Multi-level Hyperedge Distillation for Social Linking Prediction on Sparsely Observed Networks

Xiangguo Sun
Southeast University
China
sunxiangguo@seu.edu.cn

Hongzhi Yin
The University of Queensland
Australia
h.yin1@uq.edu.au

Bo Liu*
Southeast University
China
bliu@seu.edu.cn

Hongxu Chen
University of Technology Sydney
Australia
hongxu.chen@uts.edu.au

Qing Meng
Southeast University
China
qmeng@seu.edu.cn

Wang Han, Jiuxin Cao
Southeast University
China
{hanwn,jx.cao}@seu.edu.cn

ABSTRACT

Social linking prediction is one of the most fundamental problems in online social networks and has attracted researchers' persistent attention. Most of the existing works predict unobserved links using graph neural networks (GNNs) to learn node embeddings upon pair-wise relations. Despite promising results given enough observed links, these models are still challenging to achieve heart-stirring performance when observed links are extremely limited. The main reason is that they only focus on the smoothness of node representations on pair-wise relations. Unfortunately, this assumption may fall when the networks do not have enough observed links to support it. To this end, we go beyond pair-wise relations and propose a new and novel framework using hypergraph neural networks with multi-level hyperedge distillation strategies. To break through the limitations of sparsely observed links, we introduce the hypergraph to uncover higher-level relations, which is exceptionally crucial to deduce unobserved links. A hypergraph allows one edge to connect multiple nodes, making it easier to learn better higher-level relations for link prediction. To overcome the restrictions of manually designed hypergraphs, which is constant in most hypergraph researches, we propose a new method to learn high-quality hyperedges using three novel hyperedges distillation strategies automatically. The generated hyperedges are hierarchical and follow the power-law distribution, which can significantly improve the link prediction performance. To predict unobserved links, we present a novel hypergraph neural networks named HNN. HNN takes the multi-level hypergraphs as input and makes the node embeddings smooth on hyperedges instead of pair-wise links only. Extensive evaluations on four real-world datasets demonstrate our model's superior performance over state-of-the-art baselines, especially when the observed links are extremely reduced.

*Corresponding author

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3449912

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**.

KEYWORDS

hypergraph learning, sparsely observed networks, linking prediction

ACM Reference Format:
Xiangguo Sun, Hongzhi Yin, Bo Liu, Hongxu Chen, Qing Meng, and Wang Han, Jiuxin Cao. 2021. Multi-level Hyperedge Distillation for Social Linking Prediction on Sparsely Observed Networks. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3442381.3449912

1 INTRODUCTION

With the popularisation of online networks such as the world wide web, citation networks, and social platforms, people transfer their social life mostly online, thus resulting in considerable heterogeneous social interactions. Predicting these social relations plays a fundamental role in analyzing users' online behaviors and social phenomena [34, 42]. It has been widely used in recommendation systems [6, 13], anomaly detection [19], and sentiment analysis [32]. Formally, social linking prediction aims to distinguish whether a pair of nodes in a network has a specific type of link or not. Although researchers have paid much attention [4, 30, 40], there is still significant potential for improvement, especially when the observed links are extremely limited, which is universal in real-world situations [45].

Previous studies usually leverage network sampling strategies [26] to make node embeddings smooth on pair-wise links. For example, Grover et al. [12] propose a bias random walk framework and then use the Skip-gram model to maximize the probability of target node given specific contents. Many variants such as metapath2vec [9], HHNE [37], and HeteSpaceyWalk [17] come out recently to handle heterogeneous relations following this idea. However, converting graphs to linear paths limits these models' further improvement because these paths can not reconstruct the original graphs without missing information. To this end, graph neural networks (GNNs) [39] have been recently introduced to this subject because they have effectively addressed non-linear relations in graphs. Specifically, Wang et al. propose HAN [35], which integrates node-level attention and semantic-level attention to learn node embeddings

from heterogeneous graphs. Hu et al. [18] leverage generative adversarial networks (GANs) to learn node distribution to generate better negative samples for node embedding. Fu et al. [11] optimise GNN-based methods with intra-metapath aggregation and inter-metapath aggregation and make the model flexible. Although performance has been further improved, these models heavily rely on the network connectivity, which means the node smoothness may be destroyed when the observed links are extremely limited.

To improve linking prediction performance on sparsely observed networks, we need to explore higher-level relations to support underlying links further. Most of the existing works learn higher-level relations from ordinary local proximities. For example, Chen et al. [5] put forward PME, which integrates first-order and second-order proximities via metric learning. They first project node features to different semantic spaces, and then tighten each pair of nodes on pair-wise links. Similarly, Lu et al. [44] present RHINE, and they utilise different models to exploit affiliation and interaction relations. Wang et al. [36] use their proposed model StHNE to learn meta-path based first-order and second-order proximities, and then optimise the distance of similar nodes with spectral theory. However, all these works learn high-order relations via minimising the distance of similar nodes and maximising dissimilar nodes on observed pair-wise links, which is far from expressive on sparsely observed networks. Taking Figure 1 as an example, when we predict links only based on pair-wise relations, the model may easily treats unobserved positive links as a negative node pair, making this pair of nodes weakly correlated and the node embeddings are under-smooth.

Unlike typical graph models, which only focus on pair-wise relations, hypergraphs [31] can capture and preserve more diverse, complicated, and higher-level semantics. Hypergraphs allow one edge (a.k.a hyperedge) to connect multiple nodes, which are perfectly suitable for heterogeneous networks (HINs) and naturally promising to improve the link prediction performance on HINs. However, in many graph-structured datasets, hyperedges are not always given. To apply hypergraph models, most of the existing works have to assign hyperedges manually. For instance, Feng et al. [10] design a hypergraph neural network via analogy with normal graphs. They build hyperedges by calculating the distance between nodes features. Zhang et al. [47] treat hyperedges as node feature tuples and use the attention mechanism to fuse information within a hyperedge. Jin et al. [21] propose a convolutional manifold network guided by manually designed hyperedges. They construct their hypergraphs with k-nearest nodes. Chen et al. [4] put forward MGCN, and they discuss the impact of different manually created hypergraphs. Jiang et al. [20] and Zhang et al. [48] study dynamic networks using hypergraph learning. They treat hyperedges as clusters and network neighbours. Although these works achieve satisfactory results on various downstream tasks, they still suffer from the limitation of toneless hyperedges, leading to over-smooth expressiveness for pair-wise linking prediction. As shown in the right of Figure 1, nodes in a toneless hyperedge can be also converted as a complete graph, where each pair of nodes are tightened no matter their real connections. This may easily make node embeddings over-smooth [8], leading to the opposite corner compared with the models only based on pair-wise relations.

From the above discussion, we have realised that the vital barrier for linking prediction on a sparsely observed network is how to learn multi-level hyperedges automatically from data. To fill the research gap, we need to solve the following challenges:

- Challenge 1: How to learn better higher-level relations on sparsely observed networks. Traditional methods mostly rely on pair-wise links heavily, and they usually follow the idea that each pair of nodes should be closer on an observed link but keep away from each other if there is no observed link connecting them. However, when the observed links are dramatically reduced, the smoothness can not be guaranteed anymore, limiting the performance improvement.
- Challenge 2: How to learn multi-level hyperedges automatically from data. Most of related works use manually designed hyperedges. However, these hyperedges are far from hierarchical and flexible, resulting in over-smooth problems for linking prediction. To solve this problem, we need to break away from constant hyperedges and generate multi-level hyperedges automatically from data.
- Challenge 3: How to learn better node embeddings for linking inference. Traditional works usually use matrix decomposition and GNNs on typical graphs, which can not be directly applied to hypergraphs. To integrate hyperedges information, we need to design a uniform model from both pair-wise links and hyperedges.

To address challenge 1, we go beyond limited observed pair-wise links and use the hypergraph to learn latent higher-level information. To address challenge 2, we present three delicate hyperedge expansion strategies to distill multi-level hyperedges. The hyperedges are generated automatically, follow the long-tailed distribution, and significantly improve link prediction performance. To address challenge 3, we design a multi-level hypergraph neural network for heterogeneous graphs, which can take both pair-wise links and hyperedges together. In summary, our principal contributions are as follows:

- We propose a novel hyperedge generation framework using three well-designed hyperedge expansion strategies. The hyperedges start from basic graphlets and then expand themselves automatically. The generated hyperedges follow the power-law distribution and outperform previous manually designed hypergraphs on network embeddings.
- We focus on sparsely observed networks when predicting links, which is more common in practical applications. To overcome the limited links, we propose to leverage hypergraphs to learn better higher-level informations and put forward a multi-level hypergraph neural network, which can dramatically improve the performance.
- We extensively evaluate our approach with state-of-the-art baselines on four real-world datasets. Experimental results demonstrate that our method can achieve significant improvements over existing methods.

2 PRELIMINARY AND PROBLEM FORMULATION

In this section, we briefly present related concepts and give the formal definition of our target problem.

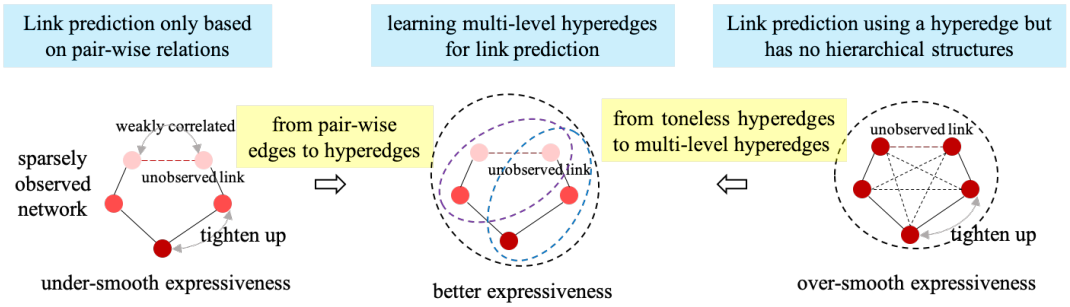


Figure 1: pair-wise relations (left), multi-level hyperedges (middle), and toneless hyperedges (right).

Definition 1. ***(Heterogeneous Networks and Meta-path)***. *A heterogeneous network (HIN) refers to a graph like $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{T}_v\}$ where $\mathcal{V}$ denotes the objects, and $\mathcal{E}$ is the pair-wise relations between these objects. Each object is treated as a node with a specific type, and $\mathcal{T}_v$ denotes all node types. When $|\mathcal{T}_v| \geq 2$, the network becomes heterogeneous. A meta-path is a pre-defined path scheme where each position on the path has one assigned node type.*

This paper studies three heterogeneous networks (DBLP, ACM, Yelp), and one homogeneous network (Cora). We treat the homogeneous network as a special case of HINs, which means it only contains one type of relation and the same kind of nodes. More details on DBLP, ACM, Yelp, and Cora are summarised in section 4.1.

Definition 2. ***(Graphlets and Orbits)***. *The graphlets of a given large network are a set of smaller connected induced subgraphs, and each graphlet is structurally distinct from all the other graphlets. Nodes within the same graphlet and topologically identical with others are put into the same orbits.*

Take Figure 2 as an illustration, we present all graphlets with 2,3,4, and 5 nodes. There are 30 graphlets in total. For simplicity, we only show these graphlets in a homogeneous graph. We can use meta-path to extract different induced subgraphs and find graphlets independently for the graph with multiple node types.

Definition 3. ***(Hypergraph)***. *A hypergraph can be represented as $\mathcal{G}^{\triangleright} = \{\mathcal{V}^{\triangleright}, \mathcal{E}^{\triangleright}\}$. Here $\mathcal{V}^{\triangleright}$ denotes node set, and $\mathcal{E}^{\triangleright}$ is the set of edges in the hypergraph (a.k.a hyperedges). Different from normal graphs, a hypergraph allows one hyperedge to connect multiple nodes, which means each hyperedge can be denoted as a subset of nodes $e^{\triangleright} = \{v_1, v_2, \cdots, v_k\}, v_i \in \mathcal{V}^{\triangleright}, e^{\triangleright} \in \mathcal{E}^{\triangleright}$. k is the size of hyperedge $e^{\triangleright}$.*

The presence of nodes in hyperedges can be represented as an incidence matrix $\mathbf{H} \in \{0,1\}^{|\mathcal{V}^{\triangleright}| \times |\mathcal{E}^{\triangleright}|}$ where each entry can be calculated as follows:

$$\mathbf{H}(i,j) = \begin{cases} 1, & \text{if node } i \text{ is in hyperedge } j \\ 0, & \text{otherwise} \end{cases}$$

Let $\mathbf{D}_v \in \mathbb{R}^{|\mathcal{V}^{\triangleright}| \times |\mathcal{V}^{\triangleright}|}$ and $\mathbf{D}_e \in \mathbb{R}^{|\mathcal{E}^{\triangleright}| \times |\mathcal{E}^{\triangleright}|}$ be two diagonal matrices, which denote the degrees of nodes and hyperedges, respectively.

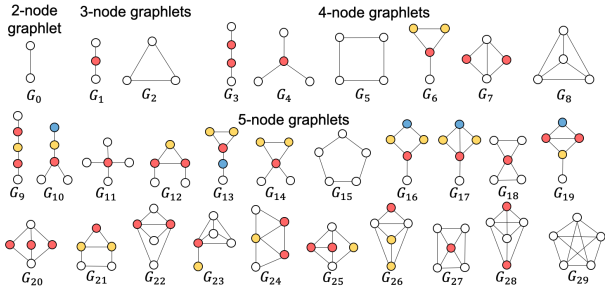


Figure 2: 2,3,4,5-node graphlets. There are 30 graphlets denoted by $G_i, i = 0, \cdots, 29$. In each graphlet, we use different colors to denote different orbits. Nodes in the same orbit are topologically identical.

Then the degree of node i is defined as follows:

$$\mathbf{D}_v(i,i) = \sum_{j=1}^{|\mathcal{E}^{\triangleright}|} \mathbf{U}_e(j,j) \cdot \mathbf{H}(i,j)$$

where $\mathbf{U}_e \in \mathbb{R}^{|\mathcal{E}^{\triangleright}| \times |\mathcal{E}^{\triangleright}|}$ is a diagonal matrix, and the diagonal entries are hyperedge weights. Likewise, the degree of hyperedge j is defined as follows:

$$\mathbf{D}_e(j,j) = \sum_{i=1}^{|\mathcal{V}^{\triangleright}|} \mathbf{U}_v(i,i) \cdot \mathbf{H}(i,j)$$

where each diagonal entry in the diagonal matrix $\mathbf{U}_v \in \mathbb{R}^{|\mathcal{V}^{\triangleright}| \times |\mathcal{V}^{\triangleright}|}$ stands for the weights of nodes.

Definition 4. ***(Hyperedge Distillation)*** *We define the hyperedge distillation as the process of generating multi-level hyperedges automatically from data. The process starts from some basic graphlets and then expands small-scaled hyperedges to develop multi-level hyperedges.*

With the above concepts, our target problem can be formulated as follows:

Problem 1. ***(Social Linking Prediction).*** *Given a sparsely observed network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{T}_v\}$, and any pair of nodes $(v_i, v_j), v_i, v_j \in \mathcal{V}$, we wish to predict whether (v_i, v_j) has a given type of pair-wise relation or not, which is not observed before.*

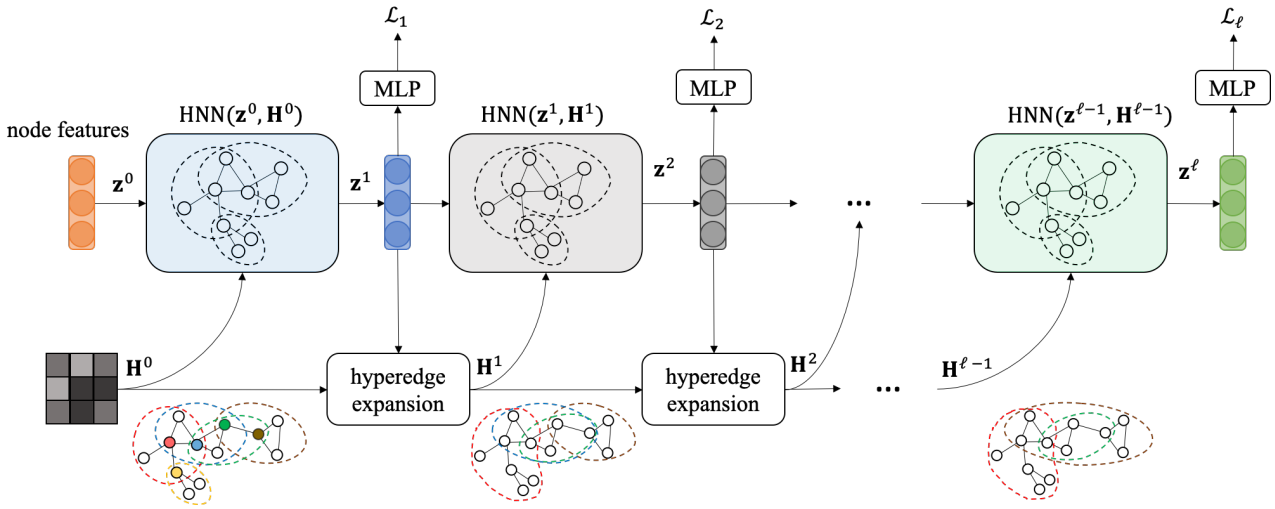


Figure 3: The flowchart of our framework.

3 HYPERGRAPH LEARNING WITH MULTI-LEVEL EXPANSION

In this section, we first review our framework of social linking prediction, and then introduce the principal components of our model.

3.1 Model Overview

Our model framework is shown in Figure 3. The model is a stack of multi-level hypergraph neural networks (HNNs), and each HNN layer utilises the corresponding hypergraph, which is updated from the previous state. The initial hyperedges are built on graphlets shown in Figure 2. Afterwards, we design three hyperedge expansion strategies (depth-first expansion, breadth-first expansion, and hybrid expansion) to expand these simple hyperedges as higher-level hyperedges. In the end, we output the final network embeddings and the hypergraph incidence matrix, which can hint the hypergraph structure after multi-level hyperedge distillation. To predict social links, we concatenate each layer's latent representations and then send them to the downstream link prediction classifier.

3.2 Graphlets-driven Hyperedges

As previously discussed, our first target is how to generate multi-level hyperedges. Here we introduce graphlets as the initial hyperedges to launch later hyperedge distillation. We choose graphlets because they are structurally complete, and each graphlet preserves an exclusive structural unit. They go beyond pair-wise relations, which means they are informative for link prediction on sparsely observed networks. However, they are not sufficient to preserve global structures and thus make the performance limit. Besides, a larger network usually contains enormous graphlets, leading to a large number of hyperedges. At last, graphlets are still not hierarchical enough owing to their small number of nodes.

In light of these, we initialise our hyperedges with only 2,3,4,5-node graphlets and ignore larger graphlets. For heterogeneous networks, we induce subgraphs with different meta-paths and then combine the graphlets in each subgraph. There are plenty of implements and algorithms [1, 28, 33] that can efficiently find graphlets, making our initialisation entirely feasible. Each graphlet instance is treated as an initial hyperedge. Then we design three hyperedge expansion strategies (depth-first expansion, breadth-first expansion, and hybrid expansion) so that these hyperedges can be expanded or merged with other hyperedges, making the total hyperedge number reduced and hyperedge multi-level. We elaborate on this in the following section.

3.3 Hyperedge Expansion

As shown in Figure 4, a hyperedge usually adjoins other hyperedges with some shared nodes, making the hyperedge expansion executable. The transition probability from one hyperedge to another relies on two aspects: (i) the correlations of hyperedges, which can be measured by the product of hyperedge representations; and (ii) the connectivity between two hyperedges, which can be evaluated by the Jaccard similarity.

Let the observed network be $\mathcal{G}^\circ = \{\mathcal{V}, \mathcal{E}^\circ\}$ where $\mathcal{V}$ is node set and $\mathcal{E}^\circ$ contains observed pair-wise links. The initial hyperedge set is $\mathcal{E}_0^\triangleright = \{e_1, e_2, \cdots, e_n\}$ where each hyperedge includes a set of nodes $e_i = \{v_{i_1}, v_{i_2}, \cdots, v_{i_m}\}$. Let node representation be $\mathbf{Z}_v \in \mathbb{R}^{|\mathcal{V}| \times d}$ where d is node embedding dimension, then we denote hyperedges via the weighted summation over all nodes in the same hyperedge:

$$\mathbf{Z}_e = \mathbf{D}_e^{-1} \cdot \mathbf{U}_e \cdot \mathbf{H}^{\mathrm{T}} \cdot \mathbf{U}_v \cdot \mathbf{Z}_v \tag{1}$$

With the above formula, the correlation between hyperedges i and j can be calculated as :

$$\alpha(i,j) = \sigma\left(\mathbf{Z}_e^{\mathrm{T}} \cdot \mathbf{Z}_e\right)_{ij} \tag{2}$$

where $\sigma(\cdot)$ is a normalization operator such as the sigmoid function. To measure the connectivity between hyperedges i and j, we

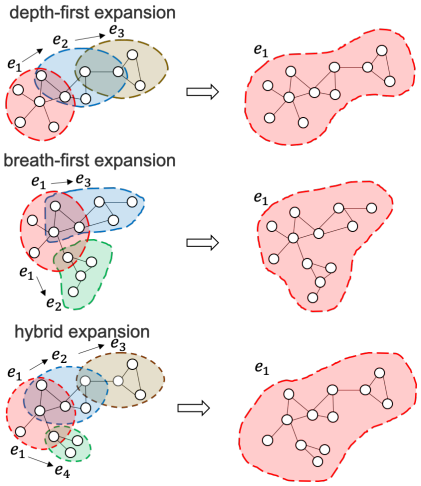


Figure 4: Three hyperedge expansion strategies. top: depth-first expansion; middle: breadth-first expansion; bottom: hybrid expansion.

calculate the Jaccard similarity of i and j as follows:

$$\beta(i,j) = \frac{\sum_{k=1}^{|\mathcal{V}|} \mathbf{H}(k,i) \cdot \mathbf{H}(k,j)}{\sum_{p=1}^{|\mathcal{V}|} \mathbf{H}(p,i) + \sum_{q=1}^{|\mathcal{V}|} \mathbf{H}(q,j)} \tag{3}$$

With the above definitions, we define the transition probability from hyperedge i to hyperedge j as follows:

$$p(j|i) = \frac{\alpha(i,j) + \beta(i,j)}{\sum_{k \in \mathcal{N}_i} (\alpha(i,k) + \beta(i,k))} \tag{4}$$

where $\mathcal{N}_i$ denote all hyperedges sharing nodes with hyperedge i. Based on formula (4), we present three novel hyperedge expansion strategies to distil multi-level hyperedges: depth-first expansion, breadth-first expansion, and hybrid expansion.

3.3.1 Depth-First Expansion. Depth-first expansion follows the idea that a hyperedge has the momentum to accommodate other hyperedges along a hyperedge path without revisiting the same hyperedges. As shown in the top of Figure 4, there exists a hyperedge path starting from e_1 and ending at e_3, which can be denoted as: $e_1 \rightarrow e_2 \rightarrow e_3$. When we expand e_1, it will first choose one adjacent hyperedge such as e_2 with the corresponding transition probability, then it continues to select the next hyperedge from the neighbours of e_2 which has never been visited before (such as e_3). We use p_a ($0 \leq p_a \leq 1$) to represent how aggressive the depth-first expansion wants to continue. At each step, the process has the probability of p_a to keep going or stop with the possibility of $1 - p_a$. More details on the depth-first expansion can be seen in Algorithm 1. Note that each hyperedge can be expanded independently because they only rely on the initial hyperedge set $\mathcal{E}_0^{\triangleright}$, thus the depth-first expansion algorithm can be efficiently conducted in parallel.

3.3.2 Breadth-First Expansion. Breadth-First Expansion first explores a sampled neighbours of the target hyperedge e_i, then adds

Algorithm 1: Depth-First Expansion Algorithm

Input: aggressive parameter p_a; initial hyperedges $\mathcal{E}_0^{\triangleright}$; hyperedge's adjacent set $\{\mathcal{N}_1, \cdots, \mathcal{N}_n\}$;
Output: updated hyperedges $\mathcal{E}_1^{\triangleright} = \{e_1, e_2, \cdots, e_m\}$;

$\mathcal{E}_1^{\triangleright} = \emptyset$.
for $e_i \in \mathcal{E}_0^{\triangleright}$ **do**
  $\mathcal{P} : e_{i1} \rightarrow e_{i2} \rightarrow \cdots e_{it}, e_{i1} \in \mathcal{N}_{e_i}$
  //generate a hyperedge random walk path using formula (4) with the stop probability $1 - p_a$.
  $e_i^{\triangleright} = e_i \cup e_{i1} \cup e_{i2} \cup \cdots e_{it}$
  $\mathcal{E}_1^{\triangleright} = \mathcal{E}_1^{\triangleright} \cup e_i^{\triangleright}$.
end
remove duplicated hyperedges in $\mathcal{E}_1^{\triangleright}$.
return $\mathcal{E}_1^{\triangleright}$

Algorithm 2: Breadth-First Expansion Algorithm

Input: sampled ratio r; initial hyperedges $\mathcal{E}_0^{\triangleright}$; hyperedge's adjacent set $\{\mathcal{N}_1, \cdots, \mathcal{N}_n\}$;
Output: updated hyperedges $\mathcal{E}_1^{\triangleright} = \{e_1, e_2, \cdots, e_m\}$;

$\mathcal{E}_1^{\triangleright} = \emptyset$.
for $e_i \in \mathcal{E}_0^{\triangleright}$ **do**
  $C_i = \{e_{i1}, e_{i2} \cdots, e_{ik}\}$, where $e_{ij} \in \mathcal{N}_{e_i}, j = 1, \cdots, k$, $k = r \times |\mathcal{N}_{e_i}|$. //sample k adjacent hyperedges from $\mathcal{N}_{e_i}$ for e_i using formula (4).
  $e_i^{\triangleright} = e_i \cup e_{i1} \cup e_{i2} \cup \cdots e_{ik}$
  $\mathcal{E}_1^{\triangleright} = \mathcal{E}_1^{\triangleright} \cup e_i^{\triangleright}$
end
remove duplicated hyperedges in $\mathcal{E}_1^{\triangleright}$.
return $\mathcal{E}_1^{\triangleright}$

Algorithm 3: Hybrid Expansion Algorithm

Input: aggressive parameter p_a; sampled ratio r; initial hyperedges $\mathcal{E}_0^{\triangleright}$; hyperedge's adjacent set $\{\mathcal{N}_1, \cdots, \mathcal{N}_n\}$;
Output: updated hyperedges $\mathcal{E}_1^{\triangleright} = \{e_1, e_2, \cdots, e_m\}$;

$\mathcal{E}_1^{\triangleright} = \emptyset$.
for $e_i \in \mathcal{E}_0^{\triangleright}$ **do**
  $C_i = \{e_{i1}, e_{i2} \cdots, e_{ik}\}$, where $e_{ij} \in \mathcal{N}_{e_i}, j = 1, \cdots, k$, $k = r \times |\mathcal{N}_{e_i}|$. //sample k adjacent hyperedges from $\mathcal{N}_{e_i}$ for e_i using formula (4).
  for $e_{ij} \in C_i$ **do**
    $\mathcal{P} : e_{ij}^1 \rightarrow e_{ij}^2 \rightarrow \cdots e_{ij}^t, e_{ij}^1 \in \mathcal{N}_{e_{ij}}$
    //generate a hyperedge random walk path using formula (4) with the stop probability $1 - p_a$.
    $e_{ij}^{\triangleright} = e_{ij} \cup e_{ij}^1 \cup e_{ij}^2 \cdots \cup e_{ij}^t$
  end
  $e_i^{\triangleright} = e_{i1}^{\triangleright} \cup e_{i2}^{\triangleright} \cdots \cup e_{ik}^{\triangleright}$
  $\mathcal{E}_1^{\triangleright} = \mathcal{E}_1^{\triangleright} \cup e_i^{\triangleright}$
end
remove duplicated hyperedges in $\mathcal{E}_1^{\triangleright}$.
return $\mathcal{E}_1^{\triangleright}$.

all nodes in the visiting hyperedges to e_i. Take the middle of Figure 4 as an example; when we expand hyperedge e_1, breadth-first expansion first select a portion of the neighbours of e_1 such as e_2 and e_3, then e_1 adopts all nodes in e_2 and e_3. We use r to denote the sampled ratio of e_1's neighbour hyperedges, and then present the breadth-first expansion strategy in Algorithm 2.

3.3.3 Hybrid Expansion. Intuitively, depth-first expansion intends to check remote hyperedges, building connections between different regions in the network. On the contrary, breadth-first expansion tries to enlarge the hyperedge with the nearest neighbours, which allows hyperedges to have more overlap. Based on depth-first expansion and breadth-first expansion strategies, we further fuse them and propose a hybrid expansion strategy shown in Algorithm 3. As depicted in the bottom of Figure 4, hybrid expansion first selects a portion of e_1's neighbours such as e_2 and e_4, then for each selected neighbour, it generates a hyperedge path and treats all nodes in these hyperedges as the new members of e_1.

To further illustrate the multiple levels of the hyperedges generated by our proposed expansion strategies, we take Figure 5 as an example. When we try to expand hyperedges e_1 and e_2, they may all succeed and include with each other. As a result, the previous two hyperedges now overlap and are reduced as one hyperedge. On the other case, e_1 may manage to swallow e_2, but e_2 fail to embrace e_1, thus e_1 becomes larger than before while e_2 stay unchanged, making the hyperedges hierarchical with multilayer structures. We can learn the optimal hyperedge numbers, hyperedge hierarchy, and keep the best balance between local and global structures. Next, we present a hypergraph neural network to integrate learned hyperedges and node features for linking prediction.

3.4 HNN: Hypergraph Neural Network for Linking Prediction

Having obtained the hyperedges generated by our proposed hyperedge expansion strategies, we now present a novel hypergraph neural network (HNN) to handle each level's hypergraph. The model takes hyperedges from each level as input and then aggregate node representations via node-level attention, hyperedge-level attention, and semantic-level attention.

3.4.1 Node-level Attention. Nodes in the same hyperedge usually have different importance, and the mutual influences are not uniform. To this end, we use the node-level attention to learn the correlations of nodes within the same hyperedge. Let $\mathbf{z}_i^l \in \mathbb{R}^{1\times d}$ and $\mathbf{z}_j^l \in \mathbb{R}^{1\times d}$ be the input representations of nodes i and j at level l. Here level l means we use the hypergraph generated by the l-th hyperedge expansion for l-th HNN layers. The conditional probability of node i given j within hyperedge e can be defined as follows:

$$p_v^l(i|j,e) = \frac{\exp\left(\sigma\left(\mathbf{H}^l(i,e)\cdot\mathbf{U}_v^l(i,i)\cdot\left[\mathbf{z}_i^l \oplus \mathbf{z}_j^l\right]\cdot\mathbf{p}_l^\top\right)\right)}{\sum_{t=1}^{|\mathcal{V}|}\exp\left(\sigma\left(\mathbf{H}^l(t,e)\cdot\mathbf{U}_v^l(t,t)\cdot\left[\mathbf{z}_t^l \oplus \mathbf{z}_j^l\right]\cdot\mathbf{p}_l^\top\right)\right)} \tag{5}$$

where $\mathbf{p}_l \in \mathbb{R}^{1\times 2d}$ is the fusion parameter. $\mathbf{H}^l$ is the hypergraph incidence matrix, which can be obtained after hyperedges expansions at the l-th level. $\mathbf{U}_v^l$ is the pre-defined node weight at l-th level. In this paper, nodes' weights are given with the original networks and keep unchanged, thus we have $\mathbf{U}_v^0 = \mathbf{U}_v^1 = \cdots \mathbf{U}_v^L$. We let $p_v^l(i|j,e) = 0$ if node j is not in hyperedge e. With the above formula, we update node i within hyperedge e as $\mathbf{z}_{i|e}^l$, which can be aggregated from other nodes in the same hyperedge using corresponding coefficients:

$$\mathbf{z}_{i|e}^l = \sigma\left(\sum_{j=1}^{|\mathcal{V}|}\frac{\mathbf{U}_v^l(j,j)\,\mathbf{H}^l(j,e)}{\mathbf{D}_e^l(e,e)}\cdot p_v^l(i|j,e)\cdot\mathbf{z}_j^l\right) \tag{6}$$

Note that when $l = 0$, the input representation $\mathbf{z}_j^0$ is the initial feature of node j.

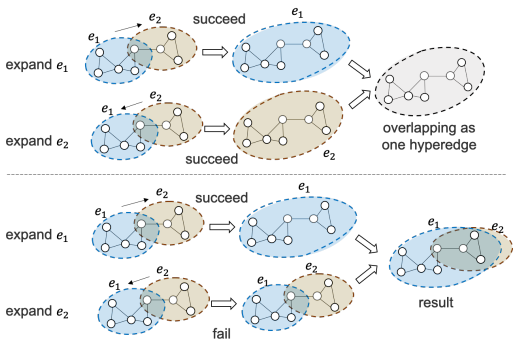


Figure 5: Hyperedges expansion cases. Through hyperedge expansion, we can reduce the number of hyperedges and construct multi-level hyperedges.

3.4.2 Hyperedge-level Attention. Many nodes in the network belong to multiple hyperedges because these hyperedges are hierarchical, and the impact from different hyperedges is also diverse. To evaluate this impact, we first obtain hyperedge representations, and then calculate each hyperedge's weight. Specifically, the representation of hyperedge e can be calculated as follows:

$$\mathbf{m}_e^l = \sum_{i=1}^{|\mathcal{V}|}\mathbf{H}^l(i,e)\cdot\mathbf{z}_{i|e}^l \tag{7}$$

where $\mathbf{m}_e^l$ is the representation of hyperedge e. Following the above, the weight of hyperedge e is defined as follows:

$$\alpha_e^l = \sigma\left(\mathbf{U}_e(e,e)\cdot\tanh\left(\mathbf{m}_e^l\cdot\mathbf{W}_\alpha^l + \mathbf{b}_\alpha^l\right)\cdot\mathbf{q}_l^\top\right) \tag{8}$$

where $\mathbf{W}_\alpha^l$, $\mathbf{b}_\alpha^l$, and $\mathbf{q}_l$ are all learnable variables.

Note that previous equations (5)-(8) are all calculated under the same meta-path. For simplicity, we omit the meta-path notation Φ without loss of generality. To deal with heterogeneous networks, we just need to generate induced graphs according to different meta-paths, and then obtain corresponding notations simultaneously. Furthermore, the representation of node i under meta-path Φ can be aggregated by all hyperedges which includes i:

$$\mathbf{z}_i^{l|\Phi} = \sum_{t=1}^{|\mathcal{E}_l^{\triangleright|\Phi}|}\frac{\mathbf{H}^{l|\Phi}(i,t)}{\mathbf{D}_v^{l|\Phi}(i,i)}\cdot\alpha_{e_t}^{l|\Phi}\cdot\mathbf{z}_{i|e_t}^{l|\Phi} \tag{9}$$

Here $\mathbf{z}_i^{l|\Phi}$ is the representation of node i after aggregated by all hyperedges under meta-path Φ. $\mathcal{E}_l^{\triangleright|\Phi}$ is the hyperedge set under meta-path Φ generated after l-th hyperedge expansion.

3.4.3 Semantic-level Attention. Note that the node embeddings from just one meta-path only reflect one aspect of information. Therefore we need to fuse the information from all candidate meta-paths. The importance of each meta-path can be defined as the average weights of all node embeddings in the same meta-path:

$$\beta_{\Phi_i}^l = \frac{\exp\left(\sum_{j=1}^{|\mathcal{V}|}\tanh\left(\mathbf{W}_\beta^l \cdot \mathbf{z}_j^{l|\Phi_i} + \mathbf{b}_\beta^l\right)\cdot \mathbf{f}_l^{\mathrm{T}}\right)}{\sum_{t=1}^{P}\exp\left(\sum_{j=1}^{|\mathcal{V}|}\tanh\left(\mathbf{W}_\beta^l \cdot \mathbf{z}_j^{l|\Phi_t} + \mathbf{b}_\beta^l\right)\cdot \mathbf{f}_l^{\mathrm{T}}\right)} \tag{10}$$

where P is the total number of all meta-paths. $\mathbf{W}_\beta^l$, $\mathbf{b}_\beta^l$, and $\mathbf{f}_l$ are learnable parameters. Based on formula (10), the final embedding is aggregated by all selected meta-paths:

$$\mathbf{Z}^{l+1} = \sum_{i=1}^{P}\beta_{\Phi_i}^l \cdot \mathbf{Z}^{l|\Phi_i} \tag{11}$$

3.5 Training Framework for Linking Prediction

Let us assume there exist L HNN layers in our framework. HNN at l-th layer takes the previous embedding $\mathbf{Z}^l$ as input, and utilises the hypergraph generated by l-th hyperedges expansion to generate $\mathbf{Z}^{l+1}$. Then we present the training framework for linking prediction as follows:

3.5.1 Training Method. To train our framework efficiently, we first train the 1-st layer by minimising the Cross-Entropy over all observed pair-wise links and sampled negative links. Then we start the 1-st hyperedge expansion and generate a new hypergraph. With the new hypergraph and the node embedding of the 1-st layer, we then fix the 1-st layer and start to train the 2-nd layer. We repeat this process until the L-th layer training is finished.

3.5.2 Negative Sampling. To train and test our model, we also need to generate some negative links. In this paper, we use a bidirectional negative sampling strategy mentioned in [5], which will draw K negative samples from both sides of a positive link, and generate $2K$ negative links in total.

3.5.3 Linking Prediction. To predict a pair-wise link with a type ϕ, we concatenate all the embeddings from layer 1 to layer L, and send them to a multilayer perceptron (MLP) for linking prediction. The predicted score is defined as follows:

$$p(i,j|\phi) = MLP_\phi\left(f\left(\mathbf{z}_i^1 \oplus \mathbf{z}_i^2 \oplus \cdots \mathbf{z}_i^L\right) \oplus f\left(\mathbf{z}_j^1 \oplus \mathbf{z}_j^2 \oplus \cdots \mathbf{z}_j^L\right)\right) \tag{12}$$

where $\oplus$ is the concatenation operator, $f(\cdot)$ is a fully connected network to reduce the dimensions of the input vectors. $MLP_\phi(\cdot)$ is the multilayer perceptron predicting pair-wise links with type ϕ. $f(\cdot)$ and $MLP_\phi(\cdot)$ are also trainable using the cross-entropy loss.

4 EXPERIMENTAL SETTINGS

4.1 Datasets

We conduct our experiments on the following widely used datasets:

Table 1: Statistics of the datasets

Datasets	Node Type #Number	Relation Type #Number	Ave. Degree	Ave. Path Length
Cora	Paper (P) #2,708	P-P #5,429	3.9	6.3
DBLP	Paper (P) #14,376	P-A #41,794	2.9	2.9
	Author (A) #14,475	P-C #14,376	2.0	2.0
	Conference (C) #20	P-T #114,624	9.8	7.9
	Term (T) #8,811			
ACM	Paper (P) #3,025	P-A #9,744	2.4	11.1
	Author (A) #5,835	P-S #3,025	2.0	2.0
	Subjects (S) #58			
Yelp	User (U) #1,286	B-U #30,838	15.8	4.1
	Service (S) #2	B-S #2,614	2.0	2.0
	Business (B) #2,614	B-R #2,614	2.0	2.0
	Star Level (L) #9	B-L #2,614	2.0	2.0
	Reservation (R) #2			

If the network is not connected, we calculate the average path length for the largest connected component.

- Cora [24]: It is a citation network which contains 2,708 papers from 7 research areas. Each paper has a one-hot code denoting the presence of 1433 unique words. There are 5,429 pair-wise links in this dataset, which stand for the citation relations of papers.
- DBLP [44]: It is an academic network that includes four types of nodes and three types of relations. There are 14,376 paper nodes (P), 14,475 author nodes (A), 20 conference nodes (C), and 8,811 term nodes (T) in total. The network has 41,794 P-A links, 14,376 P-C links, and 114,624 P-T links. In this paper, we predict both P-A links and P-C links. We transfer P-T links as one-hot codes and treat them as the features of papers.
- ACM [35]: The ACM dataset contains 3,025 papers and 5,835 authors related to 58 subjects. The network comprises two types of relations: 9,744 paper-author links and 3025 paper-subject links.
- Yelp [44]: Yelp is an online social network. The dataset contains five types of nodes, including user (U), service (S), business (B), star level (L), and reservation (R), and four types of relations: B-U, B-S, B-R, and B-L. In this paper, we only predict links with types of B-U and B-L.

Specifically, we use Cora, DBLP, ACM, and Yelp to evaluate linking prediction performance with only 10% training links. Then we discuss more profound topics on our model using the Cora dataset. We use Cora for further analysis because it has only one type of relation, which can eliminate unnecessary disturbance from relations types and more clearly uncover our model's properties. More details on these datasets are illustrated in Table 1.

4.2 Baselines

We compare our model with the following state-of-the-art methods:

- Metapath2vec [9]: This method samples a corpus of walk paths from the network according to different meta-paths and maximises the conditional probability of the target node given its context.
- PME [5]: This method first projects node embeddings into different semantic spaces and then measures each pair of nodes' distance. It learns node embeddings via minimising nodes distances on positive links and maximising the distances on negative links.
- HAN [35]: This method uses novel-level attention and semantic-level attention on normal graphs and learn node embeddings

Table 2: Link prediction results (10% observed links)

Metrics	Methods	Cora (P-P)	DBLP (P-C)	DBLP (P-A)	ACM (P-A)	ACM (P-S)	Yelp (B-U)	Yelp (B-L)
AUC	Ours	0.6288	0.7251	0.6661	0.6284	0.5975	0.5672	0.5924
	Metapath2vec	0.5013	0.5534	0.5429	0.5432	0.5367	0.5016	0.5035
	PME	0.5187	0.6070	0.5859	0.5509	0.5477	0.5054	0.5002
	HAN	0.5564	0.6206	0.6032	0.5673	0.5564	0.5203	0.5326
	HGNN	0.5552	0.6531	0.5678	0.5298	0.5435	0.5191	0.5498
	MGCN	0.5673	0.6904	0.6035	0.5964	0.5975	0.5367	0.5564
AP	Ours	0.1455	0.1606	0.1469	0.1868	0.1438	0.1104	0.1206
	Metapath2vec	0.0993	0.1025	0.1054	0.1003	0.0997	0.0993	0.0919
	PME	0.0979	0.1009	0.1033	0.0941	0.0913	0.0930	0.0914
	HAN	0.1203	0.1326	0.1207	0.1154	0.1207	0.1069	0.1204
	HGNN	0.1133	0.1393	0.1061	0.0943	0.1372	0.1009	0.1165
	MGCN	0.1208	0.1524	0.1203	0.1206	0.1438	0.1096	0.1201
F1	Ours	0.8705	0.8804	0.8742	0.8972	0.8964	0.7506	0.8002
	Metapath2vec	0.5602	0.6932	0.7321	0.5564	0.6037	0.5735	0.5942
	PME	0.6336	0.7348	0.7675	0.6194	0.6857	0.6015	0.6088
	HAN	0.7806	0.8802	0.8697	0.8742	0.8703	0.7562	0.8009
	HGNN	0.7952	0.8697	0.8635	0.8597	0.8713	0.7438	0.7929
	MGCN	0.8079	0.8702	0.8633	0.8864	0.8806	0.7506	0.8002

via the cross-entropy loss over all labeled nodes. In this paper, we concatenate the embeddings of a pair of nodes and then predict whether they have a link or not.

- HGNN [10]: This method uses a hypergraph convolutional network to learns node embeddings from a pre-defined hypergraph. The hypergraph is manually designed via features clustering and keeps fixed in the whole learning process.
- MGCN [4]: This method first uses a graph convolutional network to learn a temporal node embeddings, and then further refines these embeddings via a hypergraph neural network. The hypergraph is manually constructed via community finding, and each community is treated as one hyperedge.

4.3 Evaluation Metrics and Parameter Settings

We compare our model's performance with other baselines on AUC (a.k.a AUC@ROC), AP (a.k.a AUC@PR), and F1 value. For a fair comparison, the dimension of node embeddings is set as 100 for all models. We sampled 10 negative links for each positive link, then sample 10% links as training set, 10% links as valid set, and another 10% links as testing set. We also change the training ratio from 30% to only 1% to further evaluate our model's reliability and potential. For all models, we use PP as a meta-path in Cora. The meta-paths considered in DBLP include PAP, and PCP. The meta-paths in ACM include PAP and PSP. The meta-paths in Yelp are BUB, BLB, BSB, and BRB.

Parameters settings for the baselines are selected by the grid search method. For Metapath2vec, we set the number of walks $n = 5$, walk length $\ell = 10$, window size $w = 5$; For PME, we set margin $m = 10$. For our model, we expand hyperedges 3 times and get 4 hypergraphs, which are numbered from level 0 to level 3. Each level uses one layer HNN and an MLP with two layers. In the hyperedge expansion algorithm, we let $r = 0.3$, $p_a = 0.65$. The model is trained using Adam optimizer with learning rate 0.001 and weight decay 0.0005.

5 EXPERIMENTAL RESULTS AND ANALYSIS

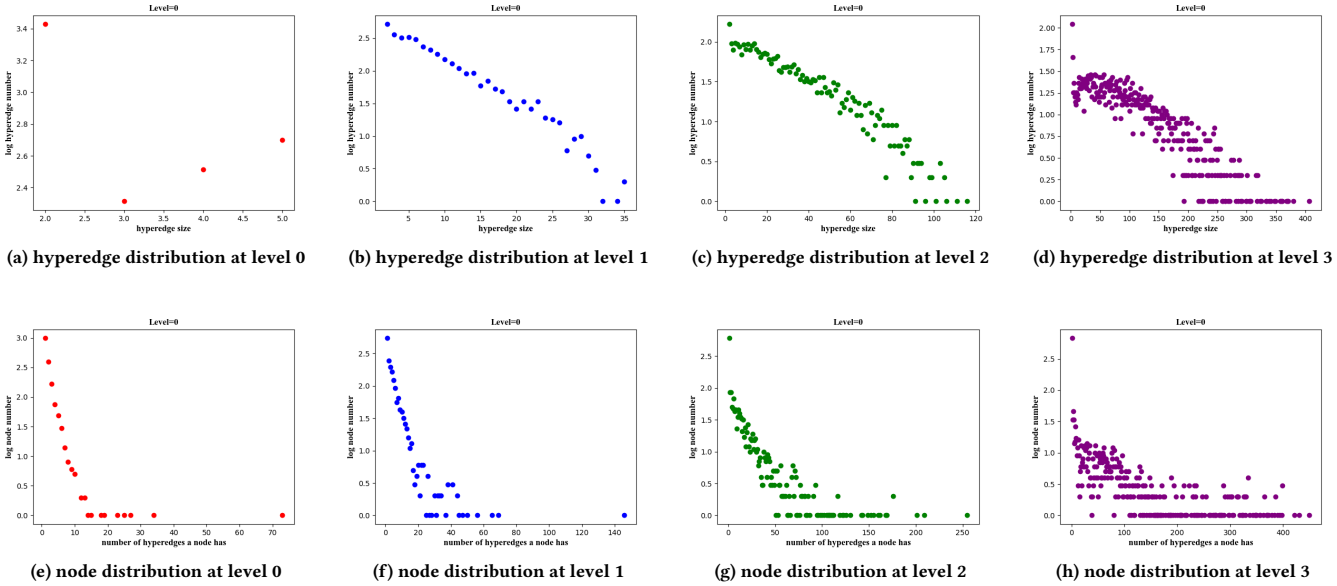
In this section, we analyse the experimental results. Before all, we wish to answer the following research questions (RQ):

- **RQ1:** How does our model work compared to the other baselines?
- **RQ2:** How do the hyperedge expansion strategies designed for multi-level hyperedges work?
- **RQ3:** How robust and potential when our model deal with network sparsity?
- **RQ4:** How does our model benefit from different hyperedge expansion strategies?
- **RQ5:** How does hyperedge expansion level impact the performance?

5.1 Effectiveness Analysis on Linking Prediction (RQ1)

We predict social links with type P-P on Cora, P-C and P-A on DBLP, P-A, P-S on ACM, and B-U, B-L on Yelp. Note that all these datasets follow the power-law distribution, which means they are all sparse networks. Based on these networks, we sampled only 10% observed links as training set to further evaluate how these models perform with extremely limited links. We compare our model and the other baselines within 50 epoch and repeat the evaluation for 10 times. Then we report the averaged AUC, AP, and F1 values in Table 2.

From Table 2, we find that our model outperforms all the other baselines. Specifically, on the Cora dataset, our model exceeds the



(a) hyperedge distribution at level 0 (b) hyperedge distribution at level 1 (c) hyperedge distribution at level 2 (d) hyperedge distribution at level 3

(e) node distribution at level 0 (f) node distribution at level 1 (g) node distribution at level 2 (h) node distribution at level 3

Figure 6: Hyperedges/nodes distributions after multi-level expansions

second at least 1.1% in AUC, 2% in AP, and 7.7% in F1. When predicting the P-C relation on the DBLP dataset, our model beats the second by 5% in AUC, 5.4% in AP, and 1.2% in F1. The relative improvements are also observed on the other link types and datasets, from which we can confirm the significant performance of our model.

Consistent with our previous discussion, we notice that GNN-based methods such as HAN, HGNN, and MGCN achieve better performance than Metapath2vec, which only sample the graph as linear paths. This suggests the importance of information aggregation when we learn node embeddings because it can make the embeddings smooth on neighbours. However, when the observed links are extremely limited, the GNN model on normal graphs such as HAN, is challenging to beat hypergraph-based methods such as HGNN and MGCN, demonstrating the advantages of hypergraphs when the network is sparsely observed. Compared with manually designed hypergraphs, our approach uses hyperedge expansion strategies to learn multi-level hyperedges and significantly improves the performance, which can be reflected in the comparison to HGNN and MGCN.

5.2 Hierarchy Analysis on Hyperedge Expansion (RQ2)

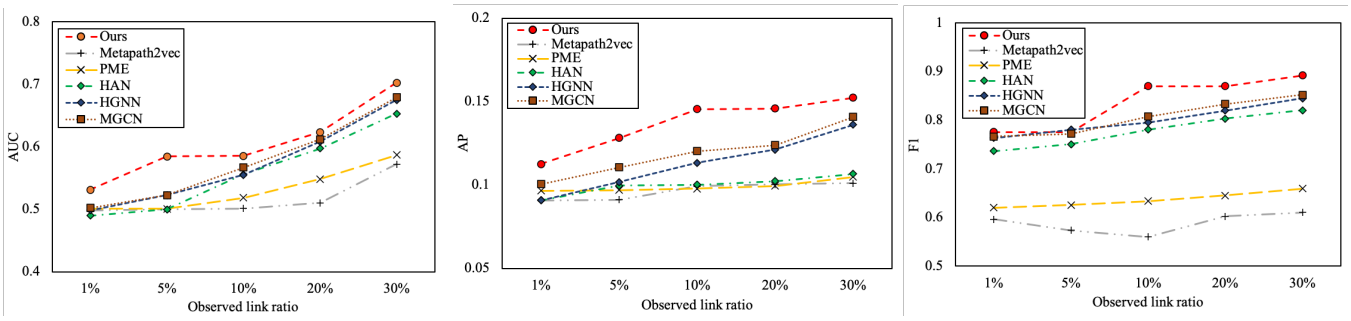
Compared with manually designed hypergraphs, our model uses hyperedge expansion to distill multi-level hyperedges, significantly improving linking prediction results. To further illustrate the hierarchy of our generated hyperedges, we use depth-first hyperedge expansion strategy to generate hyperedges with 4 levels. Then we count the hyperedge size and node size in the hypergraph, and draw Figure 6, which presents the distributions of the hyperedge size and the node degree (the number of hyperedges a node has) in each level hypergraph.

Before the hyperedge expansion, we treat graphlets with 2,3,4,5 nodes as the starting hyperedges. The distributions of the hyperedge size and the node degree at level 0 are shown in Figure 6a and Figure 6e, from which we can see the initial hyperedges only suggest a very weak hierarchy and are insufficient to support better performance. However, after the hyperedge expansion, both node degree and hyperedge size show the power-law distribution obviously, which can be seen in Figure 6b, 6c, 6d, and 6f, 6g, 6h. The distributions are very similar to Manh et al. [8], in which they discuss a simulated hypergraph with multi-level structures and also suggest similar hyperedge distributions. Through these analyses, we can find that our hyperedge expansion methods successfully learn multi-level hyperedges automatically from data.

5.3 Analysis on Network Sparsity (RQ3)

To further explore the potential of exiting methods and analyse the stability of our model, we also reduce the ratio of training links from 30% to only 1%. The performance of our model and other baselines is shown in Figure 7, from which we have the following observations:

When the training links reduced to only 1%, most baselines have dropped down to the ground bottom (nearly 0.5 in AUC and 0.0909 in AP, which is the worst case for the dataset). This means most baselines have been infeasible to learn useful representations for link predictions. Compared with baselines, our model still keeps meaningful performance on AUC and AP. The improvements are even more considerable when the training ratio increases from 1% to 5%, which can further demonstrate our model's outstanding performance in sparsely observed networks. With the training ratio increase from 5% to 30%, most baselines start to work. In particular,



(a) AUC (b) AP (c) F1

Figure 7: Performance w.r.t observed links ratio

GNN-based methods such as HAN, HGNN, and MGCN are earlier to improve their performance, while Metapath2vec and PME begin to show improvement when the training ratio is larger than 10%. This observation suggests that GNN-based methods perform better in making node embeddings smooth on pair-wise links.

With the above analysis, we can confirm that our model is robust and keeps ahead with larger tolerance to the observed link ratio. This is especially helpful in the real-world situations because most social networks are sparsely connected or sparsely observed.

5.4 Analysis on Different Hyperedge Expansion Strategies (RQ4)

To figure out the impact on our proposed hyperedge expansion strategies, we repeat the evaluation for linking prediction with different hyperedge expansion strategies on Cora (P-P), DBLP (P-C, P-A), ACM (P-A, P-S), and Yelp (B-U, B-L).

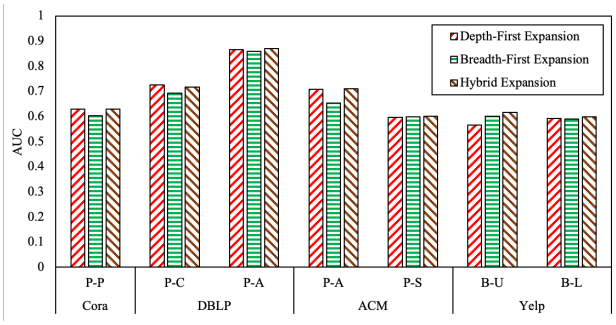


Figure 8: Performance w.r.t hyperedge expansion strategies

As depicted in Figure 8, we notice that the hybrid expansion leads by a narrow margin in most cases such as Cora (P-P), DBLP (P-A), ACM (P-A, P-S), and Yelp (B-L). The advantage of the hybrid expansion is more observable in Yelp (B-U). However, in DBLP (P-C), depth-first expansion beats hybrid expansion and keep to the top. Compare the depth-first expansion with the breadth-first expansion, we can find that the breadth-first expansion performs better in Yelp (B-U), keeps similar performances with breadth-first expansion in Yelp (B-L), ACM (P-S), and DBLP (P-A), and shows lower results in the rest cases.

Considering the network properties, we can conclude that breadth-first expansion does better when the nodes in a network have larger degrees. Depth-first expansion is better than the breadth-first expansion if the network contains longer paths. Although the hybrid expansion does better than the other strategies in most cases, it relies on both depth-first and breadth-first expansion strategies and inherits two external parameters (aggressive parameter p_a, and sampled ratio r), which need more work to fit data.

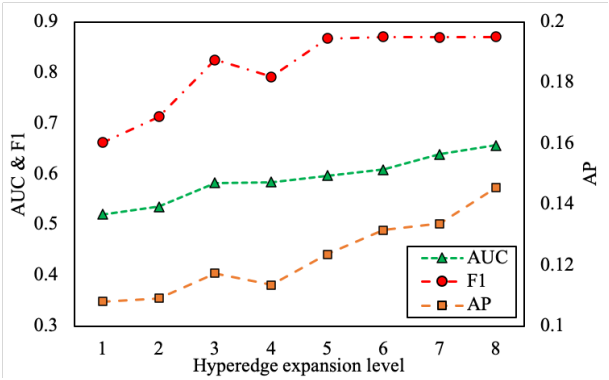


Figure 9: Performance w.r.t hyperedge expansion level

5.5 Impact on Hyperedges Expansion Levels (RQ5)

Intuitively, hypergraphs' hierarchical structures should be achieved within limited expansion levels, which means there is no more need to expand existing hyperedges and thus save the training time. Take depth-first expansion as an example, if we expand hyperedges too many times, we have to reduce the aggressive parameter p_a at higher levels so that the sizes of learned hyperedges are in a reasonable range. To illustrate this, we use depth-first expansion strategy with the aggressive parameter $p_a = 0.65$ to expand the initial hyperedges for 8 times and report the results at each level in Figure 9 where the horizontal axis is the hyperedge expansion level, the vertical axis in left is AUC and F1 results, and the right vertical axis is AP values.

As shown in Figure 9, we can see that our model's performance becomes better with the hyperedge expansion level increase. However, this trend turns to grow slowly after we conduct 4 times of hyperedge expansion. This means we do not need to conduct more hyperedge expansions because only a few times of hyperedge expansions are sufficient to make the hyperedges hierarchal and support our model to achieve the best performance.

6 RELATED WORK

In this section, we discuss related topics in our paper, including network embedding, hypergraph learning, and link prediction.

6.1 Network Embedding

Network embedding aims to learn low-dimensional representations for nodes in the network. Traditional methods [12, 26] usually sample paths from the network and then use the skip-gram method to preserve neighbouring proximity in the path. Following this idea, some extensions [17, 37] have been proposed recently so that the model can deal with heterogeneous networks. Besides these works, other researches leverage graph neural networks such as attention networks [35] and convolutional networks [16] to learn high-quality embeddings via information aggregation [3]. For example, Hu et al. [18] design an adversarial learning framework with a relation generator and discriminator to learn node embeddings preserving the heterogeneous information in the network. Fu et al. [11] propose a new information aggregation method based on meta path. The model includes intra-metapath and inter-metapath aggregations so that their learned embeddings can fit multiple semantics well. Similarly, Wang et al. [36] also leverage meta-path to calculate nodes proximities and learn node embeddings for dynamic networks. In summary, most of the related works pursue the smoothness of node embeddings so that the learned representations can support the downstream task such as node classification and link prediction. The smoothness of nodes plays a crucial role in the performance on these tasks, and most of these works achieve node smoothness using pair-wise links. However, when the pair-wise links are limited, it will be hard to keep smoothness on the network.

6.2 Hypergraph Learning

Recently, hypergraph learning has been widely used in relational inference [7], location-based social networks [41], personality perception [49], and recommender systems [50] because of its capacity for modelling high-level relations. To benefit from hypergraphs, researchers need to study hypergraph structures and have raised many interesting works. For example, Zhang et al. [46] propose an EM algorithm to predict whether a set of objects belong to the same tulpe, which they called hyperlink. Yoon et al. [43] research the correlation between the hyperedge size and the hyperedge prediction, and find the limitations of pair-wise interactions. Zhang et al. [48] study the dynamic hypergraph structure for node classification using spectral theory. Manh et al. [8] analyse a simulated hypergraph and demonstrate the importance of hierarchical structures for the hypergraphs. Besides hypergraph structures, there also exist some works trying to learn node embeddings from hypergraphs. Inspired by graph neural networks on normal graphs, hypergraph neural networks on hypergraphs are also proposed recently. For example, Feng et al. [10] offer a hypergraph convolutional networks based on spectral factorization on the Laplacian matrix of a hypergraph. Zhang et al. [47] use self-attention to learn node embeddings from the hypergraph. Jiang et al. [20] conduct the convolutional operation on both vertices and hyperedges and then they present the hypergraph neural networks on dynamic data. Although much progress has been achieved, most related works still need to use manually created hyperedges, which are far from multi-level.

6.3 Linking Prediction

Linking prediction aims to predict whether a pair of nodes in the graph has a link with the specific type. This problem can be explored by learning effective node embeddings, which can preserve neighbour proximity [5, 14, 44]. Recently, GNN-based methods have been widely applied to this area and suggest the advantages over traditional methods [4]. For example, Li et al. [23] use a graph attention network to learn the representations of two networks and then present a type-aware algorithm to align nodes from the two networks. Qu et al. [27] predict continuous-time links via temporal graph neural networks. In real-world situations, most networks follow power-law distributions, which means many nodes have only limited neighbours. To fit this situation, Hao et al. [15] use nodes' external attributes to find the most likely positions of nodes. However, this work can only be used in attributed graphs. To rely less on external attributes, Ostapuk et al. [25] propose to learn link embeddings via active learning. But this work needs persistent manual annotations. To further improve the performance, researchers have realised the importance of higher-level relations on the network [2, 29]. For example, Shao et al. study the correlations between network community and link prediction, then they propose a link prediction method based on low-rank matrix completion. Wang et al. [38] extract multi-level subgraphs and then use a graph neural network to learn the representations of each pair of nodes for link prediction. Joshi et al. [22] further propose a method to prune network subgraphs to that the neighbour proximity can be learned more effectively in knowledge graphs. Although these works have achieved good performance, rare works try to consider how their models keep stable results when the observed links are extremely limited, and most of them still heavily rely on pair-wise links. In the real world, however, most pair-wise relations are not easy to be observed, making the linking prediction under sparsely observed networks still challengeable.

7 CONCLUSION

In this paper, we aim to predict social links on sparsely observed networks, and propose a novel hypergraph-based framework. We uncover the importance of hyperedge hierarchy in the linking prediction and present three novel methods to generate multi-level hyperedges. We use a well-designed hypergraph neural network to learn node embeddings, and the experimental results confirm the superiorities of our model over state-of-the-art methods.

ACKNOWLEDGMENTS

This work is supported by National Key R&D Program of China (Grants No. 2017YFB1003000, 2019YFC1521403), National Natural Science Foundation of China (Grants No. 61972087, 61772133,

61632008), National Social Science Foundation of China (Grants No. 19@ZH014), Natural Science Foundation of Jiangsu province (Grants No. SBK2019022870), Jiangsu Provincial Key Laboratory of Network and Information Security (Grants No. BM2003201), Key Laboratory of Computer Network Technology of Jiangsu Province (Grants No. BE2018706), Key Laboratory of Computer Network and Information Integration of Ministry of Education of China (Grants No. 93K-9), ARC Discovery Project (Grant No.DP190101985). The first author Mr. Xiangguo Sun, in particular, wants to thank his parents for their support during his tough period.

REFERENCES

[1] Marco Bressan, Flavio Chierichetti, Ravi Kumar, Stefano Leucci, and Alessandro Panconesi. 2017. Counting graphlets: Space vs time. In *WSDM*. 557–566.
[2] Lei Cai and Shuiwang Ji. 2020. A Multi-Scale Approach for Graph Link Prediction. In *AAAI*. 3308–3315.
[3] Yukuo Cen, Xu Zou, Jianwei Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Representation learning for attributed multiplex heterogeneous network. In *SIGKDD*. 1358–1368.
[4] Hongxu Chen, Hongzhi Yin, Xiangguo Sun, Tong Chen, Bogdan Gabrys, and Katarzyna Musial. 2020. Multi-level Graph Convolutional Networks for Cross-platform Anchor Link Prediction. In *SIGKDD*. 1503–1511.
[5] Hongxu Chen, Hongzhi Yin, Weiqing Wang, Hao Wang, Quoc Viet Hung Nguyen, and Xue Li. 2018. PME: projected metric embedding on heterogeneous networks for link prediction. In *SIGKDD*. 1177–1186.
[6] Tong Chen, Hongzhi Yin, Hongxu Chen, Rui Yan, Quoc Viet Hung Nguyen, and Xue Li. 2019. Air: Attentional intention-aware recommender systems. In *ICDE*. 304–315.
[7] Mayukh Das, Devendra Singh Dhami, Gautam Kunapuli, Kristian Kersting, and Sriraam Natarajan. 2019. Fast relational probabilistic inference and learning: Approximate counting via hypergraphs. In *AAAI*, Vol. 33. 7816–7824.
[8] Manh Tuan Do, Se-eun Yoon, Bryan Hooi, and Kijung Shin. 2020. Structural patterns and generative models of real-world hypergraphs. In *SIGKDD*. 176–186.
[9] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. Metapath2vec: Scalable representation learning for heterogeneous networks. In *SIGKDD*. 135–144.
[10] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. 2019. Hypergraph neural networks. In *AAAI*, Vol. 33. 3558–3565.
[11] Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. 2020. MAGNN: Metapath Aggregated Graph Neural Network for Heterogeneous Graph Embedding. In *The Web Conference (WWW)*. 2331–2341.
[12] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *SIGKDD*. 855–864.
[13] Lei Guo, Hongzhi Yin, Qinyong Wang, Tong Chen, Alexander Zhou, and Nguyen Quoc Viet Hung. 2019. Streaming session-based recommendation. In *SIGKDD*. 1569–1577.
[14] Xiaorong Hao, Tao Lian, and Li Wang. 2020. Dynamic Link Prediction by Integrating Node Vector Evolution and Local Neighborhood Representation. In *SIGIR*. 1717–1720.
[15] Yu Hao, Xin Cao, Yixiang Fang, Xike Xie, and Sibo Wang. 2020. Inductive Link Prediction for Nodes Having Only Attribute Information. In *IJCAI*. 1209–1215.
[16] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *SIGIR*. 639–648.
[17] Yu He, Yangqiu Song, Jianxin Li, Cheng Ji, Jian Peng, and Hao Peng. 2019. HeteSpaceyWalk: a heterogeneous spacey random walk for heterogeneous information network embedding. In *CIKM*. 639–648.
[18] Binbin Hu, Yuan Fang, and Chuan Shi. 2019. Adversarial learning on heterogeneous information networks. In *SIGKDD*. 120–129.
[19] Binbin Hu, Zhiqiang Zhang, Chuan Shi, Jun Zhou, Xiaolong Li, and Yuan Qi. 2019. Cash-out user detection based on attributed heterogeneous information network with a hierarchical attention mechanism. In *AAAI*, Vol. 33. 946–953.
[20] Jianwen Jiang, Yuxuan Wei, Yifan Feng, Jingxuan Cao, and Yue Gao. 2019. Dynamic hypergraph neural networks. In *IJCAI*. 2635–2641.
[21] Taisong Jin, Liujuan Cao, Baochang Zhang, Xiaoshuai Sun, Cheng Deng, and Rongrong Ji. 2019. Hypergraph induced convolutional manifold networks. In *IJCAI*. 2670–2676.
[22] Unmesh Joshi and Jacopo Urbani. 2020. Searching for Embeddings in a Haystack: Link Prediction on Knowledge Graphs with Subgraph Pruning. In *The Web Conference (WWW)*. 2817–2823.
[23] Xiaoxue Li, Yanmin Shang, Yanan Cao, Yangxi Li, Jianlong Tan, and Yanbing Liu. 2020. Type-Aware Anchor Link Prediction across Heterogeneous Networks Based on Graph Attention Network. In *AAAI*, Vol. 34. 147–155.
[24] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval* 3, 2 (2000), 127–163.
[25] Natalia Ostapuk, Jie Yang, and Philippe Cudré-Mauroux. 2019. Activelink: deep active learning for link prediction in knowledge graphs. In *The Web Conference (WWW)*. 1398–1408.
[26] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *SIGKDD*. 701–710.
[27] Liang Qu, Huaisheng Zhu, Qiqi Duan, and Yuhui Shi. 2020. Continuous-Time Link Prediction via Temporal Dependent Graph Neural Network. In *The Web Conference (WWW)*. 3026–3032.
[28] Ryan A Rossi, Anup Rao, Tung Mai, and Nesreen K Ahmed. 2020. Fast and Accurate Estimation of Typed Graphlets. In *The Web Conference (WWW)*. 32–34.
[29] Paolo Rosso, Dingqi Yang, and Philippe Cudré-Mauroux. 2020. Beyond triplets: hyper-relational knowledge graph embedding for link prediction. In *The Web Conference (WWW)*. 1885–1896.
[30] Junming Shao, Zhong Zhang, Zhongjing Yu, Jun Wang, Yi Zhao, and Qinli Yang. 2019. Community Detection and Link Prediction via Cluster-driven Low-rank Matrix Completion.. In *IJCAI*. 3382–3388.
[31] Xiangguo Sun, Hongzhi Yin, Bo Liu, Hongxu Chen, Jiuxin Cao, Yingxia Shao, and Nguyen Quoc Viet Hung. 2019. Heterogeneous Hypergraph Embedding for Graph Classification. In *WSDM*.
[32] 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 *WSDM*. 592–600.
[33] Pinghui Wang, Junzhou Zhao, Xiangliang Zhang, Zhenguo Li, Jiefeng Cheng, John CS Lui, Don Towsley, Jing Tao, and Xiaohong Guan. 2017. MOSS-5: A fast method of approximating counts of 5-node graphlets in large graphs. In *ICDE*, Vol. 30. 73–86.
[34] Weiqing Wang, Hongzhi Yin, Xingzhong Du, Wen Hua, Yongjun Li, and Quoc Viet Hung Nguyen. 2019. Online user representation learning across heterogeneous social networks. In *SIGIR*. 545–554.
[35] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *The Web Conference (WWW)*. 2022–2032.
[36] Xiao Wang, Yuanfu Lu, Chuan Shi, Ruijia Wang, Peng Cui, and Shuai Mou. 2020. Dynamic Heterogeneous Information Network Embedding with Meta-path based Proximity. *IEEE Transactions on Knowledge and Data Engineering* (2020).
[37] Xiao Wang, Yiding Zhang, and Chuan Shi. 2019. Hyperbolic heterogeneous information network embedding. In *AAAI*, Vol. 33. 5337–5344.
[38] Zihan Wang, Zhaochun Ren, Chunyu He, Peng Zhang, and Yue Hu. 2019. Robust Embedding with Multi-Level Structures for Link Prediction.. In *IJCAI*. 5240–5246.
[39] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* (2020).
[40] Pinghua Xu, Wenbin Hu, Jia Wu, and Bo Du. 2019. Link prediction with signed latent factors in signed social networks. In *SIGKDD*. 1046–1054.
[41] Dingqi Yang, Bingqing Qu, Jie Yang, and Philippe Cudre-Mauroux. 2019. Revisiting user mobility and social relationships in LBSNs: A hypergraph embedding approach. In *The Web Conference (WWW)*. 2147–2157.
[42] Hongzhi Yin, Qinyong Wang, Kai Zheng, Zhixu Li, Jiali Yang, and Xiaofang Zhou. 2019. Social influence-based group representation learning for group recommendation. In *ICDE*. 566–577.
[43] Se-eun Yoon, Hyungseok Song, Kijung Shin, and Yung Yi. 2020. How Much and When Do We Need Higher-order Informationin Hypergraphs? A Case Study on Hyperedge Prediction. In *The Web Conference (WWW)*. 2627–2633.
[44] Linmei Hu Zhiyuan Liu Yuanfu Lu, Chuan Shi. 2019. Relation Structure-Aware Heterogeneous Information Network Embedding. In *AAAI*.
[45] Reza Zafarani, Mohammad Ali Abbasi, and Huan Liu. 2014. *Social media mining: an introduction*. Cambridge University Press.
[46] Muhan Zhang, Zhicheng Cui, Shali Jiang, and Yixin Chen. 2018. Beyond link prediction: Predicting hyperlinks in adjacency space. In *AAAI*.
[47] Ruochi Zhang, Yuesong Zou, and Jian Ma. 2020. Hyper-SAGNN: a self-attention based graph neural network for hypergraphs. In *ICLR*.
[48] Zizhao Zhang, Haojie Lin, and Yue Gao. 2018. Dynamic Hypergraph Structure Learning. In *IJCAI*. 3162–3169.
[49] Sicheng Zhao, Guiguang Ding, Jungong Han, and Yue Gao. 2018. Personality-Aware Personalized Emotion Recognition from Physiological Signals.. In *IJCAI*. 1660–1667.
[50] Xiaoyao Zheng, Yonglong Luo, Liping Sun, Xintao Ding, and Ji Zhang. 2018. A novel social network hybrid recommender system based on hypergraph topologic structure. *World Wide Web* 21, 4 (2018), 985–1013.