

BHIN2vec: Balancing the Type of Relation in Heterogeneous Information Network

Seonghyeon Lee
sh0416@postech.ac.kr
Pohang University of Science and
Technology
Pohang, South Korea

Chanyoung Park
pcy1302@illinois.edu
University of Illinois at
Urbana-Champaign
Urbana, USA

Hwanjo Yu*
hwanjoyu@postech.ac.kr
Pohang University of Science and
Technology
Pohang, South Korea

ABSTRACT

The goal of network embedding is to transform nodes in a network to a low-dimensional embedding vectors. Recently, heterogeneous network has shown to be effective in representing diverse information in data. However, heterogeneous network embedding suffers from the imbalance issue, i.e. the size of relation types (or the number of edges in the network regarding the type) is imbalanced. In this paper, we devise a new heterogeneous network embedding method, called BHIN2vec, which considers the balance among all relation types in a network. We view the heterogeneous network embedding as simultaneously solving multiple tasks in which each task corresponds to each relation type in a network. After splitting the skip-gram loss into multiple losses corresponding to different tasks, we propose a novel random-walk strategy to focus on the tasks with high loss values by considering the relative training ratio. Unlike previous random walk strategies, our proposed random-walk strategy generates training samples according to the relative training ratio among different tasks, which results in a balanced training for the node embedding. Our extensive experiments on node classification and recommendation demonstrate the superiority of BHIN2vec compared to the state-of-the-art methods. Also, based on the relative training ratio, we analyze how much each relation type is represented in the embedding space.

CCS CONCEPTS

• **Theory of computation** → **Unsupervised learning and clustering**; • **Computing methodologies** → **Knowledge representation and reasoning**.

KEYWORDS

network embedding, heterogeneous network, random-walk strategy, multitask learning, inverse training ratio, stochastic matrix

ACM Reference Format:

Seonghyeon Lee, Chanyoung Park, and Hwanjo Yu. 2019. BHIN2vec: Balancing the Type of Relation in Heterogeneous Information Network. In *The*

*Corresponding author

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CIKM '19, November 3–7, 2019, Beijing, China

© 2019 Association for Computing Machinery.

ACM ISBN 978-1-4503-6976-3/19/11...\$15.00

<https://doi.org/10.1145/3357384.3357893>

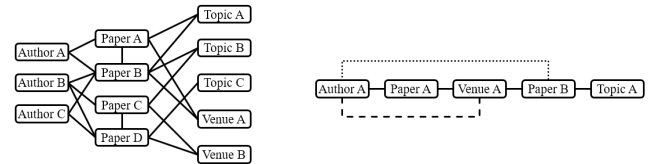


Figure 1: Citation network (left) and a random walk sampled from citation network (right). Solid line refers to explicit relation. Dashed line and dotted line refer to implicit relations using one intermediate node and two intermediate nodes, respectively.

28th ACM International Conference on Information and Knowledge Management (CIKM'19), November 3–7, 2019, Beijing, China. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3357384.3357893>

1 INTRODUCTION

Network embedding has been actively researched in the data mining field [4, 6, 8, 10, 14, 15, 20]. To apply machine learning techniques such as classification and regression [13] to network data, nodes in a graph are generally mapped to low-dimensional embedding vectors. These vectors are fed into various downstream tasks such as node classification, link prediction and visualization [4]. As a heterogeneous network can represent nodes and edges of different types within a single network, it has been widely used to represent diverse underlying information in real-world data. Citation network (Fig. 1) is a representative heterogeneous network, which typically consists of four types of nodes – author, paper, topic and venue [4, 6, 15, 19, 23]. These nodes are connected using relation types (i.e. edges in the network) such as authorship, citation, related topic, and published venue, and the size of each relation type (i.e. the number of edges regarding the type) is different.

Random walk, which is to sample a sequence of nodes from a network, has been widely adopted as a basic tool for extracting information from a network [8, 14]. Rooted at a node, the next node is repeatedly selected at random among its neighboring nodes, until a walk reaches a predefined length. Many network embedding methods [8, 14] use random walk to sample a sequence from a network because it preserves explicit and implicit relations inside the network. Explicit relation refers to a direct relationship between two nodes, and implicit relation refers to an indirect relationship between two nodes connected via some intermediate nodes. For example, in the random walk from the citation network (Fig. 1), the relationship between *Author A* and *Paper A* is explicit, whereas that

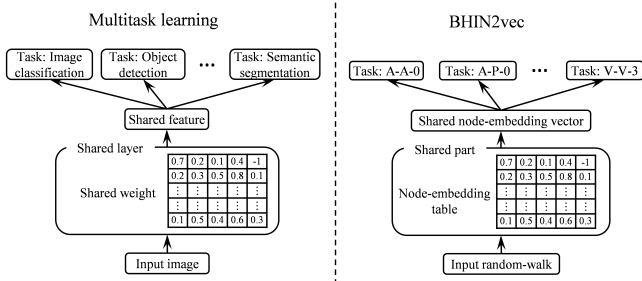


Figure 2: Overall diagram of the multitask learning model and BHIN2vec

between *AuthorA* and *VenueA* is implicit, as they are connected via *PaperA*.

However, random walk produces an imbalanced training of heterogeneous networks, because the major relation types take a large portion of training samples and thus dominate the training and minor relation types will hardly be learned. For example, in YAGO knowledge base, the number of explicit relations between *Person* and *Position* is 2,012, but that of explicit relations between *Person* and *Organization* is 886,529 (Refer to Table 1). Thus, the *Person – Position* relationship is unlikely to be captured by existing random walk strategies, as it is dominated by the *Person – Organization* relationship. Simply incorporating an equal number of samples for each relation type will not suffice either, because the learning difficulty is different for each relation type. Instead, we regulate the number of samples in a sampled random walk by considering the loss value which represents the learning difficulty. Recently proposed heterogeneous network embedding methods [6, 10] do not handle the imbalance problem, because they just use the conventional random walk or fix the ratio, by a hyperparameter, which determines whether to switch the relation type when sampling the next node in a random walk.

In this paper, we propose *Balanced Heterogeneous Information Network to Vector*, called BHIN2vec, to resolve the imbalance issue in the heterogeneous network. The core idea of BHIN2vec is to view the heterogeneous network embedding as a multi-task learning problem in that each task is derived from a relation type in a network. For example, the goal of a task "*Author – Author – 0*" is to maximize the similarity of an explicit relation between two authors (Fig. 2). We adopt GradNorm [3], which is a technique that employs inverse training ratio to solve the training imbalance issue in multi-task learning, to balance the training of different relation types in a network so as to generate a balanced embedding vector for each node. More precisely, we use inverse training ratio to first find the relation types that are less trained, i.e., relations that incur high loss values, and then generate random walks that contain more of the less-trained relations. In doing so, we propose a new random walk strategy that introduces a stochastic matrix to store the probabilities of choosing the next node type given the current one. The stochastic matrix is trained using inverse training ratio, and it eventually generates a random walk that contains more of the less-trained relations, and that considers all possible relation types.

BHIN2vec has three benefits compared to other network embedding methods. First, BHIN2vec trains all different relationships in a balanced way, and thus facilitates high quality representations for minor relations. Also, in the experimental section, we demonstrate that containing the minor relations in the embedding space actually helps represent the major relations. Second, BHIN2vec considers all possible relation types by introducing a stochastic matrix, which is a more elaborate way to handle type information within a heterogeneous network. Finally, BHIN2vec produces the inverse training ratio for each relation. This statistic can serve as a measure of how much each relation is mapped to the embedding space. By visualizing the stochastic matrix that stores the information of inverse training ratio, we can understand which type of relation is reflected well. To the best of our knowledge, this is the first work to leverage a new statistic apart from the training loss for heterogeneous network embedding.

The main contributions of our paper are summarized as follows:

- We propose a novel heterogeneous network embedding method that automatically handles the imbalance issue in a heterogeneous network.
- We provide the stochastic matrix trained from inverse training ratio to understand how much each relation is represented compared to other relations during training.
- We conduct node classification and recommendation task to demonstrate that our embedding vector outperforms the state-of-the-art methods.

The rest of this paper is organized as follow. In section 2 and 3, we explain the related work and define our problem and widely used notations. In section 4, we propose our method, BHIN2vec. In section 5, we report our experimental results and discuss them. Finally, we conclude our research in section 6.

2 RELATED WORK

2.1 Heterogeneous network embedding

The goal of network embedding is to transform nodes in a network into a embedding space while preserving their property. Two approaches exist for the network embedding: factorization based methods [25] and random walk based methods [8, 14]. We focus on the random walk based methods. The random walk based methods sample random walks from a network and maximize the similarity between two nodes contained in the random walks [8, 14]. Deepwalk [14] samples random walks without any constraint and other methods change the random walk strategy to capture the underlying structure in a network [8].

Recent heterogeneous network embedding methods consider the type in a heterogeneous network [4, 6, 10]. One approach is to use a meta-path analyzed by domain experts. Metapath2vec [4] samples random walks controlled by the meta-path. However, finding an appropriate meta-path is hard as the number of types increases. Therefore, methods that do not require a meaningful meta-path have emerged [6, 10]. HIN2vec [6] introduces a meta-path embedding table which saves the embedding vector for all possible meta-paths. These embedding vectors provide a different intensity for each dimension based on the corresponding meta-path. JUST [10] designs a new random walk strategy without meta-path.

They regulate the transition between types in a random walk using hyper-parameters.

2.2 Balance training for multi-task learning

Multi-task learning is to learn a single model that handles multiple tasks [1]. Multi-task models generate robust intermediate features because these features need to fit various tasks [2, 3, 24]. These features are fed into the task-specific layers to perform tasks. Task-specific losses are calculated from the tasks and model optimizes an aggregated loss to jointly learn them. In the recent success of deep learning, computer vision is the primary application area for multi-task learning, but it can be used in various fields such as natural language processing [18] and speech synthesis [21].

The balance among tasks is important when training a multi-task model [3, 11]. The characteristics for the tasks are diverse, which impedes the model to learn them compatibly. The irregular size of gradients can be one possible characteristic to inhibit training. The gradient size might be affected by the loss value and some large gradients can dominate the shared features. To resolve the imbalance, GradNorm [3] introduces inverse training ratio which represents how much training goes along and adjusts the gradient size proportional to the training ratio.

3 PROBLEM DEFINITION

We formally define three concepts that are widely used in this field: heterogeneous network, meta-network and network embedding.

Heterogeneous network. A heterogeneous network consists of four components, $G = (V, E, T, \phi)$. $V = \{v_i \mid i \in \mathbb{N}\}$ and $T = \{t_i \mid i \in \mathbb{N}\}$ refer to the node set and their type, respectively. $E = \{(v_i, v_j) \mid v_i \in V \wedge v_j \in V\}$ refers to the edge set. The mapping function, $\phi(\cdot) : V \rightarrow T$, indicates the type for each node.

Meta-network. Given a heterogeneous network $G = (V, E, T, \phi)$, a meta-network is defined as $G_{meta} = (V_{meta}, E_{meta})$. This network represents the relationship between node types in the heterogeneous network. The node for meta-network is the node type in the heterogeneous network, $V_{meta} = T$. Two node types are connected if there exist at least one corresponding connection in the network, $E_{meta} = \{(t_i, t_j) \mid \exists (v_x, v_y) \in E \wedge \phi(v_x) = t_i \wedge \phi(v_y) = t_j\}$.

Network embedding. Given a network G , the network embedding is to find a function $f(\cdot) : V \rightarrow \mathbb{R}^d$ which takes one node as input and gives an embedding vector for that node as output. The embedding vector captures the network structure.

4 METHOD

Our proposed method, BHIN2vec, consists of two parts: a skip-gram model and a biased random walk generator. We extend the skip-gram model and create a side-product, called inverse training ratio tensor. In doing so, we propose a new random walk strategy that uses a stochastic matrix. We connect our extended skip-gram model and the new random walk strategy by training the stochastic matrix using the inverse training ratio tensor. The conceptual diagram of BHIN2vec for citation network is illustrated in Figure 3.

4.1 BHIN2vec: Skip-gram model

Normal skip-gram model. The skip-gram model has a node embedding table $Q \in \mathbb{R}^{|V| \times d}$ which stores the d -dimensional embedding vector for all nodes [12, 14]. Our objective is to learn Q where $f(v_i) = Q[i]$. The model takes a sequence of nodes $w = (w_1, \dots, w_l)$ with length l , called walk, as an input. The skip-gram model maximizes the probability to predict context nodes using a source node. Choosing a source node v_i for a given walk, the context nodes for the v_i are the k nodes right behind the v_i in the walk. k is a predefined parameter, called the context window size. We calculate the inner product of the embedding vector for the source node and the embedding vector for the context node and apply the softmax function to get the probability to predict the context node given the source node. Using the negative log-likelihood loss, the skip-gram loss for one random walk w is calculated as

$$L = - \sum_{i=1}^l \sum_{j=1}^k \log p(w_{i+j} \mid w_i) = - \sum_{i=1}^l \sum_{j=1}^k \log \frac{e^{f(w_{i+j})^\top f(w_i)}}{\sum_{v_n} e^{f(v_n)^\top f(w_i)}}. \quad (1)$$

The denominator of the softmax function requires large computations, so we adopt the negative sampling approach to approximate that value [12]. We take m samples, called N_V , from V with their empirical distribution. Then, the above loss function is changed to

$$L = - \sum_{i=1}^l \sum_{j=1}^k \left(L_p(w_{i+j}, w_i) + \sum_{v_o}^{N_V} L_n(v_o, w_i) \right) \quad (2)$$

$$L_p(v_c, v_s) = \log \sigma(f(v_c)^\top f(v_s)) \quad (3)$$

$$L_n(v_c, v_s) = \log \sigma(-f(v_c)^\top f(v_s)), \quad (4)$$

where σ is the sigmoid function. We calculate the gradient of loss value with respect to the embedding table and update the embedding table to minimize the loss value.

Multi-task setting for heterogeneous network embedding. To apply the technique used in multi-task learning, we reconstruct heterogeneous network embedding as if we train multiple tasks simultaneously. First, we define virtual tasks using the relation type. The definition of virtual task J_{ijk} is to predict nodes which types are t_j given a node which type is t_i and the two nodes are connected via k intermediate nodes. Then, we construct a possible task set that contains the corresponding relation type which is used in the skip-gram model. Given a walk w , the skip-gram model uses the relations between two nodes where two nodes are connected via at most k nodes. Considering all combination of the source node's type and context node's type, the number of tasks that are used in the skip-gram model is $k \times |T| \times |T|$ where k is the context window size and $|T|$ is the number of types in a network. We reduce the size of possible tasks by removing the tasks that always not appear in any random walks. If an implicit relation between t_i and t_j via k nodes doesn't exist in meta-network, then the task J_{ijk} is always not contained in any walk. Finally, we define a possible task set $\mathcal{J}_{possible}$ which can be contained in a random walk.

$$\mathcal{J}_{possible} = \{J_{xyz} \mid (A^z)_{xy} > 0\} \quad (5)$$

$$A = \text{the adjacency matrix of } G_{meta} \quad (6)$$

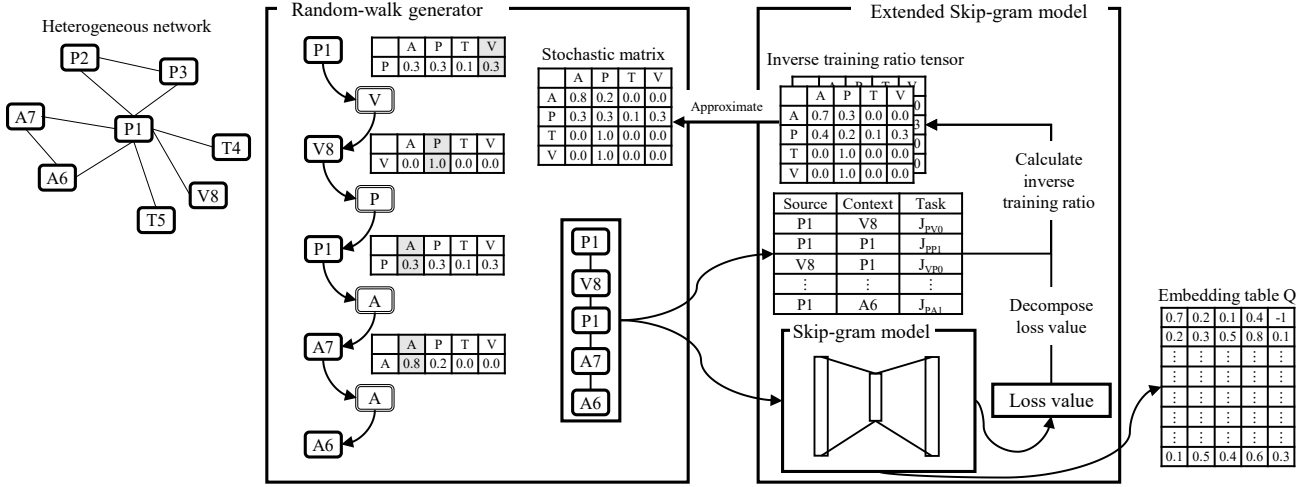


Figure 3: The conceptual diagram of BHIN2vec. We use a simple citation network. The node name consists of the node type and their index. The walk length is 5, the context window size is 2, the node dimension is 5. The random walk generator uses the corresponding row in the stochastic matrix to sample next type. In the extended skip-gram model, we split loss value based on the relation type. The inverse training ratio tensor is produced as a side-product and used for training the stochastic matrix in the random walk generator.

Balance in multitasks. To quantify the imbalance in virtual tasks, we formulate the loss value for each task and define inverse training ratio tensor. We split the L_p and L_n in Equation (3) and (4) for each virtual task and calculate the loss values.

$$L[J_{xyz}] = -\frac{\sum_{i=1}^l (L[J_{xyz}]_p(w_{i+z+1}, w_i) + \sum_{v_o}^{N_V} L[J_{xyz}]_n(v_o, w_i))}{\sum_{i=1}^l (\mathbb{I}[J_{xyz}](w_{i+z+1}, w_i) + \sum_{v_o}^{N_V} \mathbb{I}[J_{xyz}](v_o, w_i))} \quad (7)$$

$$L[J_{xyz}]_p(v_c, v_s) = \begin{cases} L_p(v_c, v_s) & \text{if } \phi(v_c) = t_y \wedge \phi(v_s) = t_x \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

$$L[J_{xyz}]_n(v_c, v_s) = \begin{cases} L_n(v_c, v_s) & \text{if } \phi(v_c) = t_y \wedge \phi(v_s) = t_x \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

$$\mathbb{I}[J_{xyz}](v_c, v_s) = \begin{cases} 1 & \text{if } \phi(v_c) = t_y \wedge \phi(v_s) = t_x \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

Also, some relation types are not contained in a random walk by chance. In this case, we take the previous loss calculated in the previous iteration as the surrogate value for the current one. For the loss value for each task, we take the mean to normalize the number of occurrence. These loss values represent which task doesn't fit in the embedding space compared to the others. Given the loss values, we define the inverse training ratios as

$$\tilde{L}[J_{xyz}](t) = L[J_{xyz}] / L_{initial}[J_{xyz}] \quad (11)$$

$$r[J_{xyz}](t) = \tilde{L}[J_{xyz}](t) / \mathbb{E}_{J_{possible}}[\tilde{L}[J](t)], \quad (12)$$

where $L_{initial}[J_{xyz}]$ is the initial loss when training starts [3].¹ $\tilde{L}[J_{xyz}](t)$ is the training ratio of J_{xyz} at time t , representing the amount of training that has been done. $r[J_{xyz}]$ is the relative

¹Because the initial loss is unstable, we take a theoretical loss value as the initial loss. Suppose that $p(w_{i+j}|w_i) = 0.5$ in the Equation (1), the initial loss value is $k \times (l-k) \times 0.6931$.

inverse of the training ratio. If $r[J_{xyz}](t) > 1$, the task J_{xyz} is premature compared to the other tasks. We arrange these inverse training ratios to a 3-dimensional tensor $I \in \mathbb{R}^{k \times |T| \times |T|}$, called inverse training ratio tensor. We set one for the tasks that always not occur in a random walk.

$$I_{zxy}(t) = \begin{cases} r[J_{xyz}](t) & \text{if } J_{xyz} \in J_{possible} \\ 1 & \text{otherwise} \end{cases} \quad (13)$$

Heterogeneous skip-gram model. We adopt two variants for the skip-gram model. We sample negative nodes which have the same type with positive node [4]. It gives stable loss value for each task. Also, we create a task embedding table $Q_R \in \mathbb{R}^{k \times |T| \times |T| \times d}$ and define function $f_R(k, t_i, t_j) = Q_R[k, t_i, t_j]$ which maps each task into d -dimensional vector. By multiplying the node embedding vector by the corresponding task embedding vector, relations will be embedded with different intensities for each dimension.

$$L_p(v_c, v_s) = \log \sigma \left(\left(\sqrt{r} \odot f(v_c) \right)^\top \left(\sqrt{r} \odot f(v_s) \right) \right) \quad (14)$$

$$L_n(v_c, v_s) = \log \sigma \left(- \left(\sqrt{r} \odot f(v_c) \right)^\top \left(\sqrt{r} \odot f(v_s) \right) \right) \quad (15)$$

$$r = f_R(k, \phi(v_c), \phi(v_s)) \quad (16)$$

4.2 BHIN2vec: Biased random walk generator

Random walk strategy. To provide a walk to the skip-gram model, we sample random walks from a network. Rooted at a node, we randomly select the next node from the adjacent nodes. Iterating this procedure l times, a walk with length l is generated from the network. To extract informative relations in the heterogeneous network, the random walk strategy needs to consider type. When sampling a random walk, a simple extension to consider type is to do an additional sampling for type. More precisely, we determine

Algorithm 1: BHIN2vecWalk

Input: Start node v , stochastic matrix P , Walk length l
Heterogeneous network $G = (V, E, T, \phi)$
Output: Random walk w
 $w[1] = v$;
for $i \leftarrow 1$ **to** $l - 1$ **do**
 $nxt_t = BiasedSample(T, P[\phi(w[i]), :]);$
 $target = \{v_j \mid (w[i], v_j) \in E \wedge \phi(v_j) = nxt_t\};$
 $w[i + 1] = UniformSample(target);$
end
return w

the type for the next node by sampling and do another sampling for the next node that has the sampled type. The remaining part of the random walk strategy is how to sample the next type and the next node.

Stochastic matrix. We introduce a stochastic matrix to do bias sampling when sampling the next type. The stochastic matrix $P \in \mathbb{R}^{|T| \times |T|}$ describes the transition probabilities from the current states to the next state [7]. In our case, the value $P_{ij} = p(t_j | t_i)$ in the stochastic matrix P is the probability to choose t_j for the next type when the current node type is t_i .

$$P_{ij} = p(t_j | t_i) \text{ such that } \sum_j P_{ij} = 1 \quad (17)$$

We set $p(t_j | t_i)$ to zero if no edge between t_i and t_j exists in the meta-network. So, this stochastic matrix can be viewed as a weighted adjacency matrix for the meta-network. We use the corresponding row of the current type in the stochastic matrix when sampling the next type. Also, note that the stochastic matrix represents the multi-hop type transition probability in a compact manner.

$$(P^k)_{ij} = p(t_j | t_i, k) \quad (18)$$

Based on this probability, we approximate the ratio for the implicit relation included in the sampled random walk. The pseudo-code for our biased random walk is illustrated in the Algorithm 1.

4.3 BHIN2vec: From inverse training ratio tensor to stochastic matrix

We train the stochastic matrix to store the information in the inverse training ratio tensor. We intend to sample more of less-trained relations so that the less-trained relations would be reflected more in the embedding space.

Perturbation approach. We perturb already existing stochastic matrix using inverse training ratio tensor. We use a uniform stochastic matrix $P_{uni} \in \mathbb{R}^{|T| \times |T|}$ as an existing solution. In the uniform stochastic matrix, the transition probability to target type is equal for each source type.

$$P_{uni_{xy}} = \begin{cases} \frac{1}{degree(t_x)} & \text{if } (t_x, t_y) \in E_{meta} \\ 0 & \text{otherwise} \end{cases} \quad (19)$$

Using the uniform stochastic matrix, we calculate the probability to move from t_i to t_j in k steps and perturb this probability using

Algorithm 2: BHIN2vec

Input: Heterogeneous network $G = (V, E, T, \phi)$
Latent dimension d , walk length l
Context window size k , negative sample size m
Epoch num e , learning rate r, r_2
Output: Node embedding matrix Q
Initialize $Q \in \mathbb{R}^{|V| \times d}$;
 $P = [[0, \dots, 0], \dots, [0, \dots, 0]] \in \mathbb{R}^{|T| \times |T|}$;
 $P[i, j] = 1$ for explicit edge;
for 1 **to** e **do**
 foreach $v \in V$ **do**
 $w = BHIN2vecWalk(v, P, l, G);$
 $L = SkipGram(w, k, m, Q);$
 $Q = Q - r \frac{\partial}{\partial Q} L;$
 Create $I \in \mathbb{R}^{k \times |T| \times |T|}$ using Equation (13);
 Calculate $L_{stochastic}$ using Equation (20);
 $P = P - r_2 \frac{\partial}{\partial P} L_{stochastic};$
 end
end
return Q

I_{kij} .

$$L_{stochastic} = \sum_{i=0}^{k-1} \left| P^{i+1} - \left(P_{uni}^{i+1} + \alpha (I_i - \mathbf{1}) \right) \right|_F^2 \quad (20)$$

The perturbation parameter α determines how much the perturbation is applied to the existing solution.

Update stochastic matrix. We calculate the gradient of this loss value with respect to the stochastic matrix and update the stochastic matrix to minimize the loss value. To preserve the property in the stochastic matrix, we only update nonzero values and clip the values between zero and one. Then, we normalize the row in the stochastic matrix.

In summary, given a random walk, the skip-gram model updates the embedding table and creates an inverse training ratio tensor as a side-product. Then, using the inverse training ratio tensor, the loss value for the stochastic matrix is calculated and used for training. After updating the stochastic matrix, we sample a new random walk. We alternately optimize the embedding table and the stochastic matrix by iterating this process until the training converges. The pseudo-code for the overall BHIN2vec procedure is illustrated in the Algorithm 2.

5 EXPERIMENTS

In this section, we create embedding vectors using BHIN2vec and analyze the result to understand the behavior of BHIN2vec. We focus on the following research questions.

- **RQ1:** Does our method get better node representations than other baselines?
- **RQ2:** Do the representations contain all different relation types in the heterogeneous network?
- **RQ3:** How do the hyper-parameters affect our method?

To answer **RQ1** and **RQ2**, we conduct two general tasks: multi-class node classification and recommendation using link prediction. To answer **RQ3**, we conduct the sensitivity analysis on the perturbation hyper-parameter α .

5.1 Dataset and Baseline

Dataset. We build up heterogeneous networks from the real-world data. All datasets can be accessed from the public websites.

- **BlogCatalog** is the social blog directory which manages the bloggers and their groups.² This dataset contains two types: user and group. Friendships between users and group membership exist in the network.
- **Douban** is a movie platform website. We used preprocessed version of this dataset.³ Movie, actor, director and user nodes compose this network and the nodes are connected with four relation types: an actor participates a movie; a director produces a movie; a user watches a movie; and a friendship between two users.
- **DBLP** is a website that manages the research publication.⁴ We used V10 for our experiment [17]. We build up a heterogeneous network with four type: author, paper, topic and venue. The topic is generated by splitting the title and abstract into words. Four relationships exist in the network: an author writes a paper; a paper references other paper; a paper contains a topic and a paper is published in a venue.
- **YAGO** is an open source knowledge base. We used preprocessed version for this experiment [16]. Seven types exist in the network: person, piece of work, prize, position, event, organization and location. Eight relation types exist in the network.

We removed nodes which degree is less than 2 because these nodes contain inadequate information. We summarize the statistic for each dataset in Table 1.

Baseline Method. We compare our method with other network embedding methods.

- **Deepwalk** [14] does a pioneering work by introducing the skip-gram model into network embedding task. This model samples random walks from a network and increases the similarity between the source nodes and context nodes.
- **LINE** [20] considers first and second order proximity in a network. Instead of sampling random walks, they directly optimize the similarity between nodes.
- **HIN2vec** [6] introduces a meta-path embedding table and jointly optimizes the node embedding table and the meta-path embedding table. They combine the node embedding vector with the meta-path embedding vector to control the intensity of each dimension for different relation types.
- **JUST** [10] adjusts the random walk strategy by introducing two hyper-parameters. Instead of using the meta-path, they sample random walks with the type transition probability and a queue that manages previously sampled types.

²<http://socialcomputing.asu.edu/datasets/BlogCatalog3>

³<https://bit.ly/2CDFI9z>

⁴<https://aminer.org/citation>

Table 1: Dataset statistic

Dataset	Node		Edge	
BlogCatalog	User	10312	U-U	267181
	Group	39	U-G	11581
Douban	User	12392	M-U	658412
	Movie	9322	U-U	3268
	Actor	5765	M-A	20400
	Director	2202	M-D	6646
DBLP	Author	133774	A-P	667340
	Paper	230356	P-P	996495
	Topic	119190	P-T	2463562
	Venue	165	P-V	219856
YAGO	PErson	346838	LO-LO	350005
	LOcation	134817	OR-LO	5776
	WORk	71852	EV-LO	9171
	ORganization	18716	PE-LO	177646
	EVEnt	5590	PE-OR	886529
	PRize	1711	PE-WO	298632
	POsition	301	PE-PO	2012
		PE-PR	69249	

We use the author codes except JUST because JUST doesn't publish their code.

Hyper-parameter tuning. To focus on the heterogeneous property in the network, we fix all parameters related to the homogeneous property. All methods require a walk length l , the number of epoch e , the node dimension d , the context window size k and the number of negative nodes per positive node m . We fix $l = 100$, $e = 10$, $k = 5$, $m = 5$, $d = 128$ for all methods. Heterogeneous network embedding methods introduce different hyper-parameters to handle the type. HIN2vec doesn't have additional hyper-parameters. JUST takes two hyper-parameters, α and the size of queue. We perform grid search on $\alpha = [0.25, 0.5, 0.75]$ and the size of queue = $[1, 2, 3]$. Our method takes two hyper-parameters, the perturbation parameter α and the learning rate for the stochastic matrix r_2 . We perform grid search on $\alpha = [0.05, 0.1, 0.2]$ and $r_2 = [0.25, 0.025, 0.0025]$.

5.2 Multi-class node classification

To answer **RQ1**, we measure the F1 score for the multi-class node classification task based on the node embedding table. Note that the purpose of this experiment is to evaluate the quality of the network embedding. We measure the performance of several node classification tasks and then determine the quality of network embedding based on their average. We conduct the multi-class node classification task on different node types.

Constructing label information. We create a label for DBLP network. For venue node, We adopt the same way as Metapath2vec [4] does. In Google Scholar⁵, we crawl top venues in each category and label venue nodes with their category. Then, we propagate the venue label to the paper nodes. We carefully determine the label

⁵<https://scholar.google.com/citations>

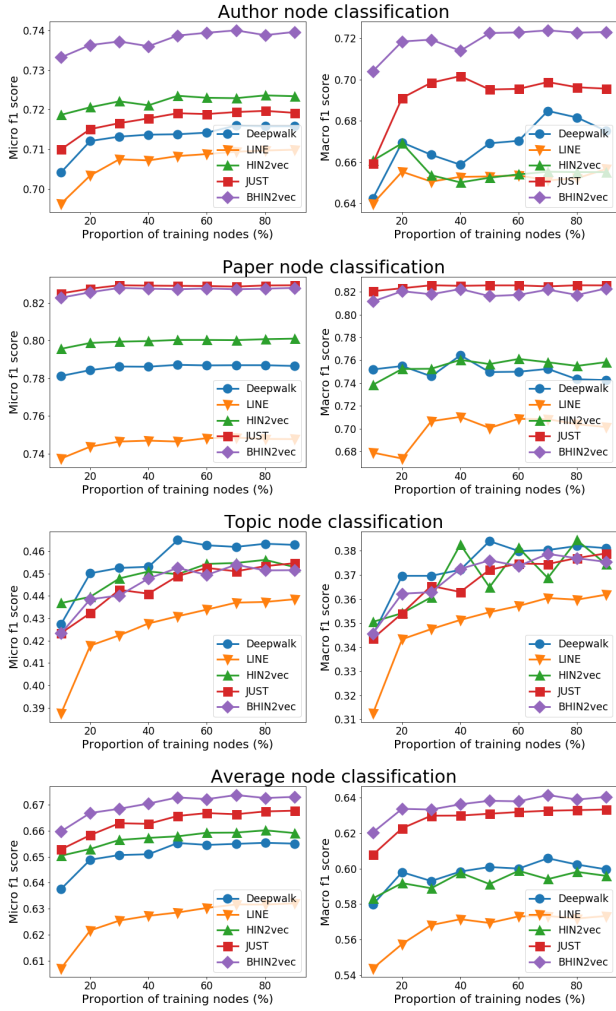


Figure 4: Node classification result on DBLP dataset

for paper nodes which have diverse information. For a given paper, we collect the labels for papers citing that paper. The labels of the papers cited by that paper are also collected separately. Then, we choose the label that contains the most in collected labels. Finally, we randomly sample one of the three labels, the label found in each label set and the venue label, to determine the label of a given paper. The labels of author and topic nodes are determined by random sampling among the labels of connected papers.

Evaluation protocol. We classify the nodes with given labels. First, we train a node embedding table using network embedding methods. Then, we choose the embedding vectors for the labeled nodes. We randomly choose 20 percent of the labeled nodes to construct the test set and the remaining labeled nodes are used to construct the training set. We train the logistic regression classifier using the training set and evaluate the micro F1 and macro F1 score using the test set. Also, we conduct this experiment with different

size of the training nodes. We report the average score from 10 repeated trials.

Result. We reports the micro F1 and macro F1 values on *Author*, *Paper*, *Topic* and its average in Figure 4. This experimental result has three noticeable points. First, our proposed method, BHIN2vec, gives better average micro F1 and macro F1 scores than the other methods do, which is the answer for **RQ1**. BHIN2vec reports comparable performance on *Topic* nodes, and gives better micro F1 and macro F1 scores on *Author* and *Paper* nodes than that of the other methods. Second, BHIN2vec prevents some relationships from dominating the embedding space. In DBLP network, the *Paper – Topic* relation is most common. Deepwalk [14] and HIN2vec [6] optimize that relation in the network. However, these methods does not optimize other minor relations inside the network, which results in the low F1 score in *Author* and *Paper* node classification task. Lastly, JUST [10] achieves comparable F1 scores in *Topic* nodes, but reports low F1 scores in *Author* nodes because that method does not consider the *Paper – Topic* relations and *Paper – Author* relations separately. However, BHIN2vec distinguishes *Paper – Author* relations and *Paper – Topic* relations, which results in the improvement in *Author* node classification. To understand how this mechanism works, we visualize the transition probability from paper to the other types in Figure 6. In the early stage of training, BHIN2vec samples equal number of all possible relations in the random walks. As the *Paper – Topic* relation becomes well reflected in the embedding space, the stochastic matrix is trained with this information and the transition probability for that relation decreases. Therefore, a random walk generated from BHIN2vec contains all possible relation type proportional to the loss value, which gives balanced embedding vectors.

5.3 Recommendation using link prediction

To answer **RQ2**, we conduct recommendation tasks for all relation types in various heterogeneous networks.

Evaluation protocol. Given a heterogeneous network, we remove 20% of total edges for the test set. With 80% of remaining edges, called the training set, we train the node embedding table using several network embedding methods. We perform recommendation tasks for all relation types in a network. A recommendation task recommends a target node for a source node. For example, we recommend a director for a movie. We add negative examples to the training set by sampling same number of arbitrary node pairs for each relation from the network. Using the node pairs and edge, we create the edge embedding vectors by applying Hadamard function to two node vectors in the training set [6]. We train a logistic regression classifier using the edge embedding vectors and evaluate the hit rate at 10 for the recommendation task, which is widely used in the recommendation field [5, 9, 22]. Given an edge (v_i, v_j) in test set, we sample 99 nodes in which the sampled node type is same as the type of v_j . Note that the node pair created with v_i doesn't occur in both the training set and the test set. Given 99 pairs of nodes and 1 edge, we create 100 edge embedding vectors and rank them using the trained classifier. If the edge is in the top-10 ranking list, we regard this result as successfully recommending the target node and increase the hit count. We apply this process to all edges

Table 2: Hit Rate @ 10 on Douban and BlogCatalog

Dataset	Douban								BlogCatalog			
Source node	User		Movie	Actor	Director	Actor	Director	Average	User		Group	Average
Target node	User	Movie	User	Actor	Director	Movie	Movie		User	User	User	
Deepwalk	0.4630	0.2100	0.1200	0.4188	0.5607	0.3883	0.5060	0.3810	0.2079	0.3733	0.4414	0.3409
LINE	0.2577	0.7142	0.5312	0.3672	0.4021	0.2694	0.2993	0.4059	0.4821	0.3734	0.2399	0.3651
HIN2vec	0.3751	0.7790	0.6567	0.5374	0.5590	0.4169	0.4356	0.5371	0.4794	0.4260	0.4109	0.4388
JUST	0.6045	0.4198	0.2664	0.5454	0.6365	0.5051	0.6107	0.5126	0.4631	0.3129	0.2992	0.3584
BHIN2vec	0.6392	0.7925	0.6485	0.6277	0.7334	0.5865	0.7154	0.6776	0.6531	0.4314	0.3709	0.4851

Table 3: Hit rate @ 10 on YAGO

Source	Target	Deepwalk	LINE	HIN2vec	JUST	BHIN2vec
OR	LO	0.4874	0.9253	0.9732	0.8914	0.9770
LO	OR	0.8620	0.3290	0.8808	0.7311	0.8196
EV	LO	0.8491	0.9612	0.9698	0.9612	0.9819
LO	EV	0.7935	0.7027	0.8912	0.6649	0.8973
LO	LO	0.8866	0.7961	0.9637	0.9306	0.9725
PE	OR	0.9453	0.5261	0.9917	0.9832	0.9936
OR	PE	0.9786	0.3244	0.9869	0.9715	0.9861
PE	PR	0.8103	0.7751	0.9503	0.9445	0.9654
PR	PE	0.9094	0.5611	0.8795	0.9113	0.9232
PE	WO	0.7524	0.3262	0.8922	0.6490	0.8404
WO	PE	0.9293	0.8367	0.9760	0.7903	0.9370
PE	PO	0.6667	0.6667	0.6290	0.7333	0.7351
PO	PE	0.8409	0.6250	0.8364	0.8409	0.8886
PE	LO	0.5175	0.8462	0.9315	0.7679	0.9116
LO	PE	0.6653	0.2257	0.6693	0.4601	0.6603
Average		0.7930	0.6285	0.8948	0.8154	0.8993

in the test set and we report the proportion of hit. We report the average score from 5 repeated trials.

Results. The hit rate at 10 on Douban and BlogCatalog dataset is reported in the Table 2. We report the average score of all tasks to measure the overall quality of the embedding vectors. BHIN2vec gives high hit rates at 10 in overall relations, whereas other methods cannot learn all relation types in the heterogeneous network evenly. This result empirically shows that BHIN2vec contains all different relation types in one embedding vector, which is the answer for RQ2. Other methods focus on a specific relation. In the Douban network, Deepwalk [14] focuses on the *Movie – Director* relation and LINE [20] focuses on the *User – Movie* relation. HIN2vec [6] represents the *User – Movie* relations well because those relations are contained mostly in that network. BHIN2vec learns not only the *User – Movie* relations but also other minor relations well, which acts as side information for embedding the *User – Movie* relation in the embedding space. Therefore, the hit rate for that relation is better than HIN2vec does. JUST [10] reports low hit rate in the *User – Movie* relations. We conclude that JUST [10] uses uniform probability when jumping to other types, so random walks cannot contain enough number of *User – Movie* relations to train. BHIN2vec considers the loss value for all possible relation types

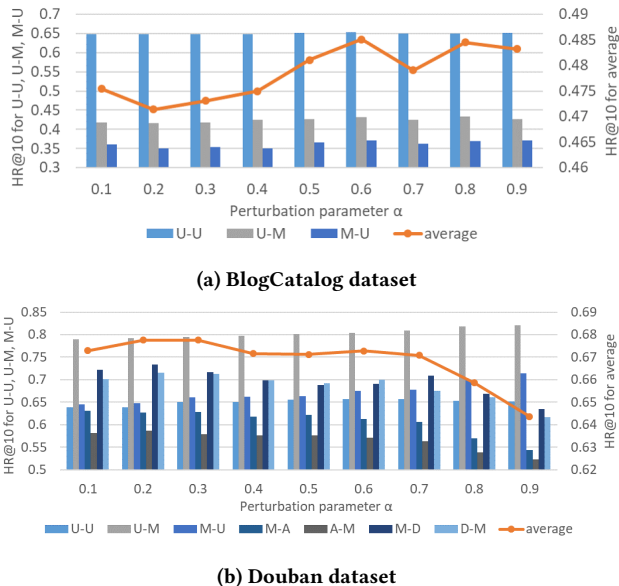


Figure 5: Hit rate scores with varying perturbation hyper-parameter α . Each bar represents a recommendation task. The source node type and target node type for a task are connected with a hyphen.

and focuses on *User – Movie* relations which report high loss value. We visualize the transition probability from *Movie* type to other types (Fig. 6). The probability from *Movie* to *User* type increases, which means that *User – Movie* relations report a high loss value compared to other relations. So, BHIN2vec reports high hit rate score by including enough *User – Movie* relations in the random walks to train.

In BlogCatalog, BHIN2vec improves both the *User – User* task and the *User – Group* task in a balanced way. However, BHIN2vec reports low hit rate for *Group – User* task because if the *User – User* relations are well represented in the embedding space, the group can not distinguish between two connected users. Therefore, *User – User* relations and *User – Group* relations show a tradeoff. We conclude that BHIN2vec finds the equilibrium point in this situation.

To scale up network size, we report the hit rate on YAGO network in the Table 3. BHIN2vec gives the better average hit rate

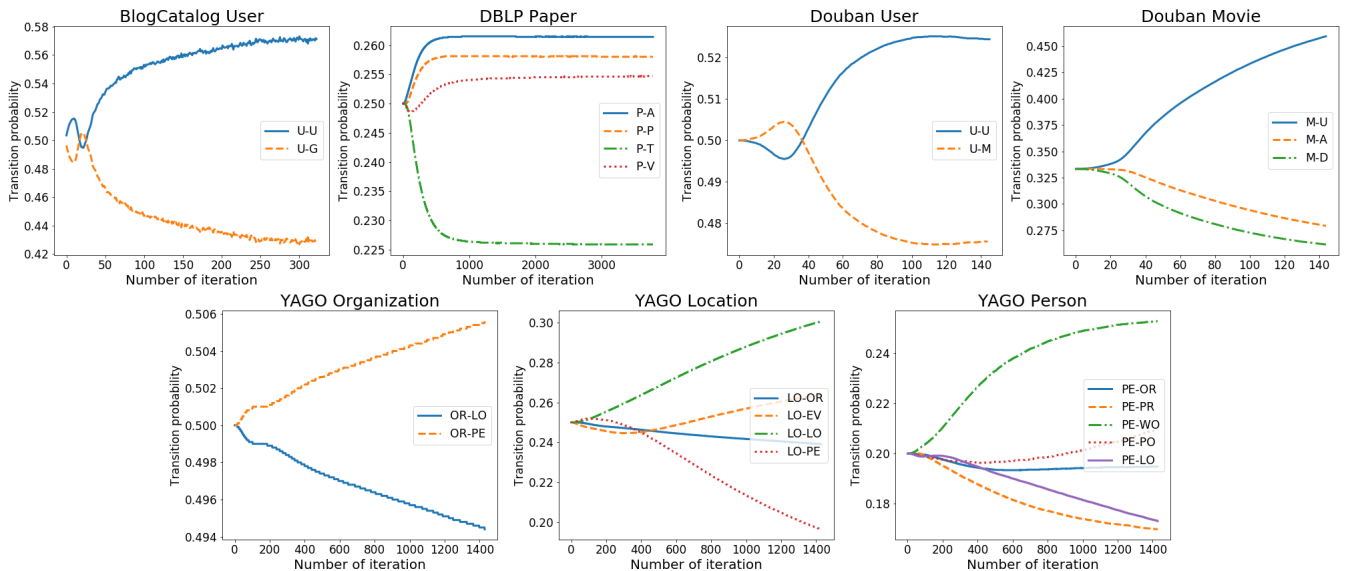


Figure 6: Transition probabilities in the stochastic matrix. We report the transition probabilities that give the highest average micro F1 score for DBLP and the highest average hit rate score for the other datasets.

than the other methods. Note that BHIN2vec embeds both the *Person – Position* relations, which is representative major relation type, and the *Person – Organization* relations, which is representative minor relation type, well in the same embedding space. It means that we resolve the imbalance issue in the heterogeneous network. However, the hit rates are dropped for some tasks. The *Person – Work* relations and the *Person – Location* relations are much more complicated than the other relations because the number of *Work* and *Location* nodes are larger than the number of *Organization* nodes. Therefore, to handle the network which has many relation types with extremely different complexities in a balanced way will be our future work.

Sensitivity Analysis. To answer RQ3, we investigate the impact of perturbation parameter α on recommendation task. In the Figure 5, we report the hit rate with different perturbation parameter α . In the BlogCatalog network, the average hit rate at 10 increases as the α increases. So, instead of using a uniform type transition probability, creating a random walk that contains the relation types proportional to the task loss is more effective to embed a network. In the Douban network, we observe that the major relations, i.e. *User – Movie* relations, increase as α increases. In the Figure 6, a random walk contains the *User – Movie* relations more. This trend is intensified as α increases, which results in the improvement of corresponding recommendation tasks. However, we also observe that the hit rate at 10 for minor relations decreases as α increases because *User – Movie* relation breaks the similarity between other minor relations. In summary, as α increases, BHIN2vec creates embedding vectors that contains overall relationships in a balanced way, but for large networks, the tradeoff between some relation types exists so that we recommend to use small α to protect the minor relation types.

6 CONCLUSION

We observe the imbalance issue in the heterogeneous network and resolve this issue by designing a new heterogeneous network embedding method. To balance in all possible relation types, we focus on the relation types that are less trained in the embedding space. To quantify how much each relation type is trained, we introduce the idea in multi-task learning. We define virtual tasks in that each task represents each relation type in the heterogeneous network. Then, we calculate the loss values for each virtual task by splitting the skip-gram loss and compute the inverse training ratios which represent how much each relation type is embedded. To focus on the tasks which report high loss value, we propose a new random-walk strategy that samples a random walk that contains more of less-trained relations. For the compact representation, we introduce the stochastic matrix in the random-walk strategy and train that stochastic matrix to store the information in the inverse training ratio.

We demonstrate that BHIN2vec produces node embeddings that contains all possible relation types evenly. We use our node embeddings to conduct two general tasks: node classification and recommendation. In node classification, we evaluate the micro F1 and macro F1 score for all node types. Our node embeddings give better F1 scores for all node types. Especially, our node embeddings give better F1 scores in *Author* node classification, which addresses the importance of considering all possible relation types. Also, visualizing the stochastic matrix, we understand the mechanism of BHIN2vec. In recommendation, we evaluate the hit rate at 10 in three different heterogeneous networks. BHIN2vec improves the hit rate at 10 in overall relation types in three different networks. In YAGO, BHIN2vec successfully embeds both the major relation type, i.e. *Person – Organization* relations, and minor relation type, i.e. *Person – Position* relations, at the same time. Also, we observe

the tradeoff between complicated relation types, which will be our future work.

ACKNOWLEDGMENTS

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (MSIT) (No. 2016R1E1A1A01942642)

REFERENCES

- [1] Rich Caruana. 1997. Multitask learning. *Machine learning* 28, 1 (1997), 41–75.
- [2] Jianhui Chen, Jiayu Zhou, and Jieping Ye. 2011. Integrating low-rank and group-sparse structures for robust multi-task learning. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 42–50.
- [3] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multi-task Networks. In *International Conference on Machine Learning*. 793–802.
- [4] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 135–144.
- [5] Travis Ebesu, Bin Shen, and Yi Fang. 2018. Collaborative memory network for recommendation systems. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 515–524.
- [6] Tao-yang Fu, Wang-Chien Lee, and Zhen Lei. 2017. Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 1797–1806.
- [7] Paul A Gagniuc. 2017. *Markov chains: from theory to implementation and experimentation*. John Wiley & Sons.
- [8] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 855–864.
- [9] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 173–182.
- [10] Rana Hussein, Dingqi Yang, and Philippe Cudré-Mauroux. 2018. Are Meta-Paths Necessary?: Revisiting Heterogeneous Graph Embeddings. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 437–446.
- [11] Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7482–7491.
- [12] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.
- [13] Kevin P Murphy. 2012. *Machine learning: a probabilistic perspective*.
- [14] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 701–710.
- [15] Yu Shi, Huan Gui, Qi Zhu, Lance Kaplan, and Jiawei Han. 2018. Aspem: Embedding learning by aspects in heterogeneous information networks. In *Proceedings of the 2018 SIAM International Conference on Data Mining*. SIAM, 144–152.
- [16] Yu Shi, Qi Zhu, Fang Guo, Chao Zhang, and Jiawei Han. 2018. Easing embedding learning by comprehensive transcription of heterogeneous information networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2190–2199.
- [17] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-june Paul Hsu, and Kuansan Wang. 2015. An overview of microsoft academic service (mas) and applications. In *Proceedings of the 24th international conference on world wide web*. ACM, 243–246.
- [18] Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Vol. 2. 231–235.
- [19] 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. *Proceedings of the VLDB Endowment* 4, 11 (2011), 992–1003.
- [20] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1067–1077.
- [21] Zhizheng Wu, Cassia Valentini-Botinhao, Oliver Watts, and Simon King. 2015. Deep neural networks employing multi-task learning and stacked bottleneck features for speech synthesis. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 4460–4464.
- [22] Hongzhi Yin, Hongxu Chen, Xiaoshuai Sun, Hao Wang, Yang Wang, and Quoc Viet Hung Nguyen. 2017. SPTF: a scalable probabilistic tensor factorization model for semantic-aware behavior prediction. In *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, 585–594.
- [23] Xiao Yu, Quanquan Gu, Mianwei Zhou, and Jiawei Han. 2012. Citation prediction in heterogeneous bibliographic networks. In *Proceedings of the 2012 SIAM International Conference on Data Mining*. SIAM, 1119–1130.
- [24] Yu Zhang and Qiang Yang. 2017. An overview of multi-task learning. *National Science Review* 5, 1 (2017), 30–43.
- [25] Ziwei Zhang, Peng Cui, Xiao Wang, Jian Pei, Xuanrong Yao, and Wenwu Zhu. 2018. Arbitrary-order proximity preserved network embedding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2778–2786.