HINGCN: Heterogeneous information network revisited with GCN

Danhao Ding*, Xiang Li*, Nikos Mamoulis†, Ben Kao*
*The University of Hong Kong, Pokfulam Road, Hong Kong
†University of Ioannina, Ioannina, Greece
*{dhding2, xli2, kao}@cs.hku.hk †nikos@uoi.gr

ABSTRACT

GCN on HIN.

KEYWORDS

Semi-supervised classification; graph convolution neural networks; heterogeneous information network

ACM Reference Format:
Danhao Ding*, Xiang Li*, Nikos Mamoulis†, Ben Kao*. 2018. HINGCN: Heterogeneous information network revisited with GCN. In *WWW 2018: The 2018 Web Conference, April 23–27, 2018, Lyon, France.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3178876.3185993

1 INTRODUCTION

Dan:TODO Recent advances in graph representation learning... **Graph Convolution Network** (GCN) gives a new direction on interpretative deep learning on graphs. Compared to traditional classification methods on graph like label propagation or matrix factorization, GCN models provides an end-to-end solution as well as state-of-art performance results.

On the other hand, however, deep learnings on heterogeneous graphs have been less explored. Successes of HIN-enhanced algorithms such as metapath2vec [6], Dan:cite another few suggest that models that consider heterogeneous data may produce better results. Intuitively, deep learning models taking heterogeneity into account may produce decent results as well. Recently, some primitive application of GCN on heterogeneous information graphs have been proposed. However these methods did not fully utilize heterogeneity of information..

Our main contributions are:

• We proposed a heterogeneous graph convolution algorithm that is capable of capturing edge information.
• Dan:don't know
• We conduct extensive experiments to evaluate the performance of HINGCN against 9 other classification methods. Our results show that HINGCN performs very well against the competitors. In particular, it is very robust in that it consistently performs well over all the datasets tested. Also, it outperforms others by wide margins for datasets that are highly multi-scale.

WWW 2018, April 23–27, 2018, Lyon, France

ACM ISBN 978-1-4503-5639-8/18/04..
https://doi.org/10.1145/3178876.3185993

The rest of the paper is organized as follows. Section 2 mentions related works on heterogeneous graph neural networks, graph embedding and described several semi-supervised classification algorithms. Section 4 presents the HINGCN algorithm. Section 5 describes the experiments and presents experimental results. Finally, Section 6 concludes the paper.

2 RELATED WORK

2.1 Semi-supervised Learning on Graphs

The main focus in this paper is semi-supervised classification on graphs. Given a graph with limited labels on a set of nodes, the goal of learning is to predict the label of the remaining vertices.

In the past few decades, graph-based semi-supervised algorithms often rely on the clustering assumption [4], which assumes that spatially close vertices on a graph usually share the same label. Min-cuts, randomized min-cuts, spectral graph transducer, label propagation, iterative classification are examples of this line of research.

In many applications, vertex instances often come along with feature vectors that needs to be considered apart from the graph. Regularization-based algorithms like LapSVM [2] and Planetoid [26] rely on adding graph Laplacian regularizer to supervised learners. Graph embedding is a class of algorithms aimed to learn low dimensional representations that jointly model vertex attributes and graph structure. In semi-supervised classification, the learnt embeddings serve as input features of downstream classification engines such as logistics, SVM, and MLP classifiers Dan:cite?. There are various embedding methods designed for homogeneous graphs, which generally falls into three categories: deep neural network methods[22], random walk based methods[8, 15], matrix factorization methods and label propagation methods.

Heterogeneous graph embedding methods differ from homogeneous ones, in the sense that meta-path based structure information is considered. ESim[16] is proposed to learn structure-embedded similarity by sampling meta-path instances. Metapath2vec [6] proposes a meta-path based random walk and feed these walk instances to skip-gram engine. HIN2Vec [7] considers a joint loss of multiple tasks, where embeddings of vertex and meta-path are learnt simultaneously. Sun et al. [17] proposes a meta-graph model, which captures hidden relations in a meta-graph by tensor decomposition.

2.2 Graph Convolution Networks

Recently, graph convolution neural networks (GCNNs) have became popular approaches in graph embedding tasks. Inspired by convolution neural networks (CNNs), GCNNs neatly combine graph structures and vertex features, by propagating features of labeled

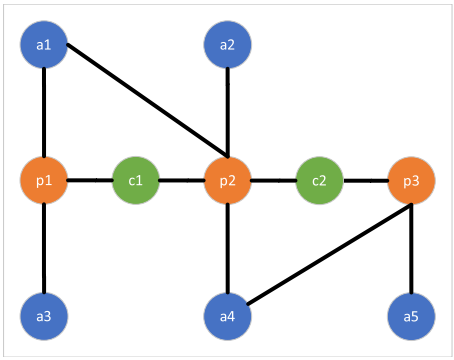

Figure 1: An example of a heterogeneous information network (DBLP). There are three types of nodes (i.e. author, paper, conference).

vertices to unlabeled neighbors through multiple layers of convolution. These methods generally fall into two categories: spectral GCNNs and spatial GCNNs. Spectral GCNNs generally decompose graph signals on the spectral domain using graph Fourier transform, and then convolve on the spectral components. Bruna et al. [3] generalizes CNNs to graphs by finding Fourier basis of graph Laplacian. To reduce computation cost in large graphs, Defferrard et al. [5] employs a K-order Chebyshev polynomial approximation to provide a faster filtering. Spatial GCNNs view convolutions differently, where a new feature vector is constructed for each vertex using an aggregation of spatially close neighbors. Hamilton et al. [9] proposes GraphSAGE for inductive learning on unseen data. GraphSAGE learns a function that aggregate node embeddings from a fixed size neighborhood.

It is also worth mentioning Graph Attention Networks [21], where attention mechanisms [1, 20] are applied to discriminate different nodes in a neighborhood. New embeddings are generated as a weighted sum of attention coefficients and neighbor feature vectors.

More recently, HAN [23] is proposed to compute node embeddings on heterogeneous graphs, where all meta-paths are simultaneously considered. This method, however, only consider meta-path specific neighbors of target nodes and is unable to discriminate different intermediate nodes in meta-path instances.

3 PRELIMINARY

Definition 1. **Heterogeneous Information Networks (HIN)**[11]. Let $T = \{T_1, \ldots, T_m\}$ be a set of m object types. For each type T_i, let n_i and $X_i = x_i 1, \ldots, x_{in_i}$ be the number and the set of objects of type Ti, respectively. An HIN is a graph $G = (V, E)$, where $V = \bigcup_{i=1}^{m} X_i$, and E is a set of links, each represents a binary relation between two objects in V. If $m = 1$ (i.e., there is only one object type), G reduces to a homogeneous information network.

Example 1. *In Fig. 1, we give an example of HIN. There are three types of nodes, Author, Paper and Conference. And we illustrate two types of edges: Author-Paper, Paper-Conference.*

Table 1: Description of symbols

Symbol	Description
Φ	Meta-path
N^{Φ}	Set of neighbors with respect to a given meta-path Φ
$h^{(t)}$	Node embedding at layer t; $h^{(0)}$ is input node feature
$r^{(t)}$	Edge embedding at layer t; $r^{(0)}$ is input edge feature
$a_{i,j}^{\Phi}$	Attention of node pair (i,j) under meta-path Φ
$w_i^{\Phi_p}$	Weight of meta-path Φ_p related component of w_i
z_i	Output embedding for node i
W_F	Learnable weight matrix of fully connected layer F
σ	Sigmoid function
$\oplus$	Concatenate function
$\odot$	Hadamard (element-wise) product function

Definition 2. **Metapath**[18]. A meta-path Φ is a path defined on a graph schema. Meta-path Φ: $T_1 \xrightarrow{R_1} \cdots \xrightarrow{R_l} T_{l+1}$ defines a composite relation $R = R_1 \circ \cdots \circ R_l$ that relates objects of type T_1 to objects of type T_{l+1}. We say a path $p = (x_1 x_2 \ldots x_{l+1})$ between x_1 and x_{l+1} in network G follows the meta-path Φ, if each object is of type T_i and each link $ei = \langle x_i x_{i+1} \rangle$ belongs to each relation R_i in Φ. We say that $p_{x_1 \rightsquigarrow x_{l+1}}$ is an instance of meta-path Φ, denoted by $p_{x_1 \rightsquigarrow x_{l+1}} \vdash \Phi$. Moreover, in this paper we say x_v is a meta-path Φ related neighbor of x_u, denoted by $x_v \in N^{\Phi}(x_u)$

Example 2. *In Fig. 1, a_3 is an APA-related neighbor of a_1, and a_5 is an APCPA-related neighbor of a_1.*

The notations throughout the rest of paper are shown in Table 1.

4 ALGORITHM

In this section we describe our HINGCN algorithm. Figure **??**(c) shows a flow diagram of HINGCN. Dan:TODO:signposting

4.1 Edge features as supplementary information

According to research findings of Xu et al. [24], discriminative power of a graph neural network determines its representation capability. We claim that information is loss if we only consider adjacency of target type vertices. Let's consider again the example shown in Fig. 1. In long tailed meta-path *APCPA*, a shrinking *A-A* adjacency loses information from intermediate *PCP* nodes. And a graph neural network using only adjacency information cannot distinguish between two instances $p_1 = (a_1 p_1 c_1 p_2 a_4)$ and $p_2 = (a_1 p_2 c_2 p_3 a_4)$. Instead, both instances are represented as edge $\langle a_1 a_4 \rangle$. Therefore we propose to improve expressivity of our network by adding additional edge feature to homogeneous *A-A* edges. These edge features are designed to effectly distinguish different intermediate nodes between meta-path instances. We propose edge features that is composed of three different components, namely path count, intermediate node embeddings and PathSim[18].

•**Path Count**. Different object pairs have different numbers of path instance between them and these numbers may reflect object similarity under a given meta-path. We propose to record count of

path between objects x_u and x_v under meta-path Φ by:

$$c(x_u, x_v) = |\{p_{x_u \rightsquigarrow x_v} : p_{x_u \rightsquigarrow x_v} \vdash \Phi\}|$$

•**Intermediate object embedding**. To discriminate between different intermediate objects, we propose to use a sum of pre-trained node embedding as a component of edge feature. This node embedding can be trained from heterogeneous embedding algorithms such as [6, 8]. We favor sum over other aggregators since this operation possibly preserves maximum discriminant power on graphs [24]. Given a meta-path instance $p = (x_1 x_2 \dots x_{l+1})$, we calculate

$$o(x_1, x_{l+1}) = \frac{1}{(l+1) \cdot c(x_1, x_{l+1})} \sum_{i=1}^{l+1} e_i$$

where e_i is embedding of object x_i. Division of number of vertices $l+1$ and path count $c(x_u, x_v)$ is proposed to normalize intermediate object embedding to the scale of input object embeddings. Since path count is extracted and recorded as $c(x_u, x_v)$, this does no result in extra loss of information.

•**PathSim**. PathSim is proposed in [18] to measure similarity between two objects of a same type on a heterogeneous graph. Given a symmetric meta-path Φ, PathSim between x_u and x_v is:

$$s(x_u, x_v) = \frac{2 \times |\{p_{x_u \rightsquigarrow x_v} : p_{x_u \rightsquigarrow x_v} \vdash \Phi\}|}{|\{p_{x_u \rightsquigarrow x_u} : p_{x_u \rightsquigarrow x_u} \vdash \Phi\}| + |\{p_{x_v \rightsquigarrow x_v} : p_{x_v \rightsquigarrow x_v} \vdash \Phi\}|}$$

The edge feature between objects x_u and x_v under meta-path Φ is thereby concatenated as:

$$r_\Phi^{(0)}(x_u, x_v) = c(x_u, x_v) \oplus o(x_u, x_v) \oplus s(x_u, x_v) \tag{1}$$

4.2 Aggregating node embedding

We propose a modified version of scaled dot product attention mechanism to aggregate node embedding from its neighbor. Given a meta-path Φ, we update vertex embedding as follow. First we apply 2 layer MLPs on node embedding and its neighbor respectively for scaling purpose. The query vector of node i is

$$q_i = W_{a1}^{(t)}(\tanh(W_{a2}^{(t)} h_i^{(t)})) \tag{2}$$

And key vector $k_{i,j}$ is calculated as:

$$k_{i,j} = W_{a3}^{(t)}(\tanh(W_{a4}^{(t)}(h_j^{(t)} \oplus r_{i,j}^{(t)}))) \tag{3}$$

And the value vector of neighbor j is calculated as:

$$v_{i,j}^{(t)} = W^{(t)}(h_j^{(t)} \oplus r_{i,j}^{(t)}) \tag{4}$$

where edge feature $r_{i,j}^{(t)}$ is concatenated to vertex feature of the neighbors, in order to provide additional information. It is worth noticing that due to unique feature of each edge, both key and value vectors are no longer universal for different query nodes. We believe this uniqueness improves expressivity of self-attention mechanism for HIN embedding tasks.

Afterwards, the importance of a neighbor j to object i is calculated as:

$$\tilde{a}_{i,j}^{(t)} = q_i^T \cdot k_{i,j} \tag{5}$$

Then we normalize this importance coefficient by softmax function and calculate attention coefficient as:

$$a_{i,j}^{(t)} = \text{softmax}(\tilde{a}_{i,j}^{(t)}) = \frac{\exp(\tilde{a}_{i,j}^{(t)})}{\sum_{k \in N_i^\Phi} \exp(\tilde{a}_{i,k}^{(t)})} \tag{6}$$

Finally the output of the node aggregation layer is a weighted sum of value vectors.

$$h_i^{(t+1)} = \text{ReLU}(W_{h1} \cdot \sum_{j \in N_i^\Phi} a_{i,j}^{(t)} \cdot v_{i,j}^{(t)} + W_{h2} \cdot h_i^{(t)}) \tag{7}$$

Although a vertex i is always in its own meta-path neighbor, we still explicitly add term $h_i^{(t)}$ in equation 7 following GraphSAGE[9].

4.3 Aggregating edge embedding

It is a natural idea to treat edge embeddings as parameters and update on pre-trained edge embeddings by gradient descend. However, this is not feasible in our experiment condition. Unlike language models where only a small portion of pre-trained embeddings are involved, in transductive settings, all edge embeddings we processed contribute to training process. Due to the fact that the shrinking homograph is much denser than original graph, treating these edge embeddings as parameters would drastically increase number of parameters involved the model. A dense homogeneous adjacency could result in billions of parameters, especially in long tail meta-paths like *APCPA* in DBLP dataset, where density of adjacency matrix could reach Dan:30%? or more.

Therefore, we propose another way to fine-tune edge embeddings by allowing convolution on edge embeddings. Pre-trained embeddings are served as initial feature, and the edge convolution layer is allowed to update edge embeddings by aggregating information from its endpoint nodes.

$$r_{i,j}^{(t+1)} = \text{ReLU}(W_{r1} \cdot (h_i^{(t)} \oplus h_j^{(t)} \oplus r_{i,j}^{(t)}) + W_{r2} \cdot r_{i,j}^{(t)}) \tag{8}$$

Term $W_{r2} \cdot r_{i,j}^{(t)}$ at the end of the equation serve as a shortcut residual, in order to guarantee a feasible convergence.

In multi-layer HINGCN, we propose an alternative update scheme of node embedding and edge embedding.

4.4 Dealing with over-smoothing via adaptive depth

Plain GCNs often degrades rapidly as number of layers increases [13, 14]. Stacking four or more layers into GCNs cause over-smoothing problem and features of vertices tends to converge to the same value. In heterogeneous graphs, this problem is even more severe. Let's consider again meta-path *APCPA* in Fig. 1. A two layer $a_1 a_2 a_3$ propagation on the shrinking $A - A$ homo-graph actually passes across **8** edges on the underlying heterogeneous graph, which is a terrible number of layers for plain graph convolution networks. In previous sections, our model already address this problem by including edge features in node update and updating edge feature. To further alleviate the over-smoothing problem, in equation 7 and 8, we have already included residual connections [10] in updating rule. And we propose to further improve the problem by creating jumping connections [25].

$$h_i^{(final)} = h_i^{(0)} \oplus h_i^{(1)} \cdots \oplus h_i^{(t)} \tag{9}$$

4.5 Aggregation across different meta-path semantics

Dan:need some intuition/justification shit Given a node i, and its set of embeddings under different semantics, a metapath aggregator is

a function γ in the form of $y_i = \gamma_\Theta(\{x_{i,1}, x_{i,2}, ..x_{i,P}\})$, where $x_{i,p}$ is the embedding of node i under metapath p, and y_i is the aggregated embedding of node i in the HIN. Θ is the learnable paremeters of the aggregator. Here we propose several metapath aggregators.

• **Attention** Dan:cite The first option we consider is scaled dot-product attention introduced in Transformer.

$$\tilde{h}_i^{\Phi_p} = W_{m1} \cdot \tanh(W_{m2} \cdot h_i^{\Phi_p}) \tag{10}$$

$$\tilde{w}_i^{\Phi_p} = W_w \cdot \sigma(\tilde{h}_i^{\Phi_p}) \tag{11}$$

$$w_i^{\Phi_p} = \text{softmax}(\tilde{w}_i^{\Phi_p}) = \frac{\exp(\tilde{w}_i^{\Phi_p})}{\sum_q \exp(\tilde{w}_i^{\Phi_q})} \tag{12}$$

After the fully-connected self-attention, we propose to reduce across different meta-path semantics by **Sum** operation.

$$z_i = \text{ReLU}(\sum_p w_i^{\Phi_p} \cdot h_i^{\Phi_p}) \tag{13}$$

• **Gated linear unit** Inspired by forget gate mechanisms in recurrent networks (RNN) Dan:TODO:cite, we compute a soft gate between 0 (low importance) and 1 (high importance) to represent different importance to each embedding entry. We make the element-wise gated sum layer as follows:

$$o_i^{\Phi_p} = \sigma(W_o \cdot h_i^{\Phi_p}) \tag{14}$$

$$\tilde{h}_i^{\Phi_p} = \tanh(W_z \cdot h_i^{\Phi_p}) \tag{15}$$

Finally, we have

$$z_i = \sum_p o_i^{\Phi_p} \odot \tilde{h}_i^{\Phi_p} \tag{16}$$

The gated unit o_{Φ_p} with sigmoid output (ranged (0,1)) is used to control which elements are to be aggregated in the output.

• **Other aggregators** In preliminary experiments, **Concat**, **Max-Pooling** and **Mean** aggregators did not generate a better result, and we omit results of these aggregators in experiment section.

4.6 HINGCN: Semi-supervised classification on Heterogeneous Information Network via Graph Convolution Networks

After meta-path aggregation layer, the embeddings are fed to a 2-layer MLP for final output. In semi-supervised classification tasks, we minimize the Cross-Entropy loss and the model generates final prediction based on the largest output value.

$$\mathcal{L}(W) = -\sum_{l \in \mathcal{Y}_L} Y^l ln(z^l) + \lambda ||W||_2 \tag{17}$$

$\mathcal{Y}_L$ is the set of labeled nodes in training set. W represents the weight matrices used in the layers of our HINGCN. We use L_2 normalization for parameters involved.

To this end, we give a summary of overall process of HINGCN. The part of embedding generation is summarized in Algorithm 1. HINGCN first computes initial edge feature $r_{\Phi_p}^{(0)}$ for each edge under each meta-path. Then it applies an alternative update of node embedding and edge embedding. After that each node fuses a meta-path specific embedding by jumping connections. These embeddings are further aggregated using either meta-path level attention or gated linear units. Finally, embeddings are fed to a two layer MLP and a softmax layer for classification output.

Algorithm 1 HINGCN Embedding Generation

Input: HIN $G = (V, E)$; labeled type T; meta-path set $\Phi_0, \Phi_1, \ldots, \Phi_P$; an initial node representation $\{f_i, \forall x_i \in X_T\}$; number of aggregation layers L;
Output: Final node embeddings $Z = \{z_1, ..., z_i\}$

$h_i^{(0)} \leftarrow f_i, \forall x_i \in X_T$
for $p \leftarrow 0$ to P **do**
  Compose edge features $r_{\Phi_p}^{(0)}$ according to Eq.1.
end for
for $p \leftarrow 0$ to P **do**
  for $t \leftarrow 0$ to $L-1$ **do**
    for $x_i \in X_T$ **do**
      $q_i = W_{a1}^{(t)}(\tanh(W_{a2}^{(t)} h_i^{(t)}))$
      for $x_j \in N_i^{\Phi_p}$ **do**
        $k_{i,j} = W_{a3}^{(t)}(\tanh(W_{a4}^{(t)}(h_j^{(t)} \oplus r_{i,j}^{(t)})))$
        $v_{i,j}^{(t)} = W^{(t)}(h_j^{(t)} \oplus r_{i,j}^{(t)})$
        $\tilde{a}_{i,j}^{(t)} = q_i^T \cdot k_{i,j}$
        $a_{i,j}^{(t)} = \text{softmax}(\tilde{a}_{i,j}^{(t)})$
      end for
      $h_i^{(t+1)} = \text{ReLU}(W_{h1} \cdot \sum_{j \in N_i^{\Phi_p}} a_{i,j}^{(t)} \cdot v_{i,j}^{(t)} + W_{h2} \cdot h_i^{(t)})$
    end for
    for $e \in E$ **do**
      $(x_i, x_j) \leftarrow \text{Endpoints}(e)$
      $r_{i,j}^{(t+1)} = \text{ReLU}(W_{r1} \cdot (h_i^{(t)} \oplus h_j^{(t)} \oplus r_{i,j}^{(t)}) + W_{r2} \cdot r_{i,j}^{(t)})$
    end for
  end for
  $h_i^{\Phi_p} = h_i^{(0)} \oplus h_i^{(1)} \cdots \oplus h_i^{(t)}$
end for
Aggregate meta-path specific embeddings and compute z_i according to **Attention** (Eq.13) or **Gated Linear Units** (Eq.16)
return $Z = \{z_1, ..., z_i\}$

5 EXPERIMENT

We conducted extensive experiments to evaluate the performance of HINGCN. This section summarizes our results. We compare the various methods using three popular measures, namely, *accuracy*, *micro-F1*, and *macro-F1*. These measures evaluate clustering quality and their values range from 0 to 1, with a larger value indicating a better clustering quality.

5.1 Datasets

We use three datasets, namely DBLP [1], Yelp[2] and Freebase [3]. A summary of statistics of the datasets are shown in table 2.

[1] https://dblp.uni-trier.de/
[2] https://www.yelp.com/academic_dataset/
[3] https://www.freebase.com/

Table 2: Statistics of the datasets

Dataset	Relations	Number of A	Number of B	Number o A-B	Feature	Number of target nodes	Meta-paths
DBLP	Paper-Author	14328	4057		8789	4057	APA
							APAPA
	Paper-Conf	14328	20				APCPA
Yelp	Business-Review	2614	33360				-
	Review-Keyword	33360	82				BRKRB
	Review-User	33360	1286				BRURB
Freebase	Movie-Actor	1465	4019				MAM
	Movie-Director	1465	1093				MDM
	Movie-Writer	1465	1458				MWM

• **DBLP** : We extract a subset of DBLP which contains 4057 authors (*A*), 14328 papers (*P*) and 20 conferences (*C*). Authors are classified into four areas: *database*, *data mining*, *machine learning*, *information retrieval*. In addition, 8789 keyword terms are assigned to each author as feature, following tf-idf transformation. Links include A-P (author publishes a paper) and P-C (paper published at a conference). We consider meta-paths {*APA*, *APAPA*, *APCPA*} for experiments. The task of semi-supervised classification is to predict which area of research an author is focusing on. We obtained the ground truth from the dataset dblp-4area [19], which labels each author by his/her primary research area.
• **Yelp-Restaurant**: We extracted information related to restaurant businesses in YELP. From extracted information, we constructed a dataset which contains 2614 business objects (*B*); 33360 review objects (*R*); 1286 user objects (*U*) and 82 food relevant keyword objects (*K*). Restaurant businesses falls into three categories: "Fast Food", "Sushi Bars" and "American (New) Food". Each business object is associated with 3 categorical attributes which includes reservation (whether reservation is required), service (waiter service or self service) and parking; as well as 1 numerical attribute: review count; and 1 ordinal attribute: quality star. Links include B-R (business receives a review), U-R (user writes a review), K-R (keyword included in a review). We consider the meta-path set {*BRURB*, *BRKRB*}. The semi-supervised classification task is to classify business objects by state. We use the class information provided in the dataset as the ground truth.
• **Freebase-Movie**: We extract a subset of Freebase movie data which contains ? movies (*M*); ? actors (*A*); ? directors (*D*) and ? writers (*W*). The movies are divided into three classes: *Action*, *Comedy* and *Drama*. No attribute is provided with movies. Links include M-A (movie and its actor), M-D (movie and its director), M-W (movie and its writer). We consider meta-paths {*MAM*, *MDM*, *MWM*} for experiments. The task of semi-supervised classification is to predict the class of movies. We use the class information provided in the dataset as the ground truth.

5.2 Algorithms for comparison

We evaluate HINGCN and 9 other methods. To demonstrate effectiveness of our edge update mechanism and meta-path GLU layer, we also includes two variants of HINGCN.
• **Node2vec[8]**: A random walk based homogeneous graph embedding method. Here we ignore the heterogeneity of nodes and perform node2vec on the whole heterogeneous graph. The prediction is generated by feeding embeddings to a logistics classifier.
• **Metpath2vec[6]**: A random walk based heterogeneous graph embedding method. The algorithm performs meta-path specific random walks and generates embeddings for each meta-path. The prediction is generated by feeding embeddings to a logistics classifier. We test all meta-paths and report the best performance.
• **GCN[12]**: A semi-supervised graph convolution network that is designed for homogeneous graphs. Here we ignore the heterogeneity of nodes and perform GCN on the whole heterogeneous graph.
• **GAT[21]**: A semi-supervised graph neural network that utilizes attention mechanism on homogeneous graphs. Here we ignore the heterogeneity of nodes and perform GAT on the whole heterogeneous graph.
• **HAN[23]**: A hierarchical semi-supervised graph neural network on heterogeneous graphs. HAN employs node-level attention and meta-path level aggregation to capture node-level and semantic-level heterogeneity.
• **HINGCN**$_{ne}$: A variant of HINGCN which removes edge feature in the update process of node embeddings.
• **HINGCN**$_{nu}$: A variant of HINGCN which replace update of edge embedding by identity propagation.
• **HINGCN**$_{at}$: A variant of HINGCN which uses attention mechanism for meta-path level aggregation.
• **HINGCN**: The proposed semi-supervised classification method based on heterogeneous graph convolution networks.

5.3 Experiment setup

We implement HINGCN using PyTorch. We use the same learning rate 0.001 for all HINGCN variants. The models are trained with two NVIDIA 1080Ti GPUs using data parallelism. The batch size is 1024 for each GPU. We stack two graph aggregators, and each aggregator samples 16 neighbors for both training and testing. Each aggregator is then followed by a ReLU activation layer and a dropout layer with dropout rate 0.5. Output dimension of the layers are set to 64 and 32 respectively. For dataset without node features, we encode a 1-hot matrix as input feature. As mentioned in section 4.1, we ensemble edge feature from pre-trained path-sim and node embeddings. These node embeddings are trained using metapath2vec [6] with default parameters on the same graph and corresponding meta-paths. To make our experiment results repeatable, we release our datasets and implementation on Github with an

anonymous account[4]. For all baseline methods, we employ exactly the same training set, validation set and test sets for fairness. For GCNNs including GCN, GAT and HAN, we tune hyper-parameters of baseline methods on validation sets. For unsupervised random walk methods including Node2vec and Metapath2vec, we train the embedding using the whole graph, but only send training set embeddings to downstream logistics classifier. For these random walk methods, we set window size to 5, walk length to 100, walks per node to 1000, the number of negative samples to 5. For each experiment setting, we set 10 different random seed for dataset partition and report the average statistics.

5.4 Performance results

Here we present the experiment result of applying all Dan:10? methods on DBLP,Yelp and Freebase. We report the averaged *Macro-F1* and *Micro-F1* on Tab. 3.

As is shown in Tab. 3, HINGCN generally achieves the best results on all datasets. Surprisingly, although Metapath2vec was proposed for heterogeneous graphs, it cannot produce a better result than methods designed for homogeneous graphs. One possible reason is that during training process, only part of semantic (i.e. single meta-path) is considered. ESim, however, considers multiple meta-paths at the same time and gives a better result than Node2vec. Graph neural network methods utilize both graph structure information and node feature information, and generally perform best. Among these GCNN methods, we can see that highly engineered neural structures tailored for heterogeneous graph, e.g. HAN and HINGCN, usually perform better. Compared to GCN and GAT, HAN considers hierarchical semantic relationship instead of entire neighborhood of nodes in heterogeneous graphs. And HINGCN further includes edge features that are excluded from HAN model. Also, without edge features (HINGCN$_{ne}$) or without edge updates (HINGCN$_{ne}$), the performance is worse than HINGCN, indicating the effectiveness of proposed architecture. A comparison between HINGCN$_{at}$ and HINGCN suggest that GLU is indeed a better choice for meta-path level aggregation as suggested in section 4.5.

5.5 Analysis of HINGCN layers

We provide a detailed analysis of each component layer of HINGCN model.

Analysis of edge feature. As mentioned in section 4.1, before training of HINGCN, we fuse edge features as supplementary information. To justify its effectiveness, we make a variant of HINGCN by removing edge feature in the update process in Eq. 3. We name this variant as HINGCN$_{ne}$. From Fig. 3, we can see that compared to HINGCN, removal of edge feature results in a severe degradation of performance. This experiment result confirms our claim.

Analysis of effect of updating edge.

Analysis of different meta-path aggregation units.

5.6 Sensitivity Experiments of Hyper-parameters

In this section, we investigate the sensitivity of hyper-parameters. We vary concerned parameters and compare against default settings described in Sec. 5.3

Dimension of edge feature. We first analysis the effect of changing edge dimension in pre-processing. We use experiment to see whether HINGCN is sensitive to input feature dimensions. The result is shown in Fig. **??**.

Number of sampled neighbors.

Number of node update layers. In HINGCN, we adopt an alternative update scheme of node and edge embeddings. We would like to see if stacking more update layers would cause degradation of performance. The performance of HINGCN variants with or withour jumping connections is shown in Fig. **??**.

6 CONCLUSIONS

In this paper we studied the effectiveness of semi-supervised classification methods in handling heterogeneous information network data.

Future work: in this paper we propose to use hand-crafted edge features to enhance missing information in the shrinking homogeneous graph. One possible approach would be automatic learning of edge features to provide an end-to-end solution to heterogeneous graph embedding problems.

7 ACKNOWLEDGMENTS

This research is supported by Hong Kong Research Grants Council GRF HKU 17254016.

[4]http://github.com/TODO

Table 3: Quantitative results (%) on semi-supervised classification task

Datasets	Metrics	Training	Node2vec	Metapath2vec	GCN	GAT	HAN	HINGCN$_{ne}$	HINGCN$_{nu}$	HINGCN$_{at}$	HINGCN
DBLP	Macro-F1	10%	92.63	92.08							
		20%	93.97	93.21							
		30%	93.47	92.80							
		40%	93.84	92.14							
	Micro-F1	10%	92.86	92.36							
		20%	94.21	93.47							
		30%	93.72	93.10							
		40%	94.09	92.49							
Yelp	Macro-F1	10%		90.59							
		20%		91.08							
		30%		93.07							
		40%		93.07							
	Micro-F1	10%		89.48							
		20%		90.06							
		30%		92.16							
		40%		92.16							
Freebase	Macro-F1	10%		64.68							
		20%		66.10							
		30%		65.66							
		40%		65.35							
	Micro-F1	10%		70.53							
		20%		72.10							
		30%		72.39							
		40%		72.25							

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*.

[2] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. 2006. Manifold Regularization: A Geometric Framework for Learning from Labeled and Unlabeled Examples. *J. Mach. Learn. Res.* 7 (2006), 2399–2434.

[3] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral Networks and Locally Connected Networks on Graphs. In *ICLR*.

[4] Olivier Chapelle and Alexander Zien. 2005. Semi-Supervised Classification by Low Density Separation. In *AISTATS*. Society for Artificial Intelligence and Statistics.

[5] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *NIPS*. 3837–3845.

[6] Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable Representation Learning for Heterogeneous Networks. In *KDD*. ACM, 135–144.

[7] Tao-Yang Fu, Wang-Chien Lee, and Zhen Lei. 2017. HIN2Vec: Explore Meta-paths in Heterogeneous Information Networks for Representation Learning. In *CIKM*. ACM, 1797–1806.

[8] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *KDD*. ACM, 855–864.

[9] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NIPS*. 1024–1034.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR*. IEEE Computer Society, 770–778.

[11] Ming Ji, Yizhou Sun, Marina Danilevsky, Jiawei Han, and Jing Gao. 2010. Graph Regularized Transductive Classification on Heterogeneous Information Networks. In *ECML/PKDD (1) (Lecture Notes in Computer Science)*, Vol. 6321. Springer, 570–586.

[12] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR (Poster)*. OpenReview.net.

[13] Guohao Li, Matthias Müller, Ali K. Thabet, and Bernard Ghanem. 2019. Can GCNs Go as Deep as CNNs? *CoRR* abs/1904.03751 (2019).

[14] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper Insights Into Graph Convolutional Networks for Semi-Supervised Learning. In *AAAI*. AAAI Press, 3538–3545.

[15] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In *KDD*. ACM, 701–710.

[16] Jingbo Shang, Meng Qu, Jialu Liu, Lance M. Kaplan, Jiawei Han, and Jian Peng. 2016. Meta-Path Guided Embedding for Similarity Search in Large-Scale Heterogeneous Information Networks. *CoRR* abs/1610.09769 (2016).

[17] Lichao Sun, Lifang He, Zhipeng Huang, Bokai Cao, Congying Xia, Xiaokai Wei, and Philip S. Yu. 2018. Joint Embedding of Meta-Path and Meta-Graph for Heterogeneous Information Networks. In *ICBK*. IEEE Computer Society, 131–138.

[18] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. 2011. PathSim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks. *PVLDB* 4, 11 (2011), 992–1003.

[19] Yizhou Sun, Yintao Yu, and Jiawei Han. 2009. Ranking-based clustering of heterogeneous information networks with star network schema. In *KDD*. ACM, 797–806.

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NIPS*. 5998–6008.

[21] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR (Poster)*. OpenReview.net.

[22] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural Deep Network Embedding. In *KDD*. ACM, 1225–1234.

[23] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. 2019. Heterogeneous Graph Attention Network. In *WWW*. ACM, 2022–2032.

[24] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *ICLR*. OpenReview.net.

[25] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation Learning on Graphs with Jumping Knowledge Networks. In *ICML (Proceedings of Machine Learning Research)*, Vol. 80. PMLR, 5449–5458.

[26] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2016. Revisiting Semi-Supervised Learning with Graph Embeddings. In *ICML (JMLR Workshop and Conference Proceedings)*, Vol. 48. JMLR.org, 40–48.