial distribution p_m. The probability p_m is modeled as follows :

$$p_m = \frac{e^{x_m \beta}}{e^{x_m \beta} + 1},$$

where β is the $d+1$ coefficient weights associated with the constant and each topological feature.

The standard MLE (Maximum Likelihood Estimation) to derive $\hat{\beta}$, that maximizes the likelihood of all the training pairs:

$$L = \prod_{i=1} p_m^{y_m} (1 - p_m)^{(1-y_m)}$$

For each candidate (author, venue) pair, the prediction is based the following criterion :

$$P(y_{test} = 1) = \frac{e^{x_{test}\hat{\beta}}}{e^{x_{test}\hat{\beta}} + 1}$$

where, x_{test} is the $(d+1)$ dimensional vector including constant 1 and d topological features between the candidate pair.

5. EXPERIMENTS

In this section, we evaluate the effectiveness of our approaches. We conduct our experiments on the DBLP dataset. We first describe the experiment setup, show the results and discuss the intuition behind these results.

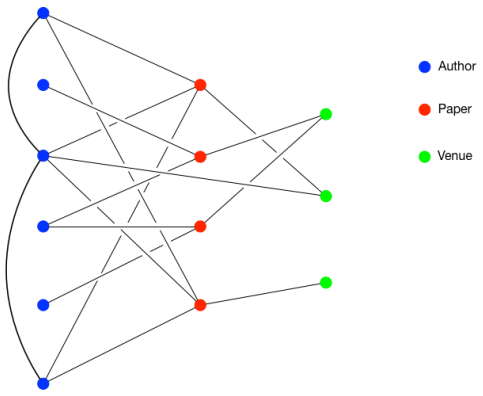

Figure 5: Enriching the original network from Figure 1 by introducing A-A and A-V weighted edges.

5.1 Experiment Setup

All our experiments were run over the DBLP dataset, whose summary statistics are as follows: 1) Number of Authors: 114308, 2) Number of Papers: 113481, 3) Number of Venues: 143. The total number of edges in the network is 328153.

5.2 Compared Algorithms

We consider the following algorithms in our experiment setup:

1. **PathPredict**: PathPredict is the supervised learning framework we discussed in Section 4.1.

2. **APV-LINE**: APV-LINE is our first proposed method which we proposed in Section 4.2.

3. **AV-LINE**: AV-LINE is an extension from the APV-LINE method we proposed initially. In APV-LINE, we embed the initial network with node similarities before performing classification. However, in the AV-LINE framework, we embed the initial network with edges *before* performing embedding. The intuition is that introducing informative edges in the network before performing embedding will improve the quality of inference.

   We demonstrate how we enrich the original APV network with edges between authors who have collaborated and (author, venue) pairs where the former has published in the latter in Figure 5.

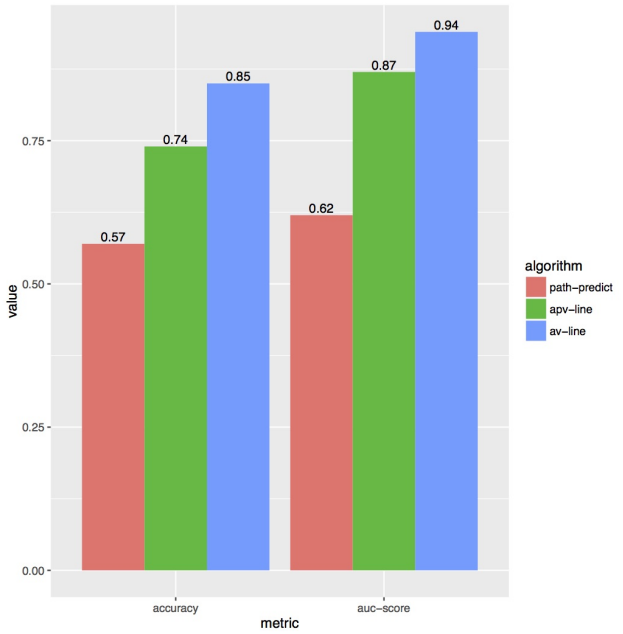

Figure 6: Comparison between PathPredict, APV-LINE and AV-LINE. AV-LINE outperforms on both Accuracy and AUC score.

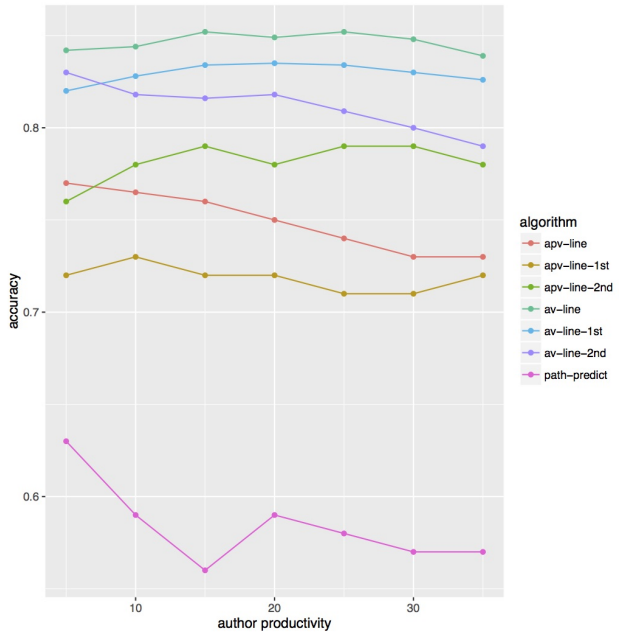

Figure 7: Performance analysis upon varying the author productivity

5.3 Head-to-Head comparison

We demonstrate the effectiveness of our method in Figure 6. PathPredict gave an accuracy of 0.57. The accuracy of APV-LINE was 0.74, which is a considerable improvement from PathPredict. However, the icing on the cake was the performance of AV-LINE that had an accuracy of 0.85, which is a 47% improvement from PathPredict.

We also observe an improvement in the AUC score. The AUC score of PathPredict was 0.62. The AUC score APV-LINE was 0.87, which is an improvement from PathPredict. Similarly to accuracy, we observe a considerable improvement in the AUC score with AV-LINE with a value of 0.94. In conclusion,

AV-LINE improves the accuracy from PathPredict by up to 47%, and improves the AUC score by up to 52%.

5.4 Varying author productivity

In our next experiment, we vary the productivity of the authors in our testing set and see if performance improves/reduces as we vary the author productivity.

We vary the number of papers the author must have written to be considered in the testing set in the axis, and show the accuracy in the y-axis. We observe that in all scenarios, we beat our baseline. However, APV-LINE is not as good when we only consider the 2nd order proximity. However, AV-LINE is better than all the other algorithms in all the scenarios, which means we are in some sense getting the best out of both worlds.

Our main insights from the above experiments are as follows:

1. Introduction of AA and AV edges further improved accuracy!
2. AV-LINE gains from both 1st-order and 2nd-order embedding

5.5 Significant Features

In this section, we analyze what features improved the accuracy of AV-LINE in comparison to PathPredict and APV-LINE, and we observed the following important fact:

The features AV, AAV and AVV single-handedly outperformed PathPredict. Even more importantly, these features can not be extracted from the original network since AA and AV edges aren't present. This further supports our intuition and justifies our methodologies.

6. CONCLUSIONS

We consider the problem of predicting relationships in large-scale heterogeneous networks. The reasoned that the main challenge is to extract latent information from such real-world networks which are typically sparse, and showed how existing relationship prediction techniques do not take into account the sparsity of real-world networks. We developed methods to combat network sparsity by using node similarities obtained from state-of-the-art embedding techniques, and we described how to enrich the original sparse network with latent relationships to further improve the quality of embedding, and consequently the prediction performance of our method. By evaluating under different scenarios, we demonstrated that in practice, our methods are highly effective in predicting relationships in sparse networks, and improve the accuracy by up to 47% over existing techniques.

7. ACKNOWLEDGEMENT

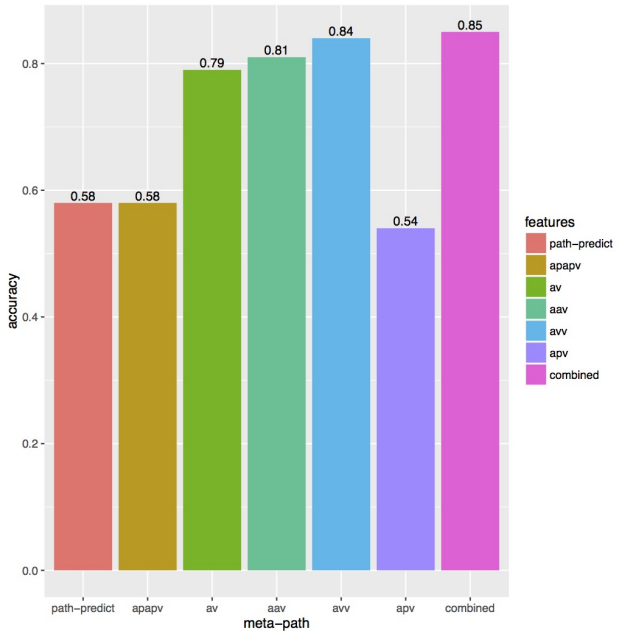

Figure 8: Performance analysis upon varying the features

We thank Meng Jiang for useful discussions and feedback regarding the research project.

8. REFERENCES

[1] Yizhou Sun and Jiawei Han. *Mining Heterogeneous Information Networks: Principles and Methodologies*. Morgan & Claypool Publishers, 2012.

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

[3] Yizhou Sun, Rick Barber, Manish Gupta, Charu C. Aggarwal, and Jiawei Han. Co-author relationship prediction in heterogeneous bibliographic networks. In *Proceedings of the 2011 International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '11, pages 121–128, Washington, DC, USA, 2011. IEEE Computer Society.

[4] Yizhou Sun and Jiawei Han. Mining heterogeneous information networks: A structural analysis approach. *SIGKDD Explor. Newsl.*, 14(2):20–28, April 2013.

[5] Jiawei Zhang and Philip S. Yu. Link prediction across heterogeneous social networks: A survey. *SOCIAL NETWORKS*, 2014.

[6] Mohammad Al Hasan and Mohammed J. Zaki. *Social Network Data Analytics*, chapter A Survey of Link Prediction in Social Networks, pages 243–275. Springer US, Boston, MA, 2011.

[7] Aditya Krishna Menon and Charles Elkan. Link prediction via matrix factorization. In *Proceedings of the 2011 European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part II*, ECML PKDD'11, pages 437–452, Berlin, Heidelberg, 2011. Springer-Verlag.

[8] 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.

[9] Jian Tang, Meng Qu, and Qiaozhu Mei. PTE: predictive text embedding through large-scale heterogeneous text networks. *CoRR*, abs/1508.00200, 2015.

[10] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. LINE: large-scale information network embedding. *CoRR*, abs/1503.03578, 2015.

[11] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. In *In VLDBâĂŹ 11*, 2011.