Disentangling Degree-related Biases and Interest for Out-of-Distribution Generalized Directed Network Embedding

Hyunsik Yoo
Hanyang University
Seoul, Korea
hsyoo32@hanyang.ac.kr

Yeon-Chang Lee
Georgia Institute of Technology
Georgia, United States
yeonchang@gatech.edu

Kijung Shin
KAIST
Seoul, Korea
kijungs@kaist.ac.kr

Sang-Wook Kim*
Hanyang University
Seoul, Korea
wook@hanyang.ac.kr

ABSTRACT

The goal of *directed network embedding* is to represent the nodes in a given directed network as embeddings that preserve the asymmetric relationships between nodes. While a number of directed network embedding methods have been proposed, we empirically show that the existing methods lack out-of-distribution generalization abilities against degree-related distributional shifts. To mitigate this problem, we propose **ODIN** (Out-of-Distribution Generalized Directed Network Embedding), a new directed NE method where we model multiple factors in the formation of directed edges. Then, for each node, ODIN learns multiple embeddings, each of which preserves its corresponding factor, by disentangling interest factors and biases related to in- and out-degrees of nodes. Our experiments on four real-world directed networks demonstrate that disentangling multiple factors enables ODIN to yield out-of-distribution generalized embeddings that are consistently effective under various degrees of shifts in degree distributions. Specifically, ODIN universally outperforms 9 state-of-the-art competitors in 2 LP tasks on 4 real-world datasets under both identical distribution (ID) and non-ID settings. The code is available at https://github.com/hsyoo32/odin.

CCS CONCEPTS

• **Information systems → Social networks**.

KEYWORDS

network embedding; degree-related distributional shifts

ACM Reference Format:
Hyunsik Yoo, Yeon-Chang Lee, Kijung Shin, and Sang-Wook Kim. 2023. Disentangling Degree-related Biases and Interest for Out-of-Distribution Generalized Directed Network Embedding. In *Proceedings of the ACM Web Conference 2023 (WWW '23), May 1–5, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3543507.3583271

1 INTRODUCTION

Background. *Network embedding* (NE) aims to represent nodes in a given network as low-dimensional vectors (*i.e.*, embeddings) that preserve the structural properties of the network [3–5, 27, 38]. The learned embeddings can be used as informative features in various applications, such as link prediction [10, 25] and recommendation [17, 26]. In recent studies, additional information (*e.g.*, edge directions [37, 42], edge signs [15, 22], and knowledge bases [21, 24]) has been incorporated to improve the accuracy of NE.

In this paper, we focus on NE that utilizes *edge directions*. On a directed network, a directed edge from node v_i to v_j expresses an asymmetric relationship between the two nodes. For example, a hyperlink from one web page to another does not necessarily imply that a hyperlink in the opposite direction exists. In order to capture such an asymmetric relationship, most *directed network embedding* (DNE) methods distinguish the source node v_i and the target node v_j according to their roles in the edge. Then, they learn the v_i's source embedding and the v_j's target embedding in such a way to preserve their properties as a source and a target, respectively.

Challenges. The existing DNE methods lack *out-of-distribution* (OOD) generalization abilities [9, 30, 40] against degree-related distributional shifts. Specifically, they are designed on the assumption that the *degree distributions* of the training and test data for downstream tasks are *identical*. For example, for a link prediction task, the edges in an input network are assumed to be split into training and test data so that their in- and out-degree distribution becomes almost identical to each other.

However, in real-world scenarios, degree-related distributional shifts occur frequently for reasons such as the temporal/spatial evolution of the network, ruining the identical distribution (ID) assumption. During such an evolution, it is well-known as *preferential attachment* that the degrees of high-degree nodes tend to increase faster than those of lower-degree nodes [1, 11–13]. According to [1, 6], however, it is also common that dominant hubs are overtaken by *"new kids on the block" with higher fitness* (*i.e.*, intrinsic ability of a node to attract new edges). Therefore, it is important for a DNE method to take OOD generalization into consideration so that it can be robust to degree-related distributional shifts.

As yet, to the best of our knowledge, it has not been discussed how degree-related distributional shifts affect the accuracy of DNE methods on downstream tasks. In this work, we carefully examine the link prediction accuracy of the existing methods in both ID and non-ID settings. Specifically, we consider non-ID settings where the in- and out-degree distributions of training and test data are different (see Section 4 for details). As shown in Figure 1, the accuracies of all methods significantly degrade in the non-ID settings compared to the ID settings. The results indicate that the existing methods learn their embeddings that can hardly be generalized to the test data with different degree distributions.

*Corresponding author.

WWW '23, May 1–5, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9416-1/23/04…$15.00
https://doi.org/10.1145/3543507.3583271

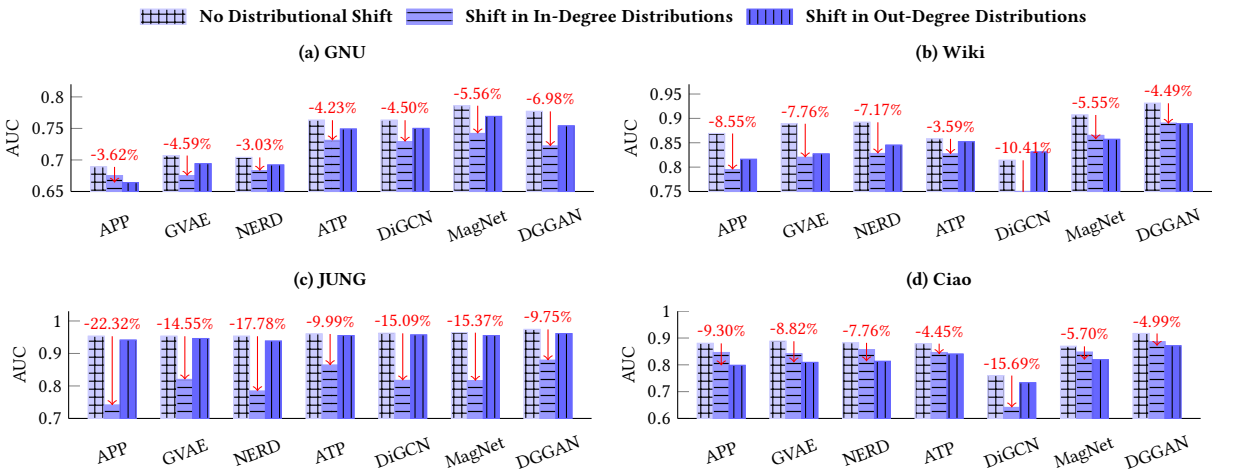

Figure 1: The out-of-distribution (OOD) generalization problem in directed network embedding (DNE). The link-prediction accuracies of state-of-the-art DNE methods significantly degrade when degree distributions in the training and test data are not identical, indicating their lack of OOD generalization abilities. See Section 4 for detailed experimental settings.

Proposed Ideas. In this work, we propose a new DNE method, Out-of-Distribution Generalized Directed Network Embedding (**ODIN**). ODIN models and exploits biases related to node degrees for robustness against distributional shifts in in- and out-degree distributions of the input network. By designing ODIN, we aim to answer these relevant research questions: (1) *How to model the formation of each directed edge?* (2) *How to leverage such modeled factors for learning OOD generalized embeddings?*

Regarding the first question, we define six node factors that can influence the formation of a directed edge from v_i to v_j. The six node factors are grouped into three: (a) the *authority* factors model a bias related to the target's in-degree, (b) the *hub* factors model a bias related to the source's out-degree, and (c) the *interest* factors model v_i's pure interest in forming an edge to v_j after removing degree-related biases [14]. We leverage all the factors to separately model biases and interests from interactions between nodes in a directed network. By doing so, we can mitigate the biases above when degree-related distributional shifts occur (*e.g.*, biases in the training data no longer significantly affect the edge formation).

Regarding the second question, we learn multiple factor embeddings, each of which captures the characteristics of its corresponding factor. Specifically, ODIN represents a node as six factor sub-embeddings that indicate the authority, hub, and interest factors as a source and a target. Then, ODIN learns the disentangled source and target embeddings per node via the following three objectives: (O1) to preserve *asymmetric proximity* between nodes in an input network; (O2) to disentangle the hub factor from the other two factors; (O3) to disentangle the authority factor from the other two factors. It is worth noting that through (O2) and (O3), we can naturally infer the interest factor from the other two factors, thus disentangling all three factors.

Why does ODIN work? By pursuing these multiple objectives, the final embeddings, where the six sub-embeddings are concatenated, preserve structural proximity between nodes and at the same time capture the characteristics of six node factors. That is, we jointly learn (bias-aware) hub/authority and (bias-free) interest embeddings. If degree-related biases entirely disappear in test data, it could be beneficial to use interest embeddings only. However, in realistic scenarios, degree-related-biases remain but the degree of them shifts over time. Also, it is not trivial to predict accurately the degree of biases in the future. For this reason, we leverage all-factor-embeddings jointly to achieve high accuracy in any scenario. We confirm experimentally that ODIN mitigates the OOD generalization problem against degree-related distributional shifts.

Contributions. Our contributions are summarized as follows:

- **Observation.** To the best of our knowledge, we are the first who study the out-of-distribution generalization problem against degree-related distributional shifts on DNE. We show empirically that the accuracies of existing DNE methods significantly degrade under degree-related distributional shifts.
- **Effective Algorithm.** We propose ODIN, where we model multiple factors in the formation of directed edges. By learning disentangled embeddings for each factor, it performs DNE with an excellent out-of-distribution generalization ability against degree-related distributional shifts.
- **Extensive Experiments.** We validate the effectiveness of ODIN by comparing it with 9 competitors on four real-world datasets under various degrees of shifts in degree distributions. The embeddings obtained by ODIN lead to **up to 5.18% more accurate link prediction** compared to the best one among those obtained by the state-of-the-art competitors.

The rest of this paper is organized as follows. In Section 2, we review previous studies on NE and OOD generalization problems. In Section 3, we present our proposed method in detail. In Section 4, we validate the effectiveness of the proposed method through extensive experiments. Finally, we conclude the paper in Section 5.

2 RELATED WORKS

NE methods. The earliest NE methods, including DeepWalk [27] and Node2Vec [5], train a shallow encoder, which is simply an

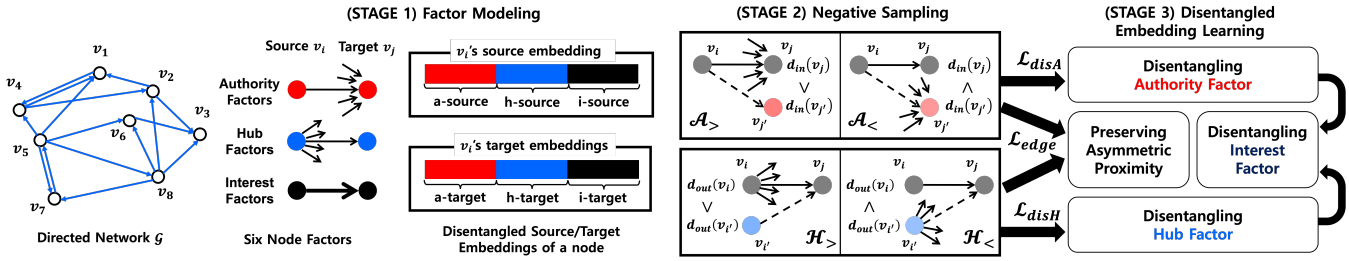

Figure 2: Overview of ODIN, which consists of three stages: (1) factor modeling, (2) direction- and degree-based negative sampling, and (3) disentangled embedding learning.

Table 1: Notations used in this paper.

Notation	Description
$\mathcal{G}$	Input directed network
$\mathcal{V}, \mathcal{E}$	Sets of nodes and edges
(v_i, v_j)	Directed edge from the node v_i to the node v_j
$\mathsf{s}_i, \mathsf{t}_i$	Source and target embeddings of the node v_i
$\mathsf{a}_i^{src}, \mathsf{a}_i^{tar}$	Sub-embeddings of v_i for the a-source and the a-target
$\mathsf{h}_i^{src}, \mathsf{h}_i^{tar}$	Sub-embeddings of v_i for the h-source and the h-target
$\mathsf{i}_i^{src}, \mathsf{i}_i^{tar}$	Sub-embeddings of v_i for the i-source and the i-target
$s_{ij}^{auth}, s_{ij}^{hub}$	Authority-factor and hub-factor scores for (v_i,v_j)
$s_{ij}^{int}, s_{ij}^{edge}$	Interest-factor and edge scores for (v_i,v_j)
$\mathcal{A}, \mathcal{H}$	Sets of training instances for the factor disentanglement
$\mathcal{L}_{edge}(\mathcal{A} \cup \mathcal{H})$	Objective to preserve asymmetric proximity between nodes
$\mathcal{L}_{disA}(\mathcal{A})$	Objective to disentangle the authority factor from others
$\mathcal{L}_{disH}(\mathcal{H})$	Objective to disentangle the hub factor from others

embedding-lookup table, to generate node embeddings. They aim to maximize the proximities between nodes visited during each random walk. Recently, many works employ advanced encoders, varying from basic graph convolution layers [7] to attention-based [33] and GAN-based [35] layers, which map the input network and node attributes to the embeddings of nodes. However, they focus on undirected networks without taking edge directions into consideration.

To address this limitation, there has been a surge of research efforts on the DNE problem. APP [41] and NERD [16] design different random walk strategies to sample node pairs to learn while taking edge directions into consideration. ATP [31] factorizes an asymmetric proximity matrix that captures the hierarchy and reachability between nodes in the network. GVAE [29] extends the graph variational autoencoder [18, 34] to directed networks. DGGAN [42] adversarially trains a discriminator and two generators to jointly learn source and target embeddings. DiGCN [32] and MagNet [39] extend the graph convolutional network (GCN) [19, 20] to directed networks by employing graph Laplacians that effectively encode the directional information. However, we observed that the embeddings obtained by the aforementioned methods suffer from considerable degradation in performance on downstream tasks when degree-related distributional shifts occur in the input network, due to a lack of consideration of such possibilities.

OOD generalization on graphs. While graph out-of-distribution generalization has been in the spotlight recently, most of them focus on *graph-level* representation learning for graph classification tasks. They aim to accurately classify graphs in the test set, based on their embeddings, even when their properties have different distributions from those in the training set. They address various types of distribution shifts, such as shifts on graph sizes [2, 23, 36], node features [2, 23], and graph structures [23]. However, most existing works are not designed for *node-level* representation learning, and/or they do not consider shifts on degree-related distribution that we aim to address in this paper.

3 ODIN: THE PROPOSED METHOD

In this section, we propose ODIN, a novel DNE method based on factor modeling in the formation of directed edges. ODIN learns disentangled source and target embeddings of each node in the training data that can be generalized to test data even under the circumstance where degree distributions of training and test data are non-identical. In Section 3.1, we first formulate the problem of DNE and present an overview of ODIN. In Sections 3.2, 3.3, and 3.4, we describe the three stages of ODIN in detail.

3.1 Overview

The DNE problem is formulated as follows: Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a given directed network, where $\mathcal{V} = \{v_1, v_2, \cdots, v_m\}$ denotes the set of m nodes and $\mathcal{E}$ denotes the set of directed edges. Let $e_{ij} \in \mathcal{E}$ be the directed edge from v_i (*i.e.*, source) to v_j (*i.e.*, target). DNE methods aim to learn source and target embedding functions which map each node $v_i \in \mathcal{V}$ to d-dimensional source and target embeddings (s_i and t_i) so that asymmetric proximities between nodes in $\mathcal{G}$ are well preserved in them.

Now, we present an overview of ODIN. Table 1 provides a list of notations used in this paper. As shown in Figure 2, ODIN consists of three stages. In Stage 1, we model the formation of a directed edge from v_i to v_j by using six node factors, which are grouped as the authority, hub, and interest factors. In Stage 2, for an edge from v_i to v_j in the input network, ODIN obtains different types of training instances by sampling negative source/target nodes while taking edge directions and node degrees into consideration. In Stage 3, from the sampled instances, ODIN learns the six factor sub-embeddings of each node v_i so that they are disentangled. Specifically, ODIN uses not only an objective for preserving proximities between v_i and v_j but also two extra objectives for disentangling the authority, hub, and interest factors.

3.2 Factor Modeling

We first model the formation of each directed edge (v_i, v_j) based on the node factors of the source v_i and the target v_j. Then, for each node, we design six factor sub-embeddings, each of which aims to preserve its characteristic for the corresponding factor. Based on these embeddings, we also compute factor scores, which represent how much the factor affects the edge formation.

The formation of each directed edge (v_i, v_j) can be modeled by six node factors grouped as follows[1]:

- **Authority Factors:** The target v_j's authority status (**a-target**) and the source v_i's bias toward authorities (**a-source**) together model a bias related to v_j's authority status (*i.e.*, in-degree).
- **Hub Factors:** The source v_i's hub status (**h-source**) and the target v_j's bias toward hubs (**h-target**) together model a bias related to v_i's hub status (*i.e.*, out-degree).
- **Interest Factors:** The source v_i's intrinsic property as a source (**i-source**) and the target v_j's intrinsic property as a target (**i-target**) together model their *pure interest* in forming an edge from v_i to v_j *after removing degree-related biases*.

Note that the six node factors can also be grouped into: (a) those as a source (*i.e.*, a-source, h-source, i-source) and those as a target (*i.e.*, a-target, h-target, i-target).

We represent a node v_i by six factor sub-embeddings (*i.e.*, a_i^{src}, h_i^{src}, i_i^{src}, a_i^{tar}, h_i^{tar}, and i_i^{tar}) corresponding to its six node factors. Then, we consider the concatenation (*i.e.*, $\oplus$) of the three factor sub-embeddings as a source as the overall source embedding s_i, and those as a target as the overall target embedding t_i, *i.e.*,

$$\mathsf{s}_i = \mathsf{a}_i^{src} \oplus \mathsf{h}_i^{src} \oplus \mathsf{i}_i^{src}, \quad \mathsf{t}_i = \mathsf{a}_i^{tar} \oplus \mathsf{h}_i^{tar} \oplus \mathsf{i}_i^{tar}. \tag{1}$$

Based on the above six factor sub-embeddings for v_i and v_j, we quantify three factor scores, which represent how much (a) the authority factor, (b) the hub factor, and (c) the interest factor, respectively, affect the formation of the directed edge (v_i, v_j). Here, each factor score is calculated by the dot product between the corresponding sub-embedding of v_i as a source and the corresponding sub-embedding of v_j as a target, *i.e.*,

$$s_{ij}^{auth} = \mathsf{a}_i^{src} \cdot \mathsf{a}_j^{tar}, \quad s_{ij}^{hub} = \mathsf{h}_i^{src} \cdot \mathsf{h}_j^{tar}, \quad s_{ij}^{int} = \mathsf{i}_i^{src} \cdot \mathsf{i}_j^{tar}. \tag{2}$$

Finally, we compute the overall edge score s_{ij}^{edge}, which represents the likelihood of the formation of a directed edge (v_i, v_j), by adding the three factor scores, *i.e.*,

$$s_{ij}^{edge} = s_{ij}^{auth} + s_{ij}^{hub} + s_{ij}^{int} = \mathsf{s}_i \cdot \mathsf{t}_j. \tag{3}$$

To sum up, when inferring edge scores, we consider the different influences modeled by the node factors.

3.3 Direction- and Degree-based Negative Sampling

We sample different types of training instances for learning the embeddings of nodes modeled in Section 3.2. Among the three groups of node factors, the authority and hub factors reflect biases related to in- and out-degrees, respectively, which are obtained from the network topology. We thus sample the instances by taking edge direction and node degree in the input network into consideration.

Regarding the authority factors, for an edge (v_i, v_j), we sample a set of n negative targets among nodes whose in-degree is smaller than the in-degree of the positive target v_j. Then, for such a negative sample $v_{j'}$, we add $(v_i, v_j, v_{j'})$ to the set $\mathcal{A}_{\dot{\gamma}}$, which is initially empty. As a result, the following inequality holds:

$$d_{in}(v_j) \ \dot{\gamma} \ d_{in}(v_{j'}), \ \forall (v_i, v_j, v_{j'}) \in \mathcal{A}_{\dot{\gamma}}, \tag{4}$$

where $d_{in}(v_j)$ denotes the in-degree of v_j. We sample another set of n negative targets $v_{j'}$ whose in-degree is greater than v_j's in-degree, and add $(v_i, v_j, v_{j'})$ to the set $\mathcal{A}_{\ddot{\gamma}}$:

$$d_{in}(v_j) \ \ddot{\gamma} \ d_{in}(v_{j'}), \ \forall (v_i, v_j, v_{j'}) \in \mathcal{A}_{\ddot{\gamma}}. \tag{5}$$

In a similar manner, regarding the hub factors, we construct $\mathcal{H}_{\dot{\gamma}}$ and $\mathcal{H}_{\ddot{\gamma}}$ satisfying the following conditions:

$$d_{out}(v_i) \ \dot{\gamma} \ d_{out}(v_{i'}), \forall (v_i, v_j, v_{i'}) \in \mathcal{H}_{\dot{\gamma}}, \tag{6}$$

$$d_{out}(v_i) \ \ddot{\gamma} \ d_{out}(v_{i'}), \forall (v_i, v_j, v_{i'}) \in \mathcal{H}_{\ddot{\gamma}}, \tag{7}$$

where $d_{out}(v_i)$ represents the out-degree of v_i. It should be noted that $\mathcal{A}_{\dot{\gamma}} \cap \mathcal{A}_{\ddot{\gamma}} = \emptyset$ and $\mathcal{H}_{\dot{\gamma}} \cap \mathcal{H}_{\ddot{\gamma}} = \emptyset$ hold.

We denote the union of $\mathcal{A}_{\dot{\gamma}}$ and $\mathcal{A}_{\ddot{\gamma}}$ by a set $\mathcal{A}$ and denote the union of $\mathcal{H}_{\dot{\gamma}}$ and $\mathcal{H}_{\ddot{\gamma}}$ by a set $\mathcal{H}$. In the next subsection, we discuss how to capture the influence of (1) biases related to v_j's in-degree based on $\mathcal{A}$, and (2) biases related to v_i's out-degree based on $\mathcal{H}$.

3.4 Disentangled Embedding Learning

We present the process of learning disentangled embeddings of nodes based on the triplets sampled in Section 3.3. We learn the source and target embeddings of nodes by using an objective for preserving asymmetric proximities between nodes in an input network. Furthermore, we learn six factor sub-embeddings of nodes by using two extra objectives for disentangling the authority, hub, and interest factors. Recall that the source and target embeddings of nodes are composed of the three factor sub-embeddings as a source and those as a target, respectively.

3.4.1 ***Preserving Asymmetric Proximities.*** For each triplet $(v_i, v_j, v_{j'}) \in \mathcal{A}$, the source and target embeddings are learned so that the *edge score* (*i.e.*, Eq. (3)) of the positive pair (v_i, v_j) is higher than that of the negative pair $(v_i, v_{j'})$, *i.e.*, $s_{ij}^{edge} \ \dot{\gamma} \ s_{ij'}^{edge}$. Similarly, for each triplet $(v_i, v_j, v_{i'}) \in \mathcal{H}$, the source and target embeddings are learned so that the edge score of the positive pair (v_i, v_j) is higher than that of the negative pair $(v_{i'}, v_j)$, *i.e.*, $s_{ij}^{edge} \ \dot{\gamma} \ s_{i'j}^{edge}$. Towards this goal, we use the *edge loss* based on Bayesian personalized ranking (BPR) [28], defined as follows:

$$\mathcal{L}_{edge}(\mathcal{A} \cup \mathcal{H}) = \sum_{(v_i, v_j, v_{j'}) \in \mathcal{A}} \mathrm{BPR}(s_{ij}^{edge}, s_{ij'}^{edge}) + \sum_{(v_i, v_j, v_{i'}) \in \mathcal{H}} \mathrm{BPR}(s_{ij}^{edge}, s_{i'j}^{edge}), \tag{8}$$

$$\mathrm{BPR}(s_{ij}^{edge}, s_{ij'}^{edge}) = -\log(\sigma(s_{ij}^{edge} - s_{ij'}^{edge})), \tag{9}$$

where σ indicates a sigmoid function.

Since $\mathcal{L}_{edge}(\mathcal{A} \cup \mathcal{H})$ is for directly updating the overall source and target embeddings of nodes, not considering each of their factor sub-embeddings separately, this objective alone does not contribute to preserving the desired factor in each factor sub-embedding. As mentioned in Section 1, using this alone may cause embeddings to lack an ODD generalization ability against degree-related distributional shifts. To address this problem, we model two additional objectives in the subsequent sections: (1) one for disentangling the authority factor from the hub and interest factors; (2) the other for disentangling the hub factor from the authority and interest factors. Through them, we can naturally distinguish the interest factor from the others, thereby disentangling all three factors.

[1] We can easily identify those factors just by counting out-/in-degrees of nodes without requiring significant times.

3.4.2 ***Disentangling the Authority Factor from the Others.*** We use the triplets in $\mathcal{A}$ to disentangle the authority factor from the remaining factors. We employ different learning strategies for the triplets in $\mathcal{A}_i$ and for those in $\mathcal{A}_{\ddot{Y}}$.

First, we describe the learning strategy for the triplets in $\mathcal{A}_i$. For a triplet $(v_i, v_j, v_{j'})$ in $\mathcal{A}_i$, $d_{in}(v_j)$ ¡ $d_{in}(v_{j'})$ holds. Thus, the authority status of v_j, which we measure by the in-degree of v_j, is higher than that of $v_{j'}$. If authority-factor scores capture the biases towards authorities, as desired, the following inequality regarding the authority factors should hold: $\forall(v_i, v_j, v_{j'}) \in \mathcal{A}_i$, s^{auth}_{ij} ¡ $s^{auth}_{ij'}$.

Regarding the other factors, *i.e.*, the hub and interest factors, we cannot draw inequalities that should hold for every triplet in $\mathcal{A}_i$. Based on these facts, in order to learn the sub-embeddings for authority factors, we use the *authority-loss function* for $\mathcal{A}_i$, which is defined as follows:

$$\mathcal{L}_{auth}(\mathcal{A}_i) = \tilde{\sum}_{(v_i,v_j,v_{j'})\in\mathcal{A}_i} \text{BPR}(s^{auth}_{ij}, s^{auth}_{ij'}). \quad (10)$$

Intuitively, this loss is for learning the sub-embeddings for authority factors of v_i, v_j, and $v_{j'}$ by maximizing the difference between s^{auth}_{ij} and $s^{auth}_{ij'}$, *i.e.*, achieving high s^{auth}_{ij} for a positive pair (v_i, v_j) and low $s^{auth}_{ij'}$ for a negative pair $(v_i, v_{j'})$.

Now, we describe the learning strategy for the triplets in $\mathcal{A}_{\ddot{Y}}$. For a triplet $(v_i, v_j, v_{j'})$ in $\mathcal{A}_{\ddot{Y}}$, $d_{in}(v_j)$ $\ddot{Y}$ $d_{in}(v_{j'})$ holds. Thus, if authority-factor scores capture the biases towards authorities, as desired, the following inequality should hold: $\forall(v_i, v_j, v_{j'}) \in \mathcal{A}_{\ddot{Y}}$, s^{auth}_{ij} $\ddot{Y}$ $s^{auth}_{ij'}$. For this inequality to hold, we use the authority-loss function for $\mathcal{A}_{\ddot{Y}}$, which is defined as follows:

$$\mathcal{L}_{auth}(\mathcal{A}_{\ddot{Y}}) = \tilde{\sum}_{(v_i,v_j,v_{j'})\in\mathcal{A}_{\ddot{Y}}} -\text{BPR}(s^{auth}_{ij}, s^{auth}_{ij'}). \quad (11)$$

Intuitively, this loss is for achieving low s^{auth}_{ij} for a positive pair (v_i, v_j) and high $s^{auth}_{ij'}$ for a negative pair $(v_i, v_{j'})$.

It should be noted that for each triplet $(v_i, v_j, v_{j'}) \in \mathcal{A}_{\ddot{Y}}$, even if s^{auth}_{ij} $\ddot{Y}$ $s^{auth}_{ij'}$ holds, the overall edge score s^{edge}_{ij} $(= s^{auth}_{ij} + s^{hub}_{ij} + s^{int}_{ij})$ between v_i and v_j, which form an edge in the input network, should be *higher* than $s^{edge}_{ij'}$ between v_i and $v_{j'}$, which do not form an edge. Thus, the following inequality regarding the remaining factors, *i.e.*, hub and interest factors, is implied: $\forall(v_i, v_j, v_{j'}) \in \mathcal{A}_{\ddot{Y}}$, $s^{hub}_{ij} + s^{int}_{ij}$ ¡ $s^{hub}_{ij'} + s^{int}_{ij'}$. Based on the above inequality, we additionally use the *hub- and interest-loss function* for $\mathcal{A}_{\ddot{Y}}$, which is defined as follows:

$$\mathcal{L}_{hub+int}(\mathcal{A}_{\ddot{Y}}) = \tilde{\sum}_{(v_i,v_j,v_{j'})\in\mathcal{A}_{\ddot{Y}}} \text{BPR}(s^{hub}_{ij} + s^{int}_{ij}, s^{hub}_{ij'} + s^{int}_{ij'}). \quad (12)$$

Intuitively, this loss is for having high $(s^{hub}_{ij} + s^{int}_{ij})$ for a positive pair (v_i, v_j) and low $(s^{hub}_{ij'} + s^{int}_{ij'})$ for a negative pair $(v_i, v_{j'})$.

Finally, we define the *disA loss* as the sum of the aforementioned losses, which together are for disentangling the authority factors from the others, *i.e.*,

$$\mathcal{L}_{disA}(\mathcal{A}) = \mathcal{L}_{auth}(\mathcal{A}_i) + \mathcal{L}_{auth}(\mathcal{A}_{\ddot{Y}}) + \mathcal{L}_{hub+int}(\mathcal{A}_{\ddot{Y}}). \quad (13)$$

3.4.3 ***Disentangling the Hub Factor from the Others.*** Next, we use the triplets in $\mathcal{H}_i$ and $\mathcal{H}_i$ to disentangle the hub factor from the remaining factors. Since *the learning strategies are symmetric to those for the triplets in $\mathcal{A}_i$ and $\mathcal{A}_{\ddot{Y}}$*, we briefly introduce them below. Details can be found in Appendix A.

For a triplet $(v_i, v_j, v_{i'})$ in $\mathcal{H}_i$, we use the *hub-loss function* for $\mathcal{H}_i$ based on the inequality s^{hub}_{ij} ¡ $s^{hub}_{i'j}$, which is defined as follows:

$$\mathcal{L}_{hub}(\mathcal{H}_i) = \tilde{\sum}_{(v_i,v_j,v_{i'})\in\mathcal{H}_i} \text{BPR}(s^{hub}_{ij}, s^{hub}_{i'j}). \quad (14)$$

For a triplet $(v_i, v_j, v_{i'})$ in $\mathcal{H}_{\ddot{Y}}$, on the other hand, we use the *hub-loss function* for $\mathcal{H}_{\ddot{Y}}$ and the *authority- and interest-loss function* for $\mathcal{H}_{\ddot{Y}}$ based on the inequalities s^{hub}_{ij} $\ddot{Y}$ $s^{hub}_{i'j}$ and $s^{auth}_{ij} + s^{int}_{ij}$ ¡ $s^{auth}_{i'j} + s^{int}_{i'j}$, respectively, which are defined as follows:

$$\begin{aligned}\mathcal{L}_{hub}(\mathcal{H}_{\ddot{Y}}) &= \tilde{\sum}_{(v_i,v_j,v_{i'})\in\mathcal{H}_{\ddot{Y}}} -\text{BPR}(s^{hub}_{ij}, s^{hub}_{i'j}),\\ \mathcal{L}_{auth+int}(\mathcal{H}_{\ddot{Y}}) &= \tilde{\sum}_{(v_i,v_j,v_{i'})\in\mathcal{H}_{\ddot{Y}}} \text{BPR}(s^{auth}_{ij} + s^{int}_{ij}, s^{auth}_{i'j} + s^{int}_{i'j}).\end{aligned} \quad (15)$$

Finally, we define the *disH* loss as the sum of the aforementioned losses, which together are for disentangling the hub factors from the others, *i.e.*,

$$\mathcal{L}_{disH}(\mathcal{H}) = \mathcal{L}_{hub}(\mathcal{H}_i) + \mathcal{L}_{hub}(\mathcal{H}_{\ddot{Y}}) + \mathcal{L}_{auth+int}(\mathcal{H}_{\ddot{Y}}). \quad (16)$$

3.4.4 ***Multi-Objective Learning.*** For multi-objective learning, we use the three aforementioned losses, *i.e.*, $\mathcal{L}_{edge}(\mathcal{A}\cup\mathcal{H})$ (Eq. (8)), $\mathcal{L}_{disA}(\mathcal{A})$ (Eq. (24)), and $\mathcal{L}_{disH}(\mathcal{H})$ (Eq. (16)), simultaneously. The final loss $\mathcal{L}$, which we use for ODIN, is defined as follows[2]:

$$\mathcal{L} = \mathcal{L}_{edge}(\mathcal{A}\cup\mathcal{H}) + \alpha(\mathcal{L}_{disA}(\mathcal{A}) + \mathcal{L}_{disH}(\mathcal{H})), \quad (17)$$

where α is to control the weight of $\mathcal{L}_{disA}(\mathcal{A})$ and $\mathcal{L}_{disH}(\mathcal{H})$.

To sum up, using Eq. (17), we learn the six sub-embeddings of each node for authority, hub, and interest factors as a source and a target. Note that we disentangle the authority factor from the hub and interest factors by using $\mathcal{L}_{disA}(\mathcal{A})$, and the hub factor from the others by using $\mathcal{L}_{disH}(\mathcal{H})$. Furthermore, **by jointly using both losses, we naturally disentangle the interest factor from the others as well.** That is, we can disentangle all three factors, thereby separately capturing each factor into corresponding two (*i.e.*, source and target) factor sub-embeddings.

Now, we obtain the source and target embeddings $\mathbf{s}_i$ and $\mathbf{t}_i$ of v_i by concatenating the three sub-embeddings as a source and a target, respectively (see Eq. (1)). Then, we regard $\mathbf{v}_i = \mathbf{s}_i \oplus \mathbf{t}_i$ as v_i's final embedding. As the sub-embeddings capture degree-related biases and pure interest separately, the final embeddings are expected to be robust to the shifts in degree distributions, as demonstrated experimentally in the following section.

4 EVALUATION

We designed our experiments, aiming at answering the following key research questions (RQs):

- RQ1: Does ODIN outperform its competitors under distributional shifts in degree distributions?
- RQ2: How robust is ODIN under various levels of distributional shifts in degree distributions?
- RO3: Is factor disentanglement effective in ODIN?
- RQ4: How sensitive is ODIN to its hyperparameters?

4.1 Experimental Settings

Datasets. We used four real-word datasets of directed networks from different domains: Gnutella (GNU), Wiki-Vote (Wiki), JUNG,

[2]We tried to add a decoupling regularizer [40] to further encourage disentanglement, but the accuracy improvement was marginal.

Table 2: Dataset statistics

Datasets	GNU	Wiki	JUNG	Ciao
Nodes	6,301	7,115	6,120	4,658
Edges	20,777	103,689	50,535	40,133
Reciprocity	0.00%	5.64%	0.90%	34.90%
Types	P2P	Election	Software	Trust

and CiaoDVD (Ciao). They are all publicly available.[3] Table 2 provides some statistics of the four datasets.

- **GNU** is a peer-to-peer network for file sharing. A node represents a host, and a directed edge from a host v_i to a host v_j represents that v_i made a connection to v_j.
- **Wiki** is a voting network for electing managers in Wikipedia. A node represents a user, and a directed edge from a user v_i to a user v_j represents that v_i voted on v_j.
- **JUNG** is a software class dependency network of JUNG 2.0.1 libraries. A node represents a Java class, and a directed edge from a class v_i to a class v_j represents that v_i is dependent on v_j.
- **Ciao** is a trust network on a DVD review site. A node represents a user, and a directed edge from a user v_i to a user v_j represents that v_i trusts v_j's reviews.

Competitors. We compare ODIN with two baselines (DeepWalk [27] and Node2Vec [5]), and seven state-of-the-art DNE methods (APP [41], GVAE [29], NERD [16], ATP [31], DiGCN [32], MagNet [39], and DGGAN [42]). We carefully tuned the hyperparameters of the competitors and ODIN. Note that the only learnable parameters of ODIN are node embeddings. For the full reproducibility of our research, we provide complete implementation details in Appendix B.

Non-ID Settings. To evaluate the out-of-distribution generalization ability of ODIN and its competitors, we design non-ID settings by splitting the edges in an input network into training and test sets *with different degree distributions*, rather than randomly splitting them: (1) Non-ID (in), where in-degree distributions are different between training and test data; (2) Non-ID (out), where out-degree distributions are different between training and test data.

For Non-ID (in), each edge (v_i, v_j) is sampled into the test set with acceptance probability $p_{ij}^{in} \propto d_{in}(v_j)^k$, which is dependent on the target v_j's in-degree $d_{in}(v_j)$. For Non-ID (out), each edge (v_i, v_j) is sampled into the test set with acceptance probability $p_{ij}^{out} \propto d_{out}(v_i)^k$, which is dependent on the source v_i's out-degree $d_{out}(v_i)$. The unsampled edges are used for training.

In the acceptance probabilities, k is a parameter for controlling the level of distributional shifts. Note that when $k = 0$, test edges are randomly sampled (among all edges), *i.e.*, it is the setting where training and test data are almost identically distributed (ID). When $k = -1$, test edges are sampled *inversely proportional* to the out-degree of the source nodes or in-degree of the target nodes. In this work, k is fixed to -1 for RQ1, RQ3, and RQ4. For RQ2, we carefully examine how robust ODIN and the state-of-the-arts competitors are in various non-ID settings by increasing the value of k.

We admit that the above setting is not the only way how degree-related distribution shifts occur in a directed network, and we leave the exploration of other settings as future work. We would like to emphasize that, to the best of our knowledge, we are the first to tackle degree-related distribution shifts in a directed network, and no standard settings have been defined regarding such shifts.

[3] http://snap.stanford.edu/ | http://konect.cc/networks/

Evaluation Tasks. To measure the effectiveness of ODIN and the competitors, we employ link prediction (LP) tasks for directed networks [16, 39, 41]. The goal of this task is to evaluate how accurately we can predict the directed edges removed from the input directed network by using each NE method.

We split the edges in the input network into training (80%) and test (20%) sets, with the goal of making them non-identically distributed as described above. In each of the training/test sets, we consider the existent edges as positive examples. Depending on how we sample the negative examples, we divide the LP task into two types: uniform LP (U-LP, in short) and biased LP (B-LP, in short) [16, 39, 41]. For U-LP, we consider the same number of non-existent edges sampled uniformly at random as negative examples. For B-LP, we consider the edges with the opposite directions to unidirectional positive examples as negative examples. That is, we evaluate how accurately each NE method can predict the directions of the edges. Finally, we classify whether each testing example is positive or negative through *logistic regression* with the embeddings obtained by each NE method as the input. We use the *area under curve* (AUC) [8], which has been widely used in many NE methods as the performance measure. We build five different training/test splits by using the aforementioned acceptance probabilities $p_{ij}^{in}, p_{ij}^{out}$ and report the performance averaged over the splits.

4.2 Results

Due to space limitation, we omit some experimental results, including (1) comparison with DICE [40], (2) effect of using a single-factor-embedding, and (3) scalability analysis for ODIN, in this paper. The details for all experiments are available in Appendix C.

RQ1: Comparison with nine competitors. We conducted comparative experiments for 2 LP tasks on 4 datasets in 2 types of non-ID settings to demonstrate the superiority of ODIN over 9 competitors. In Table 3, the values in boldface and underlined indicate the best AUC in each row and the AUC of the best 'competitor' in each row, respectively. Below, we summarize the results.

First, no single competitor consistently outperforms the other competitors. That is, best competitors change depending on tasks, datasets, and non-ID settings. Second, ODIN significantly and consistently outperforms *all* competitors for *all* LP tasks on *all* datasets in *all* non-ID settings. Specifically, in Non-ID (in), ODIN yields up to 9.43%, 8.75%, and 5.18% higher AUC than the strongest competitors ATP, MagNet, and DGGAN, respectively. Also, in Non-ID (out), ODIN yields up to 5.64%, 7.75%, and 3.67% higher AUC than the strongest competitors ATP, MagNet, and DGGAN, respectively.

The results show that our ODIN is effective compared to all the competitors in addressing the out-of-distribution generalization problem against degree-related distributional shifts on DNE. Note that most competitors represent each node as source and target embeddings. On the other hand, ODIN models authority, hub, and interest factors in the formation of directed edges, and leverages them to learn six factor sub-embeddings, which separately preserve different factors. Therefore, we attribute the superiority of ODIN to its sophisticated factor modeling and learning.

Table 3: Link prediction accuracies (in terms of AUC) of nine competitors and ODIN. ODIN significantly and consistently outperforms all competitors for both LP tasks on all datasets in both non-ID settings.

(a) Non-ID (in)

Datasets	Tasks	Undirected NE		Directed NE							ODIN
		DeepWalk	Node2Vec	APP	GVAE	NERD	ATP	DiGCN	MagNet	DGGAN	
GNU	U-LP	0.593±0.005	0.587±0.004	0.675±0.003	0.675±0.003	0.683±0.008	0.731±0.003	0.729±0.001	0.742±0.001	0.722±0.003	**0.760±0.004**
	B-LP	0.648±0.006	0.621±0.010	0.700±0.006	0.748±0.013	0.838±0.004	0.910±0.002	0.878±0.003	0.900±0.004	0.901±0.003	**0.924±0.001**
Wiki	U-LP	0.806±0.001	0.804±0.002	0.795±0.001	0.820±0.005	0.828±0.001	0.827±0.002	0.729±0.002	0.865±0.001	0.890±0.001	**0.905±0.001**
	B-LP	0.852±0.002	0.855±0.007	0.637±0.008	0.901±0.012	0.915±0.002	0.954±0.001	0.862±0.002	0.928±0.001	0.963±0.001	**0.973±0.001**
JUNG	U-LP	0.725±0.005	0.777±0.006	0.741±0.002	0.820±0.003	0.784±0.006	0.864±0.001	0.817±0.004	0.816±0.002	0.879±0.003	**0.884±0.002**
	B-LP	0.810±0.005	0.861±0.005	0.772±0.005	0.902±0.006	0.883±0.005	0.961±0.001	0.926±0.001	0.891±0.003	0.964±0.002	**0.969±0.001**
Ciao	U-LP	0.776±0.004	0.778±0.002	0.846±0.001	0.841±0.002	0.857±0.002	0.846±0.002	0.641±0.004	0.847±0.001	0.886±0.001	**0.892±0.001**
	B-LP	0.688±0.005	0.725±0.006	0.768±0.002	0.797±0.004	0.869±0.006	0.887±0.003	0.751±0.006	0.873±0.004	0.912±0.003	**0.914±0.003**

(b) Non-ID (out)

Datasets	Tasks	Undirected NE		Directed NE							ODIN
		DeepWalk	Node2Vec	APP	GVAE	NERD	ATP	DiGCN	MagNet	DGGAN	
GNU	U-LP	0.627±0.004	0.617±0.005	0.664±0.006	0.694±0.007	0.692±0.011	0.749±0.003	0.750±0.003	0.769±0.004	0.754±0.005	**0.782±0.005**
	B-LP	0.650±0.007	0.574±0.005	0.692±0.005	0.787±0.005	0.831±0.001	0.916±0.005	0.906±0.001	0.917±0.003	0.919±0.002	**0.927±0.003**
Wiki	U-LP	0.811±0.002	0.804±0.003	0.816±0.001	0.827±0.003	0.845±0.001	0.852±0.001	0.831±0.001	0.857±0.001	0.889±0.001	**0.900±0.001**
	B-LP	0.832±0.007	0.830±0.006	0.487±0.026	0.873±0.004	0.900±0.007	0.960±0.001	0.930±0.001	0.896±0.001	0.952±0.001	**0.962±0.001**
JUNG	U-LP	0.893±0.004	0.929±0.001	0.941±0.002	0.946±0.001	0.938±0.004	0.955±0.002	0.957±0.002	0.955±0.002	0.961±0.002	**0.962±0.002**
	B-LP	0.959±0.003	0.984±0.001	0.985±0.001	0.989±0.001	0.987±0.002	0.994±0.001	0.994±0.001	0.991±0.001	0.995±0.001	**0.995±0.001**
Ciao	U-LP	0.765±0.003	0.767±0.003	0.798±0.002	0.809±0.001	0.813±0.007	0.840±0.002	0.733±0.004	0.819±0.004	0.871±0.002	**0.883±0.003**
	B-LP	0.635±0.011	0.695±0.004	0.597±0.004	0.684±0.008	0.750±0.004	0.867±0.003	0.836±0.003	0.827±0.003	0.871±0.002	**0.883±0.003**

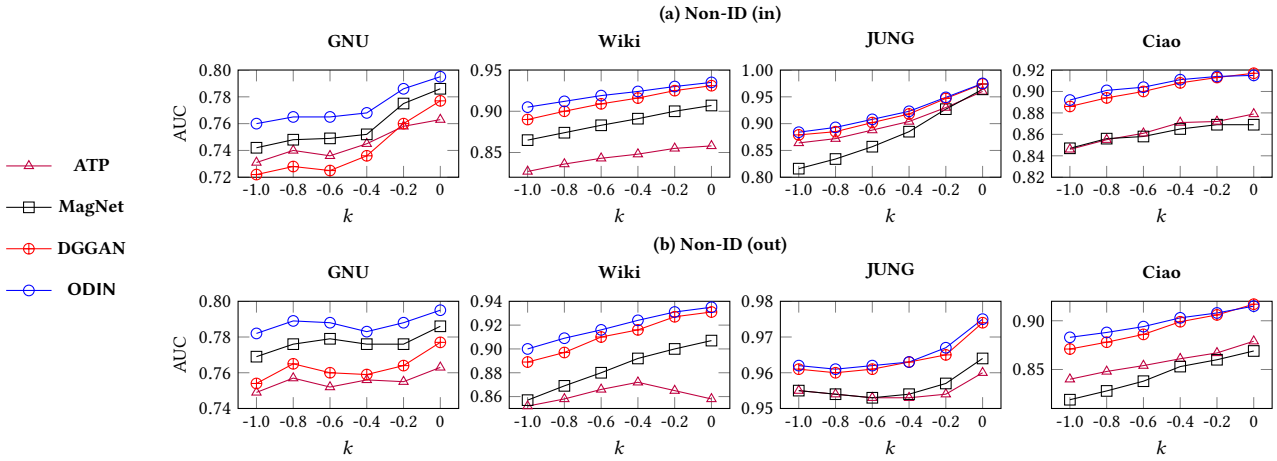

Figure 3: The effect of k on the link prediction performance. ODIN consistently achieves best AUCs in almost all cases. Notably, as k decreases, the accuracy gain of ODIN over the competitors increases.

RQ2: Effect of k on the link prediction performance. We evaluate the link prediction performances of ODIN and the strongest competitors (*i.e.*, ATP, MagNet, and DGGAN) in various non-ID settings. Specifically, we measure AUCs of each method by increasing the value of k from -1 to 0 (*spec.*, -1, -0.8, -0.6, -0.4, -0.2, and 0). Note that when $k = 0$, it becomes the ID setting, which is employed by all existing DNE methods; the smaller the value of k is, the stronger the degree of distributional shifts is.

Figure 3 shows that ODIN consistently outperforms all competitors in almost all cases, regardless of the changes of k. First, we note that, *in the ID setting (i.e., $k = 0$)*, the AUCs of ODIN are comparable to or even higher than that of the strongest competitors. Then, as k decreases (*i.e.*, as settings become more challenging), the AUCs of all methods including ODIN tend to decrease for all tasks on all datasets. However, among them, ODIN shows the smallest accuracy degradation; accordingly, the accuracy gain of ODIN against competitors steadily increases. This indicates that ODIN successfully obtains out-of-distribution generalized embeddings that are robust to distributional shifts in degree distributions.

RQ3: Effectiveness of the factor disentanglement. We model the two objectives for disentangling each of the authority and hub factors from the other two factors, in addition to the objective for

Table 4: Ablation study on the factor disentanglement strategy in ODIN. Each of the two types of disentanglement losses is effective in ODIN. Moreover, when they are jointly exploited, ODIN yields most-accurate embeddings in all datasets.

(a) Non-ID (in)

Datasets	Tasks	ODIN_A	ODIN_{disA}	ODIN_H	ODIN_{disH}	ODIN
GNU	U-LP	0.632±0.005	**0.763±0.004**	0.604±0.010	0.678±0.001	0.760±0.004
	B-LP	0.704±0.010	**0.927±0.001**	0.669±0.015	0.820±0.010	0.924±0.001
Wiki	U-LP	0.842±0.002	0.896±0.001	0.793±0.007	0.898±0.001	**0.905±0.001**
	B-LP	0.918±0.001	0.965±0.001	0.863±0.011	0.968±0.001	**0.973±0.001**
JUNG	U-LP	0.825±0.004	0.878±0.003	0.714±0.006	**0.884±0.002**	**0.884±0.002**
	B-LP	0.929±0.003	0.966±0.002	0.830±0.004	**0.970±0.001**	0.969±0.001
Ciao	U-LP	0.820±0.003	0.890±0.001	0.853±0.001	0.886±0.001	**0.892±0.001**
	B-LP	0.788±0.009	0.912±0.002	0.867±0.005	0.909±0.002	**0.914±0.003**

(b) Non-ID (out)

Datasets	Tasks	ODIN_A	ODIN_{disA}	ODIN_H	ODIN_{disH}	ODIN
GNU	U-LP	0.648±0.004	**0.786±0.006**	0.668±0.005	0.692±0.007	0.782±0.005
	B-LP	0.718±0.005	**0.934±0.003**	0.770±0.010	0.835±0.008	0.927±0.003
Wiki	U-LP	0.853±0.002	0.893±0.001	0.833±0.003	0.895±0.001	**0.900±0.001**
	B-LP	0.918±0.001	0.956±0.001	0.894±0.004	0.959±0.001	**0.962±0.001**
JUNG	U-LP	0.957±0.003	0.961±0.002	0.890±0.007	**0.963±0.002**	0.962±0.002
	B-LP	0.993±0.001	**0.995±0.001**	0.969±0.003	**0.995±0.001**	**0.995±0.001**
Ciao	U-LP	0.814±0.003	0.877±0.003	0.841±0.002	0.873±0.002	**0.883±0.003**
	B-LP	0.764±0.007	0.875±0.002	0.816±0.006	0.867±0.003	**0.883±0.003**

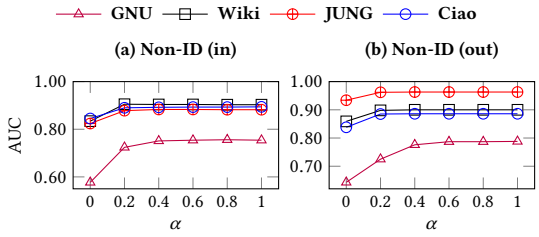

Figure 4: The effect of α on the link prediction performance. The disentanglement losses are effective, and their effectiveness is not highly sensitive as long as α is not too small.

preserving asymmetric proximities between nodes. By pursuing such multiple objectives, we can disentangle the authority, hub, and interest factors. To verify the effectiveness of this disentanglement, we conduct experiments to answer the following two sub-questions:

- RQ3-1: Is each of two disentanglement losses effective in ODIN?
- RQ3-2: Is jointly using the both losses effective in ODIN?

For RQ3-1, to evaluate the effectiveness of the objective design for disentangling the authority factor from the others, we first compare the AUCs of two variants of ODIN: (1) ODIN_A, which uses only the *edge loss* based on $\mathcal{A}$; (2) ODIN_{disA}, which uses the *disA loss* (*i.e.*, Eq. (24)) in addition to the *edge loss* based on $\mathcal{A}$. To evaluate the effectiveness of the objective design for disentangling the hub factor from the others, we also compare these two variants of ODIN: (1) ODIN_H, which uses only the *edge loss* based on $\mathcal{H}$; (2) ODIN_{disH}, which uses the *disH loss* (*i.e.*, Eq. (16)) in addition to the *edge loss* based on $\mathcal{H}$. Tables 4-(a) and -(b) show that ODIN_{disA} consistently and significantly outperforms ODIN_A for both LP tasks on all datasets in both non-ID settings. *i.e.*, Non-ID (in) and Non-ID (out). ODIN_{disH} also outperforms ODIN_H in all cases. The results indicate that the disentanglement losses are effective in obtaining embeddings robust to distributional shifts in degree distributions.

For RQ3-2, to verify whether using both losess, thereby disentangling all three factors, is effective in ODIN, we compare the AUCs of ODIN_{disA}, ODIN_{disH}, and ODIN. Table 4 shows that ODIN performs best or very close to the best one in all cases. Although ODIN_{disA} or ODIN_{disB} is slightly more accurate than ODIN at times, their accuracies are susceptible to datasets, which indicates (hidden) beneficial factors could be different according to datasets. However, ODIN yields robust embeddings to any dataset, being able to selectively adopt beneficial factors for given datasets.

RQ4: Hyperparamter analysis for ODIN To understand the learning stability of ODIN, we analyze how the parameter α (*i.e.*, the loss weight for the losses for disentangling the authority and hub factors from the other factors) affects the link prediction performance of ODIN. Figures 4-(a) and -(b) show how the performance of ODIN changes in the four datasets depending on α in Non-ID (in) and Non-ID (out), respectively. We observe that the AUCs of ODIN steadily increase until α reaches 0.4 and then the AUCs converge. The results indicate that ODIN is not highly sensitive to α, and it shows excellent out-of-distribution generalization ability as long as α is not too small.

5 CONCLUSIONS

In this work, we pointed out that the existing DNE methods face difficulties in effectively addressing the OOD generalization problem. We empirically demonstrated that their accuracies significantly degrade under degree-related distributional shifts. To address this limitation, we proposed a novel DNE method, named ODIN, which models multiple factors in the formation of directed edges and learns nodes' multiple factor sub-embeddings. In ODIN, the node embeddings are learned to jointly pursue the following three objectives: (O1) to preserve asymmetric proximities between nodes; (O2) to disentangle the authority factors from the others; (O3) to disentangle the hub factors from the others. The embeddings, which capture degree-related biases and pure interest, are shown to be robust to the shifts in degree distributions. We demonstrated that ODIN universally outperforms 9 competitors on 4 real-world datasets under various degrees of distributional shifts. Through extensive ablation studies, we showed clearly the effectiveness of our strategies for factor modeling and disentangled embedding learning.

To the best of our knowledge, we are the first to study the OOD problem of degree-distributional-shift. As the first step, we started exploring the most well-known degree-biases (*i.e.*, hub and authority) in graph-mining-research. Our discovery could encourage follow-up studies to leverage other factors (*e.g.*, sensitive attributes of nodes) that influence edge formation.

ACKNOWLEDGMENT

This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.RS-2022-00155586 and No.2022-0-00352) and by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education (NRF-2022R1A6A3A03069212).

REFERENCES

[1] Albert-László Barabási. 2003. Linked: The new science of networks.

[2] Beatrice Bevilacqua, Yangze Zhou, and Bruno Ribeiro. 2021. Size-Invariant Graph Representations for Graph Classification Extrapolations. In *Proc. of the International Conference on Machine Learning (ICML)*, Vol. 139. 837–851.

[3] Chanyoung Chung and Joyce Jiyoung Whang. 2021. Knowledge Graph Embedding via Metagraph Learning. In *Proc. of the ACM International Conference on Research & Development in Information Retrieval (ACM SIGIR)*. 2212–2216.

[4] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. 2019. A Survey on Network Embedding. *IEEE Trans. Knowl. Data Eng.* 31, 5 (2019), 833–852.

[5] Aditya Grover and Jure Leskovec. 2016. Node2vec: Scalable Feature Learning for Networks. In *Proc. of the ACM International Conference on Knowledge Discovery and Data Mining (ACM KDD)*. 855–864.

[6] Evert Gummesson. 2007. Case study research and network theory: birds of a feather. *Qualitative Research in Organizations and Management: An International Journal* (2007).

[7] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Proc. of the Annual Conference. on Neural Information Processing Systems (NeurIPS)*. 1024–1034.

[8] James A Hanley and Barbara J McNeil. 1982. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology* 143, 1 (1982), 29–36.

[9] Yue He, Zheyan Shen, and Peng Cui. 2021. Towards Non-I.I.D. image classification: A dataset and baselines. *Pattern Recognit.* 110 (2021), 107383.

[10] Hyunjin Hwang, Seungwoo Lee, Chanyoung Park, and Kijung Shin. 2022. AHP: Learning to Negative Sample for Hyperedge Prediction. In *Proc. of the ACM International Conference on Research & Development in Information Retrieval (ACM SIGIR)*. 2237–2242.

[11] Myung-Hwan Jang, Yun-Yong Ko, Dongkyu Jeong, Jeong-Min Park, and Sang-Wook Kim. 2022. RealGraph$^{\text{GPU}}$: A High-Performance GPU-Based Graph Engine toward Large-Scale Real-World Network Analysis. In *Proc. of the ACM International Conference on Information and Knowledge Management (ACM CIKM)*. 4074–4078.

[12] Hawoong Jeong, Zoltan Néda, and Albert-László Barabási. 2003. Measuring preferential attachment in evolving networks. *EPL (Europhysics Letters)* 61, 4 (2003), 567.

[13] Yong-Yeon Jo, Myung-Hwan Jang, Sang-Wook Kim, and Sunju Park. 2019. RealGraph: A Graph Engine Leveraging the Power-Law Distribution of Real-World Graphs. In *Proc. of The Web Conference (WWW)*. 807–817.

[14] Jian Kang, Yan Zhu, Yinglong Xia, Jiebo Luo, and Hanghang Tong. 2022. Rawlsgcn: Towards rawlsian difference principle on graph convolutional network. In *Proc. of The Web Conference (WWW)*. 1214–1225.

[15] Yoonsuk Kang, Woncheol Lee, Yeon-Chang Lee, Kyungsik Han, and Sang-Wook Kim. 2021. Adversarial Learning of Balanced Triangles for Accurate Community Detection on Signed Networks. In *Proc. of the IEEE International Conference on Data Mining (ICDM)*. 1150–1155.

[16] Megha Khosla, Jurek Leonhardt, Wolfgang Nejdl, and Avishek Anand. 2019. Node Representation Learning for Directed Graphs. In *Proc. of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, Vol. 11906. 395–411.

[17] Taeri Kim, Yeon-Chang Lee, Kijung Shin, and Sang-Wook Kim. 2022. MARIO: Modality-Aware Attention and Modality-Preserving Decoders for Multimedia Recommendation. In *Proc. of the ACM International Conference on Information and Knowledge Management (ACM CIKM)*. 993–1002.

[18] Thomas N. Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. In *Proc. NeurIPS Workshop Bayesian Deep Learning*. 1–3.

[19] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proc. of the International Conference on Learning Representations (ICLR)*.

[20] Taeyong Kong, Taeri Kim, Jinsung Jeon, Jeongwhan Choi, Yeon-Chang Lee, Noseong Park, and Sang-Wook Kim. 2022. Linear, or Non-Linear, That is the Question!. In *Proc. of the ACM International Conference on Web Search and Data Mining (ACM WSDM)*. 517–525.

[21] Yeon-Chang Lee, JaeHyun Lee, Dongwon Lee, and Sang-Wook Kim. 2022. THOR: Self-Supervised Temporal Knowledge Graph Embedding via Three-Tower Graph Convolutional Networks. In *Proc. of the IEEE International Conference on Data Mining (ICDM)*. 1035–1040.

[22] Yeon-Chang Lee, Nayoun Seo, Kyungsik Han, and Sang-Wook Kim. 2020. ASiNE: Adversarial Signed Network Embedding. In *Proc. of the ACM International Conference on Research & Development in Information Retrieval (ACM SIGIR)*. 609–618.

[23] Haoyang Li, Xin Wang, Ziwei Zhang, and Wenwu Zhu. 2021. OOD-GNN: Out-of-Distribution Generalized Graph Neural Network. *CoRR* abs/2112.03806 (2021).

[24] Zelong Li, Jianchao Ji, Zuohui Fu, Yingqiang Ge, Shuyuan Xu, Chong Chen, and Yongfeng Zhang. 2021. Efficient Non-Sampling Knowledge Graph Embedding. In *Proc. of The Web Conference (WWW)*. 1727–1736.

[25] David Liben-Nowell and Jon M. Kleinberg. 2003. The link prediction problem for social networks. In *Proc. of the ACM International Conference on Information and Knowledge Management (ACM CIKM)*. 556–559.

[26] Fan Liu, Zhiyong Cheng, Lei Zhu, Zan Gao, and Liqiang Nie. 2021. Interest-aware Message-Passing GCN for Recommendation. In *Proc. of The Web Conference (WWW)*. 1296–1305.

[27] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In *Proc. of the ACM International Conference on Knowledge Discovery and Data Mining (ACM KDD)*. 701–710.

[28] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* (2012).

[29] Guillaume Salha, Stratis Limnios, Romain Hennequin, Viet-Anh Tran, and Michalis Vazirgiannis. 2019. Gravity-Inspired Graph Autoencoders for Directed Link Prediction. In *Proc. of the ACM International Conference on Information and Knowledge Management (ACM CIKM)*. 589–598.

[30] Zheyan Shen, Jiashuo Liu, Yue He, Xingxuan Zhang, Renzhe Xu, Han Yu, and Peng Cui. 2021. Towards Out-Of-Distribution Generalization: A Survey. *CoRR* (2021). https://arxiv.org/abs/2108.13624

[31] Jiankai Sun, Bortik Bandyopadhyay, Armin Bashizade, Jiongqian Liang, P. Sadayappan, and Srinivasan Parthasarathy. 2019. ATP: Directed Graph Embedding with Asymmetric Transitivity Preservation. In *Proc. of the AAAI International Conference on Artificial Intelligence (AAAI)*. 265–272.

[32] Zekun Tong, Yuxuan Liang, Changsheng Sun, Xinke Li, David S. Rosenblum, and Andrew Lim. 2020. Digraph Inception Convolutional Networks. In *Proc. of the Annual Conference on Neural Information Processing Systems (NeurIPS)*.

[33] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *Proc. of the International Conference on Learning Representations (ICLR)*.

[34] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural Deep Network Embedding. In *Proc. of the ACM International Conference on Knowledge Discovery and Data Mining (ACM KDD)*. 1225–1234.

[35] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. GraphGAN: Graph Representation Learning With Generative Adversarial Nets. In *Proc. of the AAAI Conference on Artificial Intelligence (AAAI)*. 2508–2515.

[36] Gilad Yehudai, Ethan Fetaya, Eli A. Meirom, Gal Chechik, and Haggai Maron. 2021. From Local Structures to Size Generalization in Graph Neural Networks. In *Proc. of the International Conference on Machine Learning (ICML)*, Vol. 139. 11975–11986.

[37] Hyunsik Yoo, Yeon-Chang Lee, Kijung Shin, and Sang-Wook Kim. 2022. Directed Network Embedding with Virtual Negative Edges. In *Proc. of the ACM International Conference on Web Search and Data Mining (ACM WSDM)*. 1291–1299.

[38] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. 2020. Network Representation Learning: A Survey. *IEEE Trans. Big Data* 6, 1 (2020), 3–28.

[39] Xitong Zhang, Nathan Brugnone, Michael Perlmutter, and Matthew J. Hirn. 2021. MagNet: A Magnetic Neural Network for Directed Graphs. In *Proc. of the Annual Conference on Neural Information Processing Systems (NeurIPS)*.

[40] Yu Zheng, Chen Gao, Xiang Li, Xiangnan He, Yong Li, and Depeng Jin. 2021. Disentangling User Interest and Conformity for Recommendation with Causal Embedding. In *Proc. of The Web Conference (WWW)*. 2980–2991.

[41] Chang Zhou, Yuqiong Liu, Xiaofei Liu, Zhongyi Liu, and Jun Gao. 2017. Scalable Graph Embedding for Asymmetric Proximity. In *Proc. of the AAAI International Conference on Artificial Intelligence (AAAI)*. 2942–2948.

[42] Shijie Zhu, Jianxin Li, Hao Peng, Senzhang Wang, and Lifang He. 2021. Adversarial Directed Graph Embedding. In *Proc. of the AAAI International Conference on Artificial Intelligence (AAAI)*. 4741–4748.

APPENDIX

A LOSS FOR DISENTANGLING THE HUB FACTOR FROM THE AUTHORITY AND INTEREST FACTORS

We use the triplets in $\mathcal{H}$ to disentangle the hub factor from the remaining factors. We employ different learning strategies for the triplets in $\mathcal{H}_{i}$ and for those in $\mathcal{H}_{\ddot{Y}}$.

First, we describe the learning strategy for the triplets in $\mathcal{H}_{i}$. For a triplet $(v_i, v_j, v_{i'})$ in $\mathcal{H}_{i}$, $d_{out}(v_i)$ ¡ $d_{out}(v_{i'})$ holds. Thus, the hub status of v_i, which we measure by the out-degree of v_i, is higher than that of $v_{i'}$. If hub-factor scores capture the biases towards hubs, as desired, the following inequality regarding the hub factors should hold: $\forall(v_i, v_j, v_{i'}) \in \mathcal{H}_{i}$,

$$s^{hub}_{ij} \text{ ¡ } s^{hub}_{i'j}. \tag{18}$$

Regarding the other factors, *i.e.*, the authority and interest factors, we cannot draw inequalities that always hold for every triplet in $\mathcal{H}_{i}$. Based on these observations, in order to learn the sub-embeddings for hub factors, we use the *hub-loss function* for $\mathcal{H}_{i}$, which is defined as follows:

$$\mathcal{L}_{hub}(\mathcal{H}_{i}) = \tilde{O}_{(v_i,v_j,v_{i'})\in\mathcal{H}_{i}} \text{BPR}(s^{hub}_{ij}, s^{hub}_{i'j}). \tag{19}$$

Intuitively, this loss is for learning the sub-embeddings for hub factors of v_i, v_j, and $v_{i'}$ by maximizing the difference between s^{hub}_{ij} and $s^{hub}_{i'j}$, *i.e.*, achieving high s^{hub}_{ij} for a positive pair (v_i, v_j) and low $s^{hub}_{i'j}$ for a negative pair $(v_{i'}, v_j)$.

Now, we describe the learning strategy for the triplets in $\mathcal{H}_{\ddot{Y}}$. For a triplet $(v_i, v_j, v_{i'})$ in $\mathcal{H}_{\ddot{Y}}$, $d_{out}(v_i)$ $\ddot{Y}$ $d_{out}(v_{i'})$ holds. Thus, if hub-factor scores capture the biases towards hubs, as desired, the following inequality should hold: $\forall(v_i, v_j, v_{i'}) \in \mathcal{H}_{\ddot{Y}}$,

$$s^{hub}_{ij} \text{ } \ddot{Y} \text{ } s^{hub}_{i'j}. \tag{20}$$

For this inequality to hold, we use the hub-loss function for $\mathcal{H}_{\ddot{Y}}$, which is defined as follows:

$$\mathcal{L}_{hub}(\mathcal{H}_{\ddot{Y}}) = \tilde{O}_{(v_i,v_j,v_{i'})\in\mathcal{H}_{\ddot{Y}}} -\text{BPR}(s^{hub}_{ij}, s^{hub}_{i'j}). \tag{21}$$

Intuitively, this loss is for achieving low s^{hub}_{ij} for a positive pair (v_i, v_j) and high $s^{hub}_{i'j}$ for a negative pair $(v_{(}i', v_j)$.

It should be noted that Recall that, for each triplet $(v_i, v_j, v_{i'}) \in \mathcal{H}_{\ddot{Y}}$, even if s^{hub}_{ij} $\ddot{Y}$ $s^{hub}_{i'j}$ holds, the overall edge score s^{edge}_{ij} (= $s^{auth}_{ij} + s^{hub}_{ij} + s^{int}_{ij}$) between v_i and v_j, which form an edge in the input network, should be higher than $s^{edge}_{i'j}$ between v_i and $v_{j'}$, which do not form an edge.. Thus, the following inequality regarding the remaining factors, *i.e.*, authority and interest factors, is implied: $\forall(v_i, v_j, v_{i'}) \in \mathcal{H}_{\ddot{Y}}$,

$$s^{auth}_{ij} + s^{int}_{ij} \text{ ¡ } s^{auth}_{i'j} + s^{int}_{i'j}. \tag{22}$$

Based on the above inequality, we additionally use the *authority- and interest-loss function* for $\mathcal{H}_{\ddot{Y}}$, which is defined as follows:

$$\mathcal{L}_{auth+int}(\mathcal{H}_{\ddot{Y}}) = \tilde{O}_{(v_i,v_j,v_{i'})\in\mathcal{H}_{\ddot{Y}}} \text{BPR}(s^{auth}_{ij} + s^{int}_{ij}, s^{auth}_{i'j} + s^{int}_{i'j}). \tag{23}$$

Intuitively, this loss is for having high $(s^{auth}_{ij} + s^{int}_{ij})$ for a positive pair (v_i, v_j) and low $(s^{auth}_{i'j} + s^{int}_{i'j})$ for a negative pair $(v_{i'}, v_j)$.

Finally, we define the *disH loss* as the sum of the aforementioned losses, which together are for disentangling the hub factors from the others, *i.e.*,

$$\mathcal{L}_{disH}(\mathcal{H}) = \mathcal{L}_{hub}(\mathcal{H}_{i}) + \mathcal{L}_{hub}(\mathcal{H}_{\ddot{Y}}) + \mathcal{L}_{auth+int}(\mathcal{H}_{\ddot{Y}}). \tag{24}$$

B IMPLEMENTATION DETAILS

For a fair comparison, we set the dimensionality of embeddings to 120 in all NE methods including ODIN. To this end, for most competitors, such as APP, NERD, ATP, and DGGAN, we set the dimensionality of source and target embeddings to 60, respectively. Other directed NE methods, such as GVAE, DiGCN, and MagNet, output a single embedding of dimensionality 120 per node, without dividing into source/target embeddings. For ODIN, we set the dimensionality of each of six factor sub-embeddings to 20.

For hyperparameters of the competitors, we used the best settings carefully found via grid search in the following ranges, which are suggested in their respective papers:

- Number of walks ∈ {10, 20, 40, 80} (for DeepWalk, Node2Vec)
- Walk length ∈ {60, 80, 100} (for DeepWalk, Node2Vec)
- $\gamma \in$ {5, 10, 15, 20} (for NERD)
- $\lambda \in$ {0.005, 0.05, 1, 5, 10} (for GVAE)
- $\alpha \in$ {0.05, 0.1, 0.15, 0.2} (for DiGCN)
- $q \in$ {0.05, 0.1, 0.15, 0.2, 0.25} (for MagNet)

For ODIN, we set $\alpha = 0.5$ and $n = 2$ (*i.e.*, the total number of negative samples per edge is 8), which consistently shows the best accuracy in all tasks, datasets, and degrees of distributional shifts.

C FURTHER RESULTS

We show some additional experimental results such as (1) comparison with DICE [40], (2) effect of using a single-factor-embedding, and (3) scalability analysis for ODIN.

The results in Table I show that ODIN outperforms DICE on ALL datasets (*e.g.*, on Ciao in B-LP in Non-ID (in), 0.914 (ODIN) vs. 0.789 (DICE)). DICE's intrinsic difficulty in addressing our problem lies in its inability to consider asymmetric relationships between nodes: (1) it cannot distinguish source and target roles in edge formation (not using source and target embeddings); (2) it models edge formation by using two-factors only (not six-factors).

From Table I, we can also see that using all factor-embeddings (*i.e.*, ODIN) leads to higher or comparable accuracy to using only single factor-embedding (*i.e.*, authority, hub, or interest factor). For example, on Ciao in U-LP in Non-ID (in), the AUC of ODIN (0.892) is greater than AUC when using only authority (0.859), hub (0.868), or interest factor-embedding (0.819), respectively.

Lastly, we confirmed through additional experiments with Wiki-Vote that ODIN has linear scalability in the number of edges (100K, 200K, 300K, and 400K edges). Note that ODIN performs only constant-times slower than ODIN without disentanglement, which is our key component.

Table I: Comparison of DICE, ODIN$_{auth}$ using only authority-factor-embeddings, ODIN$_{hub}$ using only hub-factor-embeddings, ODIN$_{int}$ using only interest-factor-embeddings, and ODIN using all factor-embeddings when $k = -1$ (*i.e.*, there exist degree-related distributional shifts). The results show that ODIN outperforms DICE in all cases and achieves better or comparable performance to using only single factor-embeddings.

(a) Non-ID (in)

Datasets	Tasks	DICE	ODIN$_{auth}$	ODIN$_{hub}$	ODIN$_{int}$	ODIN
GNU	**U-LP**	0.742	0.777	0.596	0.557	0.760
	B-LP	0.904	0.927	0.669	0.571	0.924
Wiki	**U-LP**	0.825	0.880	0.905	0.849	0.905
	B-LP	0.843	0.958	0.973	0.919	0.973
JUNG	**U-LP**	0.724	0.874	0.869	0.700	0.884
	B-LP	0.830	0.953	0.963	0.800	0.969
Ciao	**U-LP**	0.847	0.859	0.868	0.819	0.892
	B-LP	0.789	0.879	0.891	0.794	0.914

(b) Non-ID (out)

Datasets	Tasks	DICE	ODIN$_{auth}$	ODIN$_{hub}$	ODIN$_{int}$	ODIN
GNU	**U-LP**	0.770	0.808	0.627	0.629	0.782
	B-LP	0.912	0.940	0.701	0.685	0.927
Wiki	**U-LP**	0.862	0.884	0.878	0.825	0.900
	B-LP	0.901	0.953	0.949	0.849	0.962
JUNG	**U-LP**	0.932	0.962	0.953	0.866	0.962
	B-LP	0.987	0.995	0.993	0.951	0.995
Ciao	**U-LP**	0.836	0.876	0.838	0.811	0.883
	B-LP	0.811	0.879	0.817	0.768	0.883