PNR: How to Optimally Combine Different Link Prediction Approaches?

Anonymous Author(s)

ABSTRACT

Link prediction aims to predict missing or future links in complex networks. There exist an increasing number of link prediction methods, and each method has its own advantages and disadvantages. However, the problem of how to effectively and efficiently combine different methods is far from understood. Here, we address the problem by proposing a bayesian-based strategy to *optimally* combine different methods. Mathematically, We present a comprehensive analysis on the mechanism of the hybrid method, and then propose an explainable metric (PNR, **P**recision-**N**oise-**R**atio) to characterize the prediction accuracy of uncertain links, which is based on the characteristic difference of existing links and nonexisting links. Algorithmically, we provide a scalable and *parameter-free* algorithm to *optimally* combine different methods. Besides, we analytically prove the optimum of the proposed method. Empirically, we compare our method with the state of the art methods in different real datasets. The results not only demonstrate the effectiveness of the proposed method, but also explain why the proposed method could increase the prediction accuracy.

KEYWORDS

Link prediction, Hybrid prediction, Combination, Complex Networks, Machine learning

1 INTRODUCTION

With the explosive increase of Internet, online social networks and other different kinds of complex networks have attracted much attention in recent years [3, 25, 33]. Among the network analysis, link prediction could benefit researchers and engineers in a variety of fields: In social networks, such as Facebook and Twitter, link prediction helps users efficiently find their lost friends and potential future relationships [6]. In biology and drug research, link prediction predicts potential reaction between proteins to reduce expensively experimental tests [13]. The broad applicability demands a powerful and general framework of link prediction methods [11, 21].

Classical link predictions usually assign a specific score for each nonexisting link (between unconnected nodes) and then choose links with high scores as the predicted ones [4, 21]. Thus, the central issue is to design effective strategies to characterize the similarity scores. The straightforward approach is using the neighborhood information to calculate similarity, including *Common Neighbor*, *Jaccard coefficient* and so on [21]. Apart from local information, global information could also be used, e.g. *Katz* and *average commute time* [20]. Besides, in some researches, we suppose that networks follow some underlying rules and could be embedded into low dimensional space, and thereafter many matrix factorization based methods are proposed [19, 29]. Moreover, some other information is also introduced into the link prediction, e.g., the timestamps and tags of links, semantic information between individuals, details of users [20, 21]. To effectively combine different factors, deep learning domain provides convenient access to obtain the hyper-parameters to weigh the importance of different factors, especially the graph convolutional neural network and neural graph embedding [21].

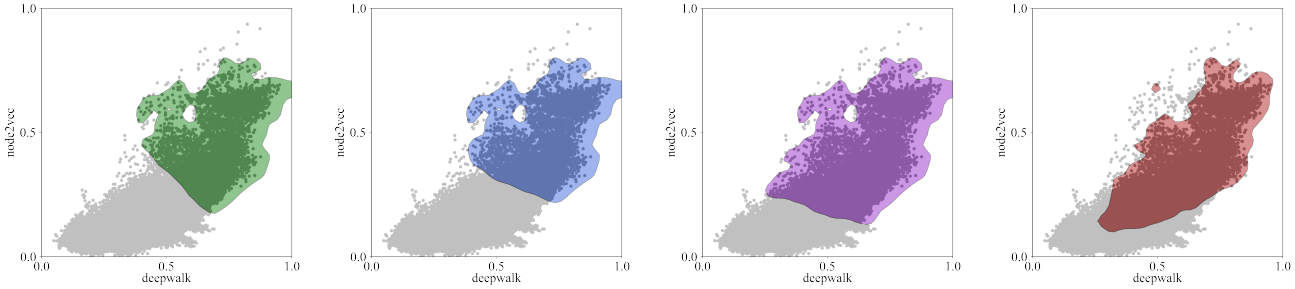
Though a large number of link prediction approaches were proposed, Maurizio et al. [7] showed that little progress about the prediction in bipartite networks was made in the past few years, implying that continuingly proposing new prediction methods is of high challenge. What is worse, considering additional auxiliary information of nodes and links benefits the prediction accuracy, which, however, would sharply increase the complexity and reduce the robustness of the prediction systems. It is noticed that different existing approaches have their own advantages and disadvantages [12]. Alternatively, we can combine different existing approaches to increase the prediction performance, including the weighted combinations (linear or nonlinear) and switching systems [1]. Besides, some researchers design elaborate combinations by modifying one method based on the specific characteristics of other methods. However, to our knowledge, the underlying mechanism of the general combination schemes is rarely investigated, and little attention was paid to the design of the optimal combination.

In the paper, we use the posterior bayesian estimation to investigate the underlying mechanism of the optimal combination and build its relationship with the characteristics of the existing links and nonexisting ones, which is formulated as a new metric (PNR, Precision-Noise-Ratio). Based on PNR, a scalable framework is proposed to combine different existing link prediction approaches. We also analytically prove the optimum of the proposed framework. We argue that the proposed framework is capable of combining arbitrary existing link prediction approaches.

The contributions of our work are as follows:

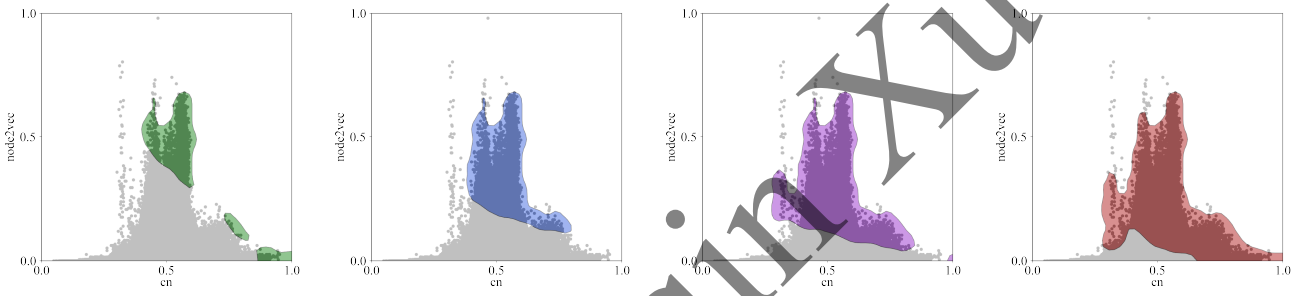
- *Effectiveness*: PNR provides convenient access to combine different methods, including:
  - *Baseline generalization*: PNR could be generalized to various existing link prediction approaches.
  - *Optimum*: PNR is the optimum combination of different methods.
  - *Sense-making*: The underlying mechanism of PNR can be easily understood based on posterior bayesian estimation.
- *Automation*: PNR is carefully designed to be parameter-free, without requiring the hyper parameters.
- *Scalability*: PNR does not increase the time complexity of the existing link prediction systems.

We want to emphasize that PNR as a combination framework is fundamentally different from the existing parameter-tuning-based combinations. We investigate the underlying mechanism of the combinations, and thus could propose the optimal, especially parameter-free method.



(a) weighted-plus predicted region (b) weighted-multiply predicted region (c) MLP predicted region (d) PNR predicted region

Figure 1: The predicted regions by four different combination schemes while combining deepwalk and node2vec models on facebook dataset. Every point in the figure, which is scattered on the gray background, represents a nonexisting link with two scores computed by deepwalk (X-axis) and node2vec (Y-axis) respectively.



(a) weighted-plus predicted region (b) weighted-multiply predicted region (c) MLP predicted region (d) PNR predicted region

Figure 2: The predicted regions by four different combination schemes while combining CN and node2vec models on facebook dataset. Every point in the figure, which is scattered on the gray background, represents a nonexisting link with two scores computed by CN (X-axis) and node2vc (Y-axis) respectively (See the detailed baseline methods in section 5).

Both Figure 1 and Figure 2 present representative examples that depict the difference between PNR method and the existing combination methods. PNR could predict links with various scores, especially the low-score links by both baseline methods(down-left region in panel **??**(a)), whereas the existing combination methods mostly predict links of high scores (at least one high score by either method).

2 RELATED WORK

The problem of how to combine different methods touches on the domain of traditional link prediction methods and the analysis of combination mechanism, which we briefly discuss here.

Link prediction: Most earlier works on link prediction were unsupervised, using neighborhoods or different kinds of paths to characterize the similarity between nodes [20]. According to the structural information used in the similarity calculation, unsupervised methods could be further divided into local and global methods [21, 32]. Local methods only require small quantity of information about neighboring connections to characterize the similarity of pairwise nodes [32], while global methods utilize the whole network structure to evaluate the similarity scores [18]. Recently, supervised methods have attracted some attention [17], since they allow some blending of topological features with semantic ones. Semantic features used in the literature include social labels (e.g., interests and hobbies), textual labels (e.g., paper titles, keywords, PACS), location features and social interactions, and so on [20]. For more information, please refer to the refs. [20, 21]. In this work, we only consider the link prediction based on topological structure.

Combination scheme: The schemes of combining different existing link prediction approaches could be divided into two classes (see Fig. 3): **(a)** We can investigate the mechanism of each method and modify the inner structure of one method based on the characteristics of another methods. For example, Leskovec et al. [30] combine random walks and graph convolutions to embed networks based on which link prediction is performed. Wang et al. [28] utilize multiple deep autoencoders to embed networks for link prediction. Besides, some other technology and other information could also be used to modify the existing methods, e.g., the Natural Language Processing (NLP) for text data, and the Conventional Neural Network (CNN) for picture information, and so on [1, 2]. See refs. [1, 21, 33] for more combination schemes. However, this kind of combination requires in-depth understanding of the structure

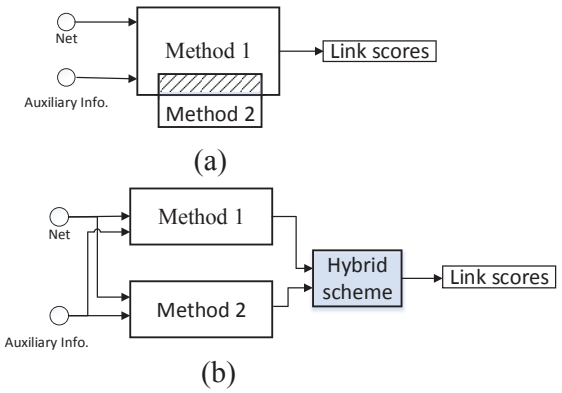


Figure 3: Two classes of hybrid link prediction approaches. (a) Modifying the inner structure of an existing method based on the specific characteristic of another method. (b) Using the results of two existing methods to generate new predicting results.

and characteristics of the link prediction approaches being studied, which cannot be generalized to combine more different approaches. **(b)** We first calculate the prediction results using each method, and then combine the different results of all methods to generate a new prediction result, including the weighted scheme, switching scheme, and the mix scheme [1]. The weighted scheme uses linear (or nonlinear) formula to combine the link scores of different link prediction approaches [22, 27]. The switching scheme adopts some certain criteria to determine which baseline approaches to use under different scenarios [5, 8]. The mix scheme synthesizes the weighted and switching schemes simultaneously. As a matter of fact, the underlying mechanism of the three combinations could be represented by a general bivariate function $y = F(x_1, x_2)$ in Fig. 3(b), with different combination schemes corresponding to different formulas $F(x_1, x_2)$. However, to our knowledge, the optimal function $F_{opt}(x_1, x_2)$ is still unknown. Since $F(x_1, x_2)$ could be easily generalized to various existing approaches, we concern the optimal bivariate function in the paper.

3 PRELIMINARIES

We first introduce some terms and notations in Table 1. Consider an undirected network $G = (V, E)$ with V and E representing the node set and link set respectively. If there exists a link between u and v, $e_{u,v} = 1$, otherwise $e_{u,v} = 0$. In the link prediction problem, we usually divide the links randomly into $1-p^H$ training set E^{train} and p^H test set E^{test} ($p^H \in (0, 1)$), with $E^{train} \cup E^{test} = E$ and $E^{train} \cap E^{test} = \emptyset$.

The link prediction approaches use the links in training set to predict the links in the test set. Most classical methods usually first assign a connecting likelihood score $s_{u,v}$ to link $e_{u,v}$ based on E^{train}, and then produce a ranked list in the descending order of $s_{u,v}$. Note that link prediction is not personalized, and we predict a fixed size of links for the whole network rather than for each node. In the paper, we set the length of ranked list as the size of the test set. The central evaluation is the ratio of correctly predicted links, i.e., *precision*.

Here, we use the bivariate function $y = F(x_1, x_2)$ to combine two different baseline approaches, where x_1 and x_2 are the two scores computed by two existing methods (see Fig. 3(b) for the framework of the our proposed combination method). The aim is to investigate the optimal bivariate function $F_{opt}(x_1, x_2)$ to maximize the prediction accuracy. Note that we can also combine more than three existing approaches by firstly combining two approaches, and then using the obtained results to combine with the third approach. That is to say, investigating the optimal combination of two existing approaches also means the combination of more than three approaches. Hence, we only concern the optimal bivariate function $y = F(x_1, x_2)$ in the paper.

Table 1: Common notations.

Notation	Description
G	An undirected network
V	The set of nodes
E	The set of all existing links
E^{train}	The set of links in the training set
E^{test}	The set of links in the test set
$s_{u,v}$	The similarity score between node u and node v
L	The length of prediction list
E^{non}	Set of nonexisting links in training set
p_n	The joint score distribution of nonexisting links
p_r	The joint score distribution of existing links

4 THE PROPOSED FRAMEWORK

In this section, we first present the motivation and the details of PNR framework, and then provide the proof of the optimum of PNR.

4.1 The Analytical Mechanism of PNR

Supposing that $s^{(A)}_{u,v}$ and $s^{(B)}_{u,v}$ are the similarity scores between node u and v, obtained by two arbitrary link prediction methods (denoted by A and B respectively). For a pair of nodes u and v in the training set, the probability that there is a link between u and v in the training set is $p(e_{u,v} = 1|s^{(A)}_{u,v}, s^{(B)}_{u,v})$ on condition of the similarity scores $s^{(A)}_{u,v}$ and $s^{(B)}_{u,v}$. According to the bayesian estimation,

$$p(e_{u,v} = 1|s^{(A)}_{u,v}, s^{(B)}_{u,v}) = \frac{p(e_{u,v} = 1) \cdot p(s^{(A)}_{u,v}, s^{(B)}_{u,v}|e_{u,v} = 1)}{p(s^{(A)}_{u,v}, s^{(B)}_{u,v})}, \quad (1)$$

where $p(s^{(A)}_{u,v}, s^{(B)}_{u,v})$ is the probability that a randomly chosen (existing or nonexisting) link in the training set has scores $s^{(A)}_{u,v}, s^{(B)}_{u,v}$. Note that $p(e_{u,v} = 1) = |E^{train}|/(|V| \times (|V| - 1)/2)$ is a constant representing the probability of an existing link between a random node pair in the training set. Besides, since real networks are usually sparse and the size of nonexisting links is much larger than that of existing links, $p(s^{(A)}_{u,v}, s^{(B)}_{u,v})$ is mainly determined by the score distribution of nonexisting links. That is to say, $p(s^{(A)}_{u,v}, s^{(B)}_{u,v}) \approx$

$p_n(s^{(A)}_{u,v}, s^{(B)}_{u,v}) = p(s^{(A)}_{u,v}, s^{(B)}_{u,v}|e_{u,v} = 0)$, where $p_n(s^{(A)}_{u,v}, s^{(B)}_{u,v})$ is the probability that a randomly chosen nonexisting link in the training set has scores $s^{(A)}_{u,v}$ and $s^{(B)}_{u,v}$ based on methods A and B. Thus, we arrive at

$$p(e_{u,v} = 1|s^{(A)}_{u,v}, s^{(B)}_{u,v}) = c_1 \cdot \frac{p(s^{(A)}_{u,v}, s^{(B)}_{u,v}|e_{u,v} = 1)}{p_n(s^{(A)}_{u,v}, s^{(B)}_{u,v})}, \quad (2)$$

where $c_1 = p(e_{u,v} = 1)$ is a constant. Neglecting the constant term, we propose **P**recision-to-**N**oise **R**atio (PNR),

$$PNR(s^{(A)}_{u,v}, s^{(B)}_{u,v}) = \frac{p(s^{(A)}_{u,v}, s^{(B)}_{u,v}|e_{u,v} = 1)}{p_n(s^{(A)}_{u,v}, s^{(B)}_{u,v})} = \frac{p_r(s^{(A)}_{u,v}, s^{(B)}_{u,v})}{p_n(s^{(A)}_{u,v}, s^{(B)}_{u,v})}, \quad (3)$$

where $p_r(s^{(A)}_{u,v}, s^{(B)}_{u,v}) = p(s^{(A)}_{u,v}, s^{(B)}_{u,v}|e_{u,v} = 1)$ is the probability that a randomly chosen existing link in the training set has scores $s^{(A)}_{u,v}$ and $s^{(B)}_{u,v}$. Note that both $p_r(s^{(A)}_{u,v}, s^{(B)}_{u,v})$ and $p_n(s^{(A)}_{u,v}, s^{(B)}_{u,v})$ could be obtained only from the training set E^{train}. $PNR(s^{(A)}_{u,v}, s^{(B)}_{u,v})$ measures the ratio of score distributions between existing links and nonexisting links.

In the evaluation of link predictions, due to the random division between training set and test set, links in the test set should have the same (or similar) score distribution with that of the training set. Thus, we would not differentiate score distributions $p_r(s^{(A)}_{u,v}, s^{(B)}_{u,v})$ and $p_n(s^{(A)}_{u,v}, s^{(B)}_{u,v})$ for the training set and test set in the following part.

Note that a nonexisting link in the training set might be an existing link in the test set. Based on random division, given a pair of nodes with the scores $s^{(A)}_{u,v}$ and $s^{(B)}_{u,v}$, the probability that the link is an existing link in the test set is

$$p' = \frac{|E^{test}|}{|E^{train}|} \cdot p(e_{u,v} = 1|s^{(A)}_{u,v}, s^{(B)}_{u,v}) = \rho \cdot PNR(s^{(A)}_{u,v}, s^{(B)}_{u,v}). \quad (4)$$

where $\rho = |E^{test}|/|E^{train}|$.

Remark 1: Eq. 4 indicates that the prediction accuracy is determined by PNR rather than the pure similarity score. Therefore, we should investigate the optimal bivariate function $F_{opt}(x_1, x_2)$ based on PNR.

4.2 The PNR-based Combination Algorithm

Based on PNR, our framework starts with re-considering the definition of *precision* that is the ratio of correctly predicted links for the test set [18].

$$precision = \frac{l_r}{L}, \quad (5)$$

where l_r is the size of correctly predicted links. Clearly, higher precision is better, $precision \in [0, 1]$.

Supposing that nonexisting links (in the training set) with scores $(s^{(A)}, s^{(B)}) \in S$ are predicted as potential links, in order to guarantee the size of prediction list $L = |E^{test}|$, the score set S satisfies $(|U| - |E^{train}|) \oint_S p_n(s^{(A)}, s^{(B)}) ds^{(A)} ds^{(B)} = |E^{test}|$. The expectation of l_r is $l_r = |E^{test}| \oint_S p_r(s^{(A)}, s^{(B)})\, ds^{(A)} ds^{(B)}$. Substituting L and l_r into eq. 5, we obtain

$$precision = \gamma \cdot \frac{\oint_S p_r(s^{(A)}, s^{(B)}) ds^{(A)} ds^{(B)}}{\oint_S p_n(s^{(A)}, s^{(B)}) ds^{(A)} ds^{(B)}}, \quad (6)$$

where $\gamma = |E^{test}|/(|U| - |E^{train}|)$ is a constant.

The upper bound of the *precision* in eq. 6 follows as (See eq. 10 for the detailed derivation):

$$precision \le \gamma \cdot PNR_{max}(s^{(A)}, s^{(B)}), \forall (s^{(A)}, s^{(B)}) \in S. \quad (7)$$

Remark 2: The *precision* (left hand side of eq. 7) could reach the maximum only if we choose links whose similarity scores correspond to maximal $PNR(s^{(A)}, s^{(B)})$ (See the *Supplementary A* for the proof). That is to say, we choose the links whose scores belong to the optimal set S satisfying $PNR(s^{(A)}_{u,v}, s^{(B)}_{u,v}) > PNR(s^{(A)}_{u',v'}, s^{(B)}_{u',v'})$, $\forall (s^{(A)}_{u,v}, s^{(B)}_{u,v}) \in S$ and $\forall (s^{(A)}_{u',v'}, s^{(B)}_{u',v'}) \notin S$.

We explicitly use the score set S to determine the predicted links in eqs.6-7. In real scenarios, we do not require to calculate the set S explicitly. Alternatively, we set the optimal bivariate function $F_{opt}(x_1, x_2) = PNR(x_1, x_2)$ and use the descending order of the hybrid scores $s'_{u,v} = PNR(s^{(A)}_{u,v}, s^{(B)}_{u,v})$ for all nonexisting links to predict links. The optimal score set S is made up of the scores of the predicted links. In fact, link prediction only requires the ranked list of nonexisting links, without calculating S explicitly.

Note that the central task is to calculate $F_{opt}(x_1, x_2) = PNR(x_1, x_2)$ that depends only on the score distributions ($p_n(s^{(A)}, s^{(B)})$ and $p_r(s^{(A)}, s^{(B)})$) of links in training set. We can first calculate $p_n(s^{(A)}, s^{(B)})$ and $p_r(s^{(A)}, s^{(B)})$, and then obtain $PNR(s^{(A)}, s^{(B)})$ based on eq. 3. In the computation, we discretize $p_n(s^{(A)}, s^{(B)})$ and $p_r(s^{(A)}, s^{(B)})$ by dividing the scores into $b \times b$ uniform cells with interval lengths $[(s^{(A)}_{max} - s^{(A)}_{min})/b]$ and $[(s^{(B)}_{max} - s^{(B)}_{min})/b]$.

We propose a generic framework to conduct link prediction based on $PNR(s^{(A)}, s^{(B)})$ (See Algorithm 1):

(1) Given two classical prediction algorithms, we use them to calculate the similarity scores for all (existing and nonexisting) links, with each link having two scores $s^{(A)}$ and $s^{(B)}$.
(2) Compute the discrete $p_n(s^{(A)}, s^{(B)})$ and $p_r(s^{(A)}, s^{(B)})$, and then obtain $PNR(s^{(A)}, s^{(B)})$.
(3) Redefine the score of each link $s'_{u,v} = PNR(s^{(A)}_{u,v}, s^{(B)}_{u,v})$ and generate the prediction list by the descending order of all scores $s'_{u,v}$. (The optimal score set S is made up of the scores of the predicted links, which does not require to be calculated explicitly.)

In order to ensure the reproducibility of our method, we are releasing an implementation of PNR at *https://github.com/google-micro-research/PNR*.

4.3 Analysis of Time Complexity

Calculating $PNR(s^{(A)}, s^{(B)})$ increases the time consumption of the link prediction systems. For the state of the art methods, we need to calculate the similarity scores of all nonexisting links in which the lower limit of the time complexity is $\Omega(|V|^2)$. In the proposed framework, counting $p_r(s^{(A)}, s^{(B)})$ and $p_n(s^{(A)}, s^{(B)})$ requires traversing all links between users with time complexity $O(|V|^2)$. Thus, introducing PNR in the framework does not increase the time complexity of most classical link prediction systems.

Algorithm 1: RNR Framework

Input: Implicit feedback matrix $\mathbf{R} \in \mathrm{R}^{n \times n}$ (training set)
Output: Hybrid similarity score matrix $\mathbf{M}^{PNR}$
Initialize the baseline link prediction methods A and B;
Compute the similarity score matrix $\mathbf{M}^{A}_{n \times n}$ and $\mathbf{M}^{B}_{n \times n}$ via methods A and B respectively;
Initialize $b \times b$ cells based on the maximum and minimum entries of $\mathbf{M}^{A}_{n \times n}$ and $\mathbf{M}^{B}_{n \times n}$;
for $i = 1; i \leq b$ **do**
  Count how many existing links $n_{s^{(A)}, s^{(B)}}$ locate in each cell (x_1, x_2) and then compute $p_r(s^{(A)}, s^{(B)})$ via $p_r(s^{(A)}, s^{(B)}) = n_{s^{(A)}, s^{(B)}} / |E^{train}|$;
  Count how many nonexisting links $n_{s^{(A)}, s^{(B)}}$ locate in each cell (x_1, x_2) and then compute $p_n(s^{(A)}, s^{(B)})$ via $p_n(s^{(A)}, s^{(B)}) = n_{s^{(A)}, s^{(B)}} / |U - E^{train}|$;
  if $p_n(s^{(A)}, s^{(B)}) == 0$ **then**
    $PNR(s^{(A)}, s^{(B)}) = 0$;
  else
    Compute $PNR(s^{(A)}, s^{(B)}) = p_r(s^{(A)}, s^{(B)}) / p_n(s^{(A)}, s^{(B)})$;
  end
end
Obtain the hybrid score matrix $\mathbf{M}^{PNR}$ by $\mathbf{M}^{PNR} = PNR(\mathbf{M}^{A}, \mathbf{M}^{B})$;
return $\mathbf{M}^{PNR}$.

Table 2: Statistics of datasets. $|V|$ denotes the number of vertexes and $|E|$ denotes the number of links.

Dataset	$\lvert V\rvert$	$\lvert E\rvert$	Average Degree	Sparsity
facebook	4,039	88,234	21.8	1.08×10^{-4}
ca-hepPh	12,008	237,010	19.7	3.29×10^{-3}
yeast	2375	11,693	4.96	4.2×10^{-3}
email-eucore	1005	25,571	25.4	5.07×10^{-2}

5 EXPERIMENTS

We first introduce the experiment setup, and then show the results and analysis on the four real-world datasets. Note that the experiments were conducted on a server with Intel Xeon(R) CPU Gold 5118 (2.30GHz) and 32G RAM. For each baseline, we use the released codes from the original authors to reproduce, which are all implemented by Python 3.5 version. Besides, we use *skleran* package to implement the combination of link prediction methods with MLP(multi-layer perceptron).

5.1 Experimental Setup

5.1.1 Datasets. We conduct the experiments in four real open networks [14, 15, 24]. All the networks are treated undirected and unweighted in Table 2. The detailed descriptions are listed as follows.

- **facebook** [1]: In the network, nodes represent users, and edges represent a friendship relation between two users.
- **ca-hepph** [2]: This is a collaboration network generated from papers submitted to the e-print arXiv where nodes represent scientists, and a link represents whether two scientists have collaborated in a paper.
- **yeast** [3]: This undirected network contains protein interactions contained in yeast. A node represents a protein and a link represents a metabolic interaction between two proteins.
- **email-eucore** [4]: The network was generated using email data from a large European research institution. There is a link (u, v) in the network if person u sent person v at least one email.

To evaluate and compare the performance, we randomly divide the links of each dataset into 80% training set and 20% test set. All the reported performances are the average of 10 independent simulations.

5.1.2 Combination Baselines and Parameter Setting. We compare the proposed combination method with three widely used baselines:

- ***weighted plus***: The method generates new scores by assigning weight to different methods, $s'_{u,v} = \beta s^{(A)}_{u,v} + (1 - \beta) s^{(B)}_{u,v}$, where $s^{(A)}_{u,v}$ and $s^{(B)}_{u,v}$ are two scores computed by methods A and B respectively. β is a tunable parameter to tune the importance of different methods, $\beta \in (0, 1)$. In the experiments, we set $\beta = 0.5$.
- ***weighted product***: The method calculates new scores based on nonlinear power function, $s_{u,v} = s^{(A)\,\beta}_{u,v} \cdot s^{(B)\,(1-\beta)}_{u,v}$, $\beta \in (0, 1)$. In the experiments, we set $\beta = 0.5$.
- ***MLP***: The Multi-Layer Perceptron (MLP) uses neural networks to imitate various functions. Here, we set the MLP two inputs and one output, where the two inputs mean the scores comupted by two existing methods and the output means the new scores of MLP.

Note that apart from the three combinations introduced above, there exist a large number of different combination schemes that could all be imitated by the MLP. MLP is an universal approach to reproduce various bivariate functions. Hence, we do not discuss other combinations here. The three methods plus the proposed PNR based method are listed in table 3.

Another notice is that we use symbols A and B to represent arbitrary existing link prediction approaches. In the experiments, we use eight existing approaches that consist of four state of the art deep learning based approaches and four classical network-based approaches. The four deep learning based approaches are: **(a) *Deepwalk*** [23] is a two-phase method for embedding graphs. We set embedding size as 128, window size as 10, walk length as 40 and walks per node as 80. **(b) *Node2vec*** [9] proposes an improvement to the random walk phase of DeepWalk. Specifically, we set embedding size as 128, window size as 10, walk length as 80 and walks per node as 10, and the optimization is run for a single epoch. Besides, in node2vec, parameters $p_{node2vec}$ and $q_{node2vec}$ that control how fast the walk explores and leaves the neighborhood of

[1] http://snap.stanford.edu/data/index.html
[2] http://snap.stanford.edu/data/cit-HepPh.html
[3] http://networkrepository.com/bio-yeast-protein-inter.php
[4] http://snap.stanford.edu/data/email-Eu-core.html

Table 3: The definitions of combination operators. The notation A and B denote two arbitrary baselines.

Operator	Definition
$A + B$	weighted plus
$A \times B$	weighted product
MLP	combining A with B using MLP
PNR	combining A with B using PNR

starting node are both set to 1.0. **(c) *DRNE*** [26] utilizes a layer normalized LSTM to represent each node by aggregating the representations of their neighborhoods in a recursive way. We generally set the length of embeddings as 16, the weight of the regularizer is 1, the learning rate is 0.0025 and the limited neighborhood number is 300. **(d) *ProNE*** [31] is a fast, scalable, and effective embedding model. For ProNE, the term number of the Chebyshev expansion is set to 10, $\mu = 0.2$, and $\theta = 0.5$, which are the default settings. All the parameters have been optimized in the experiments (or use the suggested value of the corresponding references). The other four classical approaches are: **(e) *Common Neighbors (CN)*** [16] only uses the number of common neighbors to measure the similarity between vertexes. **(f) *Jaccard's Coefficient (JA)*** [16] is a conventional similarity metric that aims to suppress the influence of large-degree nodes. **(g) *Adamic-Adar (AA)*** [16] is a variant of common neighbors, which assigns each neighbor a weight that is the reciprocal of the degree of the neighbor. This means that the more vertexes one vertex connected to, the less important it is on evaluating the proximity between a pair of vertex. **(h) *Pearson Correlation Coefficient (Pearson)*** [10] measures the extent to which there is a linear relationship between two nodes.

It is noticed that deep learning based methods first take input E^{train} to learn embedding vector φ_u for every node u. After node embedding, we use the embedding vectors of nodes to evaluate the likelihood of each potential link. Let φ_u and φ_v be the embedding vectors of nodes u and v respectively. The link likelihood function is defined as follows: For *Deepwalk*, *DRNE* and *ProNE*, we use the inner product operator to measure the similarity, i.e., $s(u, v) = \varphi_u \cdot \varphi_v$; for *Node2vec*, we use the off-shelve binary classification Logistic Regression algorithm of *sklearn* library to learn a model over the Hadamard product of the embeddings of the two nodes.

Note that all the parameter settings of baselines follow the default (or suggested) values in the original references. Please see the corresponding references for more details of the baseline methods.

Besides, for PNR framework, the size of rasterization grid is set to 50×50. For MLP model, we regard the existing links in the training set as positive examples and equal number of nonexisting links in the training set as negative samples. We manually tuned the hyper-parameters of MLP to reach the best performance, where the tuned values are given here: the hidden layer sizes are $(10, 20, 10)$, activation is *relu*, solver is *adam*, initial learning rate is 0.002, batch size is 256 and validation fraction is 0.2.

5.1.3 Evaluation Metrics. we adopt four commonly-used metrics to evaluate the performance of the proposed methods as follows:

- ***Precision@L***. It is an accuracy metric which gives equal weight to the returned instance:

$$Precision@L = \frac{1}{L}\sum_{i=1}^{L}\delta(i) \tag{8}$$

  where $\delta(i) = 1$ means the i^{th} pair is correctly predicted (i.e.the link exists in the test set), $\delta(i) = 0$ otherwise; L is the number of prediction list (of node pairs).
- ***Average Precision (AP)***. *AP* is concerned with the performance of the returned items ranked ahead. It is calculated as follows:

$$AP = \frac{1}{|E^{test}|}\sum_{j=1}^{n}(R_j \times \frac{I_j}{j}), \tag{9}$$

  where $n = |V|\times(|V|-1)/2-|E^{train}|$ is the number of nonexisting links in the training set; R_j is the relevant coefficient, $R_j = 1$ if the link j exists in the test set otherwise $R_j = 0$; I_j is the number of links existing in the test set out of the first j links.

5.2 Experimental Results

Figure 4 shows the *precision* results in ca-hepph dataset. We can see that our PNR combination method outperforms all of the other combination methods. It reveals that PNR could better combine two link prediction methods and precisely predict the potential links. Note that the deep learning based methods perform badly in the dataset, because deep learning based methods perform well in integrating auxiliary information, whereas only topological information is used. Maurizio et al. [7] even pointed out that little progress was made during the past few years based on deep learning. Similarly, Figure 5 reveals that PNR-based method also performs best in terms of *AP* metric.

From Fig. 4 (b), (d) and Fig. 5 (b), (d), it is obvious that weighted-plus, weighted-product and MLP schemes even perform worse than original methods in terms of both *precision* and *AP* metrics. However, the PNR scheme always achieves the best accuracy and outperforms any baseline methods. Note that MLP could theoretically imitate arbitrary function. However, due to limited size of samples, MLP usually has weak robustness and suffers the overfitting problem of hyper-parameters that influence its performance.

In order to explain why the PNR-based method has the best performance, Figure **??** shows the bivariate function PNR and the difference of the predicted lists. We see that PNR-based method could identify some particular links with low original scores by both methods. However, the classical combination schemes mostly predict links with high scores (at least one high score by either method). For these particular low-score links (the two scores of each link are both small by two methods), few nonexisting links have the same (or similar) cores, and thus most of them belongs to the real links. Hence, these links with some particular low scores are easily to be distinguished as real links based on PNR.

Combining the results of both Fig. 4 and Fig. 5, we conclude that our PNR outperforms MLP, weighted-plus and weighted-product combination schemes in most cases. We argue that the proposed method also wins in the other networks. Table 4 shows the detailed increase of our method than the baseline methods, where

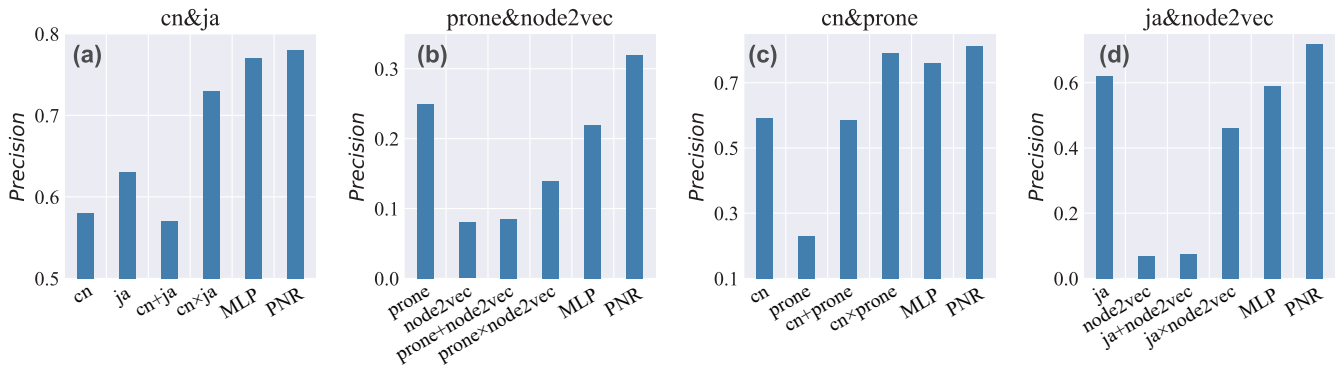


Figure 4: The precision results in ca-hepph dataset, which are experimented on combining four different methods: *CN, JA, node2vec and ProNE*. (a) CN and JA combination. (b) ProNE and node2vec combination. (c) CN and ProNE combination. (d) JA and node2vec combination.

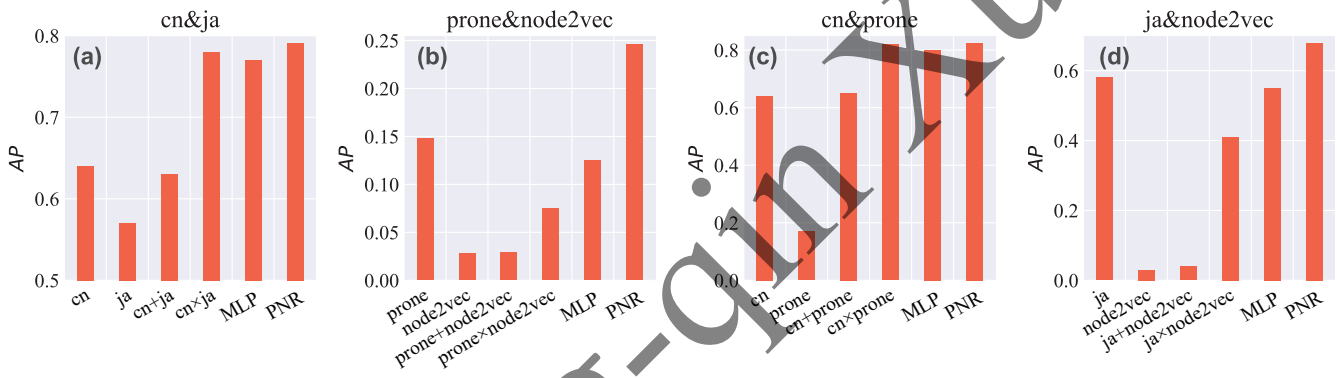


Figure 5: The AP(Average Precision) results in ca-hepph dataset, which are experimented on combining four different methods: *CN, JA, node2vec and ProNE*. (a) CN and JA combination. (b) ProNE and node2vec combination. (c) CN and ProNE combination. (d) JA and node2vec combination.

the increase characterizes how the PNR method enhances the performance, $p_{incre} = p_{PNR}/(max\{p_A, p_B\}) - 1$. We see that PNR combination method enhances the precision and AP in most cases. Note that the PNR method also has low time complexity, comparing with MLP that has hyper-parameters to tune and suffer high time complexity.

*Parameter Sensitivity:*Figure 6 shows the influence of the division ratio p^H on the precision metric in facebook dataset. We see that the proposed PNR method still outperforms the other baseline methods in most cases, showing the strong robustness of our method. Besides, other combinations in other networks also have similar performance, which are not shown here due to limited pages.

6 CONCLUSION

In this paper, we investigate the underlying mechanism of how to combine different link prediction methods, and then propose an optimal algorithm to combine different methods. We analytically prove the optimum of the PNR-based combination. The proposed PNR has three particular advantages: (1) PNR has higher prediction accuracy than the classical combination methods. (2) PNR is a general and optimal framework to combine various existing link prediction methods. (3) The PNR could be efficiently calculated only by counting the score difference of existing and nonexisting links, which does not improve the time complexity of the existing link prediction systems. Moreover, we conduct comprehensive experiments on real datasets, which validates the effectiveness and efficiency of the proposed framework. Thus, this work opens up a great deal of opportunities to design better link prediction systems only based on the existing link prediction approaches.

REFERENCES

[1] Gharbi Alshammari. 2018. *Hydra: A hybrid framework that improves collaborative filtering recommendation quality.* Ph.D. Dissertation. University of Brighton.
[2] Serpil Aslan and Mehmet Kaya. 2019. A Hybrid Recommendation System in Co-authorship Networks. In *2019 International Artificial Intelligence and Data Processing Symposium (IDAP).* IEEE, 1–5.
[3] Albert-László Barabási. 2016. *Network science.* Cambridge University Press.
[4] Nicola Barbieri, Francesco Bonchi, and Giuseppe Manco. 2014. Who to follow and why: link prediction with explanations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 1266–1275.
[5] Daniel Billsus and Michael J Pazzani. 2000. User modeling for adaptive news access. *User modeling and user-adapted interaction* 10, 2-3 (2000), 147–180.

Table 4: (Color online). The relative gain ratio(%) after combining two baselines (denoted by the corresponding row- and column-methods) with PNR framework in four datasets, $p_{incre} = p_{PNR}/(max\{p_A, p_B\}) - 1$. The upper-right elements (background green) are the relative gain ratio in terms of *precision* metric while the lower-left elements (background purple) are the relative gain ratio in terms of *AP* metric. (a) facebook. (b) ca-hepph. (c) yeast. (d) email-eucore.

A \ B	cn	ja	aa	pearson	deepwalk	node2vec	drne	prone
cn	--	19.24	17.64	17.84	14.71	12.77	1.12	16.73
ja	19.79	--	13.51	15.20	11.82	14.65	9.70	8.89
aa	16.82	12.78	--	12.94	10.06	9.90	1.16	13.12
pearson	17.50	28.06	10.89	--	14.74	19.83	12.13	13.29
deepwalk	15.30	23.63	9.83	32.41	--	-6.90	27.51	46.73
node2vec	12.41	28.61	8.97	39.36	61.51	--	47.24	27.92
drne	5.03	18.93	-0.22	26.79	38.22	37.86	--	23.06
prone	15.00	18.14	8.91	27.64	67.95	46.05	29.70	--

(a) Facebook

A \ B	cn	ja	aa	pearson	deepwalk	node2vec	drne	prone
cn	--	24.91	17.92	24.86	22.56	29.22	1.29	38.62
ja	23.48	--	15.31	16.16	13.75	10.42	4.67	8.68
aa	10.20	8.36	--	17.17	15.48	17.51	1.84	23.59
pearson	23.18	30.96	9.28	--	16.60	12.01	5.30	9.86
deepwalk	15.15	29.63	7.12	34.52	--	28.24	-3.14	48.48
node2vec	17.90	25.71	9.52	29.19	46.32	--	-0.54	32.92
drne	1.65	21.32	1.89	23.29	23.11	-8.18	--	2.91
prone	29.04	23.45	11.74	25.05	127.30	67.58	8.65	--

(b) ca-hepph

A \ B	cn	ja	aa	pearson	deepwalk	node2vec	drne	prone
cn	--	57.84	29.52	57.05	35.58	26.02	3.13	45.14
ja	61.41	--	36.05	229.20	242.58	221.50	195.70	129.70
aa	30.46	46.47	--	30.48	16.19	15.65	-3.27	21.90
pearson	55.93	186.40	38.70	--	381.06	356.80	306.10	123.90
deepwalk	38.49	170.10	13.16	143.00	--	14.96	77.46	121.80
node2vec	20.12	147.40	7.54	137.70	125.73	--	-0.82	75.10
drne	1.06	114.50	-6.83	97.09	21.13	9.91	--	33.86
prone	23.77	118.30	10.94	108.80	149.21	113.30	43.31	--

(c) Yeast

A \ B	cn	ja	aa	pearson	deepwalk	node2vec	drne	prone
cn	--	10.91	3.63	7.91	15.72	15.73	5.27	10.59
ja	11.40	--	3.04	9.58	2.40	-2.99	0.96	-18.20
aa	0.53	2.23	--	2.76	8.63	7.65	6.00	6.86
pearson	10.29	28.16	2.59	--	6.84	2.13	9.42	-21.43
deepwalk	23.63	5.84	9.37	5.79	--	18.33	47.93	33.16
node2vec	20.60	-1.84	7.38	4.29	6.57	--	36.84	3.95
drne	9.04	8.65	7.86	11.20	17.75	30.73	--	3.89
prone	19.21	-17.81	5.30	3.53	53.97	12.79	6.86	--

(d) email-eucore

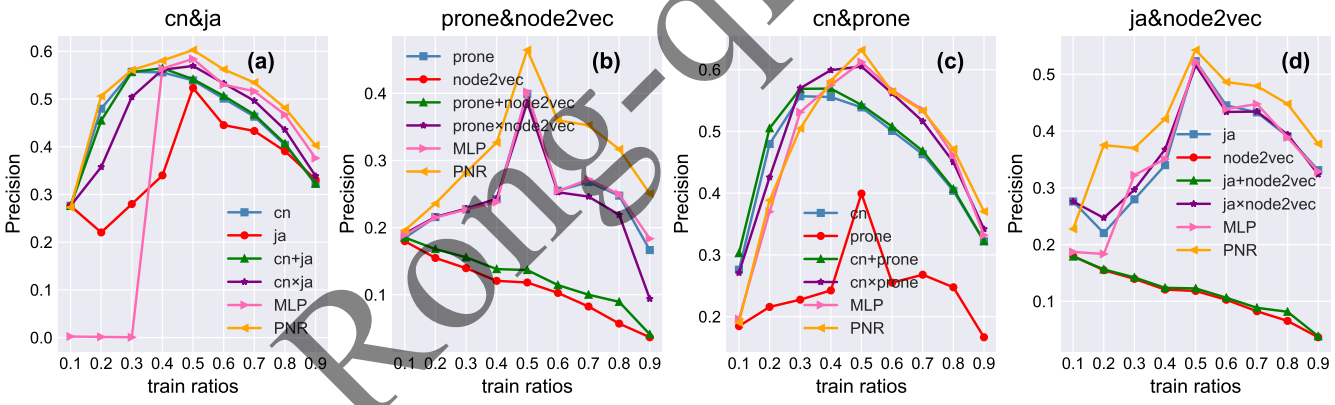


Figure 6: The *precision* performance in facebook dataset under different ratios of training set, which are experimented on combining four different methods: *CN, JA, node2vec and ProNE*. (a) CN and JA combination. (b) ProNE and node2vec combination. (c) CN and ProNE combination. (d) JA and node2vec combination.

[6] Matthias Bogaert, Michel Ballings, and Dirk Van den Poel. 2016. The added value of Facebook friends data in event attendance prediction. *Decision Support Systems* 82 (2016), 26–34.

[7] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. 2019. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 101–109.

[8] Mustansar Ghazanfar and Adam Prugel-Bennett. 2010. An improved switching hybrid recommender system using naive bayes classifier and collaborative filtering. (2010).

[9] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 855–864.

[10] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. 2004. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)* 22, 1 (2004), 5–53.

[11] Krishna Kamath, Aneesh Sharma, Dong Wang, and Zhijun Yin. 2014. Realgraph: User interaction prediction at twitter. In *user engagement optimization workshop@ KDD*.

[12] Pigi Kouki, Shobeir Fakhraei, James Foulds, Magdalini Eirinaki, and Lise Getoor. 2015. Hyper: A flexible and extensible probabilistic framework for hybrid recommender systems. In *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 99–106.

[13] István A Kovács, Katja Luck, Kerstin Spirohn, Yang Wang, Carl Pollis, Sadie Schlabach, Wenting Bian, Dae-Kyum Kim, Nishka Kishore, Tong Hao, et al. 2019. Network-based prediction of protein interactions. *Nature communications* 10, 1 (2019), 1240.

[14] Jérôme Kunegis. 2013. Konect: the koblenz network collection. In *Proceedings of the 22nd International Conference on World Wide Web*. ACM, 1343–1350.

[15] Jure Leskovec and Andrej Krevl. 2016. SNAP datasets: Stanford large network dataset collection (2014). *URL http://snap. stanford. edu/data* (2016), 49.

[16] David Liben-Nowell and Jon Kleinberg. 2007. The link-prediction problem for social networks. *Journal of the Association for Information Science and Technology* 58, 7 (2007), 1019–1031.

[17] Ryan N Lichtenwalter, Jake T Lussier, and Nitesh V Chawla. 2010. New perspectives and methods in link prediction. In *16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 243–252.

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

[19] Xiaoke Ma, Penggang Sun, and Guimin Qin. 2017. Nonnegative matrix factorization algorithms for link prediction in temporal networks using graph communicability. *Pattern Recognition* 71 (2017), 361–374.

[20] Víctor Martínez, Fernando Berzal, and Juan-Carlos Cubero. 2016. A Survey of Link Prediction in Complex Networks. *ACM Computing Surveys (CSUR)* 49, 4 (2016), 69.

[21] Babita Pandey, Praveen Kumar Bhanodia, Aditya Khamparia, and Devendra Kumar Pandey. 2019. A comprehensive survey of edge prediction in social networks: Techniques, parameters and challenges. *Expert Systems with Applications* (2019).

[22] Michael J Pazzani. 1999. A framework for collaborative, content-based and demographic filtering. *Artificial intelligence review* 13, 5-6 (1999), 393–408.

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

[24] Ryan A. Rossi and Nesreen K. Ahmed. 2015. The Network Data Repository with Interactive Graph Analytics and Visualization. In *AAAI*. http://networkrepository.com

[25] Jiliang Tang, Yi Chang, and Huan Liu. 2014. Mining social media with social theories: a survey. *ACM Sigkdd Explorations Newsletter* 15, 2 (2014), 20–29.

[26] Ke Tu, Peng Cui, Xiao Wang, Philip S Yu, and Wenwu Zhu. 2018. Deep recursive network embedding with regular equivalence. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2357–2366.

[27] Shanshan Wan and Zhendong Niu. 2019. A Hybrid E-learning Recommendation Approach Based on Learners' Influence Propagation. *IEEE Transactions on Knowledge and Data Engineering* (2019).

[28] Hongwei Wang, Fuzheng Zhang, Min Hou, Xing Xie, Minyi Guo, and Qi Liu. 2018. Shine: Signed heterogeneous information network embedding for sentiment link prediction. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 592–600.

[29] Pinghua Xu, Wenbin Hu, Jia Wu, and Bo Du. 2019. Link Prediction with Signed Latent Factors in Signed Social Networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1046–1054.

[30] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 974–983.

[31] Jie Zhang, Yuxiao Dong, Yan Wang, Jie Tang, and Ming Ding. 2019. ProNE: fast and scalable network representation learning. In *Proc. 28th Int. Joint Conf. Artif. Intell., IJCAI*. 4278–4284.

[32] Peng Zhang, Xiang Wang, Futian Wang, An Zeng, and Jinghua Xiao. 2016. Measuring the robustness of link prediction algorithms under noisy environment. *Sci. Rep.* 6 (2016), 18881.

[33] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)* 52, 1 (2019), 5.

A PROOF OF THE OPTIMAL PNR

In this section, we prove the optimal bivariate function $F_{opt}(x_1, x_2) = PNR(x_1, x_2)$ and provide its relationship with *precision*. The proof is irrelevant to the design of the algorithm and the experiments. Thus, please skip the proof if you care more about the experiment performance and it would not influence the reading of the paper.

In the proposed framework, we use the hybrid score $s'_{u,i} = f(s^{(A)}_{u,i}, s^{(B)}_{u,i}) = PNR(s^{(A)}_{u,i}, s^{(B)}_{u,i})$ to predict links. The scores of the predicted links constitute the optimal score set S^f in eq. 7. We argue that any other bivariate function $s''_{u,i} = f'(s^{(A)}_{u,i}, s^{(B)}_{u,i})$ cannot outperform the performance of $s'_{u,i} = PNR(s^{(A)}_{u,i}, s^{(B)}_{u,i})$. We take two cases to prove the optimal PNR.

Case 1. Supposing that $S^{f'}$ is the score set satisfying $S^f \cap S^{f'} = \phi$, where S^f and $S^{f'}$ are the score sets obtained by $f(x_1, x_2) = PNR(x_1, x_2)$ and an arbitrary different function $f'(x_1, x_2)$ respectively. Since the length of prediction list is L, $(U - |E^{train}|) \cdot \oint_{S^{f'}} p_n(x_1, x_2) dx_1 dx_2 = L$, substituting $\int_{S^{f'}} p_n(x_1, x_2) dx_1 dx_2 = L/(|U| - |E^{train}|)$ into eq. 6, we have

$$\begin{aligned} precision^{S^{f'}} &= \alpha \oint_{S^{f'}} p_r(x_1, x_2) dx_1 dx_2 \\ &= \alpha \oint_{S^{f'}} PNR(x_1, x_2) \cdot p_n(x_1, x_2) dx_1 dx_2 \\ &< \alpha PNR_{max}(S^{f'}) \cdot \oint_{S^{f'}} p_n(x_1, x_2) dx_1 dx_2 \\ &= \gamma \cdot PNR_{max}(S^{f'}), \end{aligned} \tag{10}$$

where $\alpha = \frac{\gamma(|U| - |E^{train}|)}{L}$, and $PNR_{max}(S^{f'})$ is the maximum value of $PNR(x_1, x_2), \forall (x_1, x_2) \in S^{f'}$.

Analogously, the *precision* of the optimal set S^f follows

$$\begin{aligned} precision^{S^f} &= \alpha \oint_{S^f} PNR(x_1, x_2) \cdot p_n(x_1, x_2) dx_1 dx_2 \\ &> \alpha PNR_{min}(S^f) \cdot \oint_{S^f} p_n(x_1, x_2) dx_1 dx_2 \\ &= \gamma \cdot PNR_{min}(S^f). \end{aligned} \tag{11}$$

Note that $RNR(x_1, x_2) > RNR(x'_1, x'_2), \forall (x_1, x_2) \in S^f, \forall (x'_1, x'_2) \in \mathrm{R} - S^f$, and $S^{f'} \subset \mathrm{R} - S^f$. Therefore, $PNR_{min}(S^f) > PNR_{max}(S^{f'})$ and we arrive at

$$precision^{S^f} > precision^{S^{f'}}. \tag{12}$$

Case 2. Supposing $S^f \cap S^{f'} = Z \neq \phi$ and $\oint_Z p_n(x_1, x_2) dZ = C_z$, according to the procedure introduced above, we can also get

$$\begin{aligned} \oint_{S^f \setminus Z} p_r(x_1, x_2) dx_1 dx_2 &= \oint_{S^f \setminus Z} PNR(x_1, x_2) \cdot p_n(x_1, x_2) dx_1 dx_2 \\ &> PNR_{min}(S^f \setminus Z) \cdot \oint_{S^f \setminus Z} p_n(x_1, x_2) dx_1 dx_2 \\ &> PNR_{max}(S^{f'} \setminus Z) \cdot \oint_{S^{f'} \setminus Z} p_n(x_1, x_2) dx_1 dx_2 \\ &> \oint_{S^{f'} \setminus Z} PNR(x_1, x_2) \cdot p_n(x_1, x_2) dx_1 dx_2 \\ &= \oint_{S^{f'} \setminus Z} p_r(x_1, x_2) dx_1 dx_2. \end{aligned} \tag{13}$$

Substituting eq. 13 into eq. 6, we get

$$\begin{aligned} precision^{S^f} &= \alpha \cdot \oint_{S^f \setminus Z} p_r(x_1, x_2) dx_1 dx_2 + \alpha \cdot \int_Z p_r(x_1, x_2) dx_1 dx_2 \\ &> \alpha \cdot \int_{S^{f'} \setminus Z} p_r(x_1, x_2) dx_1 dx_2 + \alpha \cdot \int_Z p_r(x_1, x_2) dx_1 dx_2 \\ &= precision^{S^{f'}}. \end{aligned} \tag{14}$$

Therefore, we can conclude that none bivariate function could outperform the $PNR(x_1, x_2)$. Hence, the proposed PNR is the optimal bivariate function to combine different link prediction methods.

B SUPPLEMENTARY RESULTS

During to the limited pages of main body, we provide some key experimental results in this section. Since the proposed method significantly improve the prediction accuracy, we provide the *precision* results on the other three datasets. Figure 7 and Figure 8 show the *precision* comparison results on facebook and yeast datasets respectively.

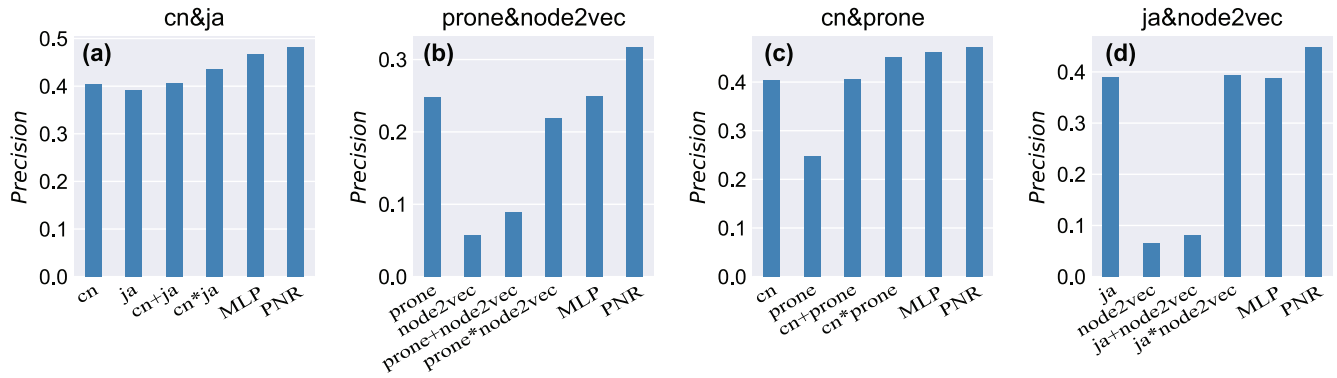


Figure 7: The precision results in facebook dataset, which are experimented on combining four different methods: *CN, JA, node2vec and ProNE*. (a) CN and JA combination. (b) ProNE and node2vec combination. (c) CN and ProNE combination. (d) JA and node2vec combination.

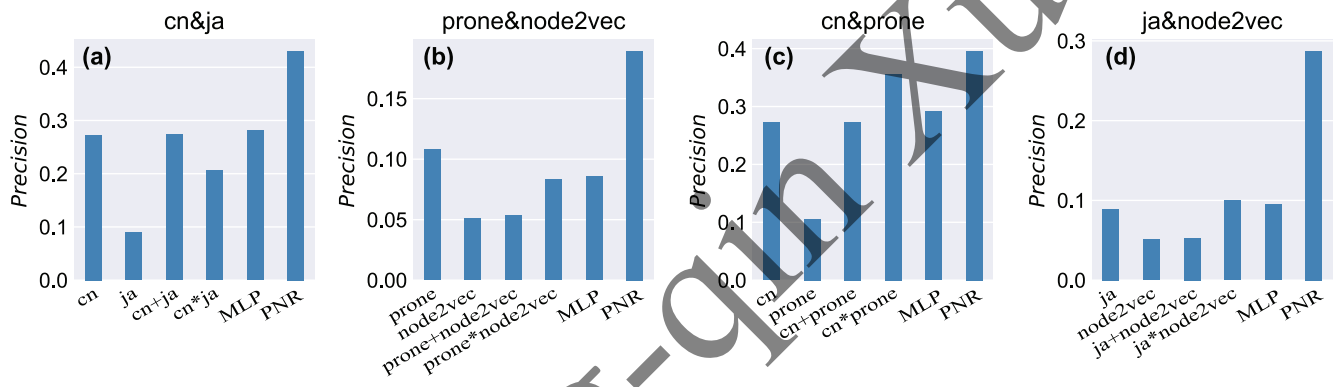


Figure 8: The precision results in yeast dataset, which are experimented on combining four different methods: *CN, JA, node2vec and ProNE*. (a) CN and JA combination. (b) ProNE and node2vec combination. (c) CN and ProNE combination. (d) JA and node2vec combination.