Topology Distillation for Recommender System

SeongKu Kang, Junyoung Hwang, Wonbin Kweon, Hwanjo Yu*
Pohang University of Science and Technology (POSTECH), South Korea
{seongku, jyhwang, kwb4453, hwanjoyu}@postech.ac.kr

ABSTRACT

Recommender Systems (RS) have employed knowledge distillation which is a model compression technique training a compact student model with the knowledge transferred from a pre-trained large teacher model. Recent work has shown that transferring knowledge from the teacher’s intermediate layer significantly improves the recommendation quality of the student. However, they transfer the knowledge of individual representation point-wise and thus have a limitation in that primary information of RS lies in the relations in the representation space. This paper proposes a new topology distillation approach that guides the student by transferring the topological structure built upon the relations in the teacher space. We first observe that simply making the student learn the whole topological structure is not always effective and even degrades the student’s performance. We demonstrate that because the capacity of the student is highly limited compared to that of the teacher, learning the whole topological structure is daunting for the student. To address this issue, we propose a novel method named Hierarchical Topology Distillation (HTD) which distills the topology hierarchically to cope with the large capacity gap. Our extensive experiments on real-world datasets show that the proposed method significantly outperforms the state-of-the-art competitors. We also provide in-depth analyses to ascertain the benefit of distilling the topology for RS.

CCS CONCEPTS

• **Information systems** → **Learning to rank**; *Collaborative filtering*; Retrieval efficiency.

KEYWORDS

Recommender System; Knowledge Distillation; Relational Knowledge; Model Compression; Retrieval efficiency

ACM Reference Format:
SeongKu Kang, Junyoung Hwang, Wonbin Kweon, Hwanjo Yu. 2021. Topology Distillation for Recommender System. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD ’21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447548.3467319

1 INTRODUCTION

The size of recommender systems (RS) has kept increasing, as they have employed deep and sophisticated model architectures to better understand the complex nature of user-item interactions [8, 12, 22, 26]. A large model with many learning parameters has a high capacity and therefore generally achieves higher recommendation accuracy. However, it also requires high computational costs, which results in high inference latency. For this reason, it is challenging to adopt such a large model to the real-time platform [8, 12, 22, 26].

To tackle this problem, *Knowledge Distillation* (KD) has been adopted to RS [8, 10, 12, 22, 26, 31]. KD is a model-independent strategy to improve the performance of a compact model (i.e., student) by transferring the knowledge from a pre-trained large model (i.e., teacher). The distillation is conducted in two steps. The teacher is first trained with the training set, and the student is trained with help from the teacher along with the training set. The student model, which is a compact model, is used in the inference time. During the distillation, the teacher can provide additional supervision that is not explicitly revealed from the training set. As a result, the student trained with KD shows better prediction performance than the student trained only with the training set. Also, it has low inference latency due to its small size.

Most existing KD methods for RS transfer the knowledge from the teacher’s predictions [8, 10, 12, 22, 26] (Figure 1a). They basically enforce the student to imitate the teacher’s recommendation results, providing guidance to the predictions of the student. There is another recent approach that transfers the *latent* knowledge from the teacher’s intermediate layer [8, 31], pointing out that the predictions incompletely reveal the teacher’s knowledge and the intermediate representation can additionally provide a detailed explanation on the final prediction of the teacher. They adopt *hint regression* [20] that makes the student’s representation approximate the teacher’s representation via a few regression layers. This enables the student to get compressed information on each entity (e.g., user and item) (Figure 1b) that can restore more detailed preference information in the teacher [8].

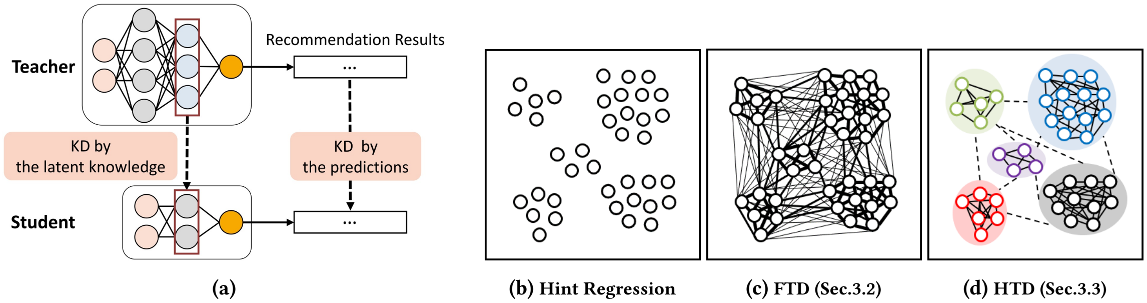
However, the existing hint regression-based methods focus on distilling the individual representation of each entity, disregarding the *relations* of the representations. In RS, each entity is better understood by its relations to the other entities rather than by its individual representation. For instance, a user’s preference is represented in relation to (and in contrast with) other users and items. Also, the student can take advantage of the space, where the relations found by the teacher are well preserved, in finding more accurate ranking orders among the entities and thereby improving the recommendation performance.

This paper proposes a new distillation approach that effectively transfers the relational knowledge existing in the teacher’s representation space. A natural question is how to define the relational

[*] Corresponding Author

KDD ’21, August 14–18, 2021, Virtual Event, Singapore

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467319



(a)

(b) Hint Regression

(c) FTD (Sec.3.2)

(d) HTD (Sec.3.3)

Figure 1: (a) The overview of KD in RS. (b-d) The conceptual illustrations of the latent knowledge that each method transfers from the teacher's representation space. Each point corresponds to a representation of each entity. (b) transfers the information of each entity to the student point-wise. However, our approach (c-d) transfers the relations among the entities. FTD/HTD refers to Full/Hierarchical topology distillation, and the solid/dotted line denotes the entity/group-level topology, respectively.

knowledge and distill it to the student. We build a *topological structure* that represents the relations in the teacher space based on the similarity information, then utilize it to guide the learning of the student via distillation. Specifically, we train the student with the distillation loss that preserves the teacher's topological structure in its representation space along with the original loss function. Trained with the topology distillation, the student can better preserve the relations in the teacher space, which not only improves the recommendation performance but also better captures the semantic of entities (reported in Section 4.3).

However, we observe that simply making the student learn all the topology information (Figure 1c) is not always effective and sometimes even degrades the student's recommendation performance (reported in Section 4.2). This phenomenon is explained by the huge capacity gap between the student and the teacher; the capacity of the student is highly limited compared to that of the teacher, and learning all the topological structure in the teacher space is often daunting for the student. To address this issue, we propose a method named Hierarchical Topology Distillation (HTD) which effectively transfers the vast teacher's knowledge to the student with limited capacity. HTD represents the topology hierarchically and transfers the knowledge in multi-levels using the hierarchy (Figure 1d).

Specifically, HTD adaptively finds *preference groups* of entities such that the entities within each group share similar preferences. Then, the topology is hierarchically structured in group-level and entity-level. The *group-level topology* represents the summarized relations across the groups, providing an overview of the whole topology. The *entity-level topology* represents the relations of entities belonging to the same group. This provides a fine-grained view on important relations among the entities having similar preferences, which directly affects the top-N recommendation performance. By compressing the complex individual relations across the groups, HTD relaxes the daunting supervision and enables the student to better focus on the important relations. In summary, the key contributions of our work are as follows:

- We address the necessity of transferring the relational knowledge from the teacher representation space and develop a general topology distillation approach for RS.
- We develop a new topology distillation method, named FTD, designed to guide the student by transferring the full topological structure built upon the relations in the teacher space.
- We propose a novel topology distillation method, named HTD, designed to effectively transfer the vast relational knowledge to the student considering the huge capacity gap.
- We validate the superiority of the proposed approach by extensive experiments. We also provide in-depth analyses to verify the benefit of distilling the topological structure.

2 RELATED WORK

Knowledge Distillation. Knowledge distillation (KD) is a model-independent strategy that accelerates the training of a student model with the knowledge transferred from a pre-trained teacher model. Most KD methods have mainly focused on the image classification task. An early work [5] matches the class distributions (i.e., the softmax output) of the teacher and the student. The class distribution has richer information (e.g., inter-class correlation) than the one-hot class label, which improves learning of the student model. Pointing out that utilizing the predictions alone is insufficient because meaningful intermediate information is ignored, subsequent methods [18, 20, 24, 28, 29] have distilled knowledge from the teacher's intermediate layer along with the predictions. [20] proposes "hint regression" that matches the intermediate representations. Subsequently, [28] matches the gram matrices of the representations, [29] matches the attention maps from the networks, and [15, 24] match the similarities on activation maps of the convolutional layer.

Reducing inference latency of RS. As the size of RS is continuously increasing, various approaches have been proposed for reducing the model size and inference latency. Several methods [16, 17, 30] have utilized the binary representations of users and items. With the discretized representations, the search costs can be considerably reduced via the hash technique. However, due to their restricted capability, the loss of recommendation accuracy is inevitable [8, 12, 26]. Also, various computational acceleration techniques have been successfully adopted to reduce the search costs. In specific, order-preserving transformations [1], pruning and compression techniques [13, 23, 25], tree-based data structures [1, 2], and approximated nearest-neighbor search [21] have been employed to reduce the inference latency. However, they have limitations in that the techniques are only applicable to specific models or easy to fall into a local optimum because of the local search [8, 12].

Knowledge Distillation for RS. KD, which is the model-agnostic strategy, has been widely adopted in RS. Similar to the progress on computer vision, the existing methods are categorized into two groups (Figure 1a): (1) the methods distilling knowledge from the predictions, (2) the methods distilling the latent knowledge from the intermediate layer. Note that the two groups of methods can be utilized together to fully improve the student [8].

(1) KD by the predictions. Motivated by [5] that matches the class distributions, most existing methods [8, 10, 12, 22, 26] have focused on matching the predictions (i.e., recommendation results) from the teacher and the student. The teacher's predictions convey additional information about the subtle difference among the items, helping the student generalize better than directly learning from binary labels [12]. This research direction focuses on designing a method effectively utilizing the teacher's predictions. First, [12, 22] distill the knowledge of the items with high scores in the teacher's predictions. Since a user is interested in only a few items, distilling knowledge of a few top-ranked items is effective to discover the user's preferable items [22]. Most recently, [10] utilizes rank-discrepancy information between the predictions from the teacher and the student. Specifically, [10] focuses on distilling the knowledge of the items ranked highly by the teacher but ranked lowly by the student. On the one hand, [8, 26] focus on distilling ranking order information from the teacher's predictions. Concretely, they adopt listwise learning [27] and train the student to follow the items' ranking orders predicted by the teacher.

(2) KD by the latent knowledge. Pointing out that the predictions incompletely reveal the teacher's knowledge, a few methods [8, 31][1] have focused on distilling latent knowledge from the teacher's intermediate layer. The existing methods are based on *hint regression* [20] proposed in computer vision. Let $h^t : \mathcal{X} \rightarrow \mathbb{R}^{d^t}$ denote a mapping function from the input feature space to the representation space of the teacher (i.e., the teacher nested function up to the intermediate layer). Similarly, let $h^s : \mathcal{X} \rightarrow \mathbb{R}^{d^s}$ denote a mapping function to the representation space of the student. Also, let $\mathbf{e}_i^t = h^t(\mathbf{x}_i)$ and $\mathbf{e}_i^s = h^s(\mathbf{x}_i)$ denote the representations of entity i from the two spaces[2], where $\mathbf{x}_i$ is entity i's input feature. The hint regression makes $\mathbf{e}_i^s$ approximate $\mathbf{e}_i^t$ as follows:

$$\mathcal{L}_{Hint} = \|\mathbf{e}_i^t - f(\mathbf{e}_i^s)\|_2^2 \tag{1}$$

where $f : \mathbb{R}^{d^s} \rightarrow \mathbb{R}^{d^t}$ is a small network to bridge the different dimensions ($d^s << d^t$). By minimizing $\mathcal{L}_{Hint}$, parameters in the student (i.e., h^s) and f are updated. Also, it is jointly minimized with the base model (i.e., $\mathcal{L}_{Base} + \lambda\mathcal{L}_{Hint}$) which can be any existing recommender. The hint regression enables $\mathbf{e}^s$ to capture compressed information that can restore detailed information in $\mathbf{e}^t$ [8, 31]. [31] adopts this original hint regression to improve the student.

The most recent work DE [8] further elaborates this approach for RS. DE argues that using a single network (f) makes the knowledge of entities having dissimilar preferences get mixed, and this degrades the quality of distillation. Its main idea is that the knowledge of entities having similar preferences should be distilled without being mixed with that of entities having dissimilar preferences. To this end, DE clusters the representations into K groups based on the teacher's knowledge and distills the representations in each group via a separate network f_k. Let $\mathbf{z}_i$ be a K-dimensional one-hot vector whose element $z_{ik} = 1$ if entity i belongs to the corresponding k-th group. For each entity i, DE loss is defined as follows:

$$\mathcal{L}_{DE} = \|\mathbf{e}_i^t - \sum_{k=1}^{K} z_{ik} f_k(\mathbf{e}_i^s)\|_2^2 \tag{2}$$

The one-hot vector is sampled from a categorical distribution with class probabilities $\boldsymbol{\alpha}_i = v(\mathbf{e}_i^t)$, i.e., $\mathbf{z}_i \sim \text{Categorical}_K(\boldsymbol{\alpha}_i)$, where $v : \mathbb{R}^{d^t} \rightarrow \mathbb{R}^K$ is a small network with Softmax output. The sampling process is approximated by Gumbel-Softmax [6] and trained via backpropagation in an end-to-end manner. In sum, the representations belonging to the same group share similar preferences and are distilled via the same network without being mixed with the representations belonging to the different groups [8].

The existing methods [8, 31] based on the hint regression distill the knowledge of individual entity without consideration of how the entities are related in the representation space. Considering a user's preference is represented in relation to (and in contrast with) other users and items, each entity is better understood by its relations to the other entities rather than by its individual representation. Also, the student can take advantage of the space, where the relations found by the teacher are well preserved, in finding more accurate ranking orders among the entities and thereby improving the recommendation performance. In this work, we propose a new distillation approach for RS that directly distills the relational knowledge from the teacher's representation space.

3 METHODOLOGIES

We first provide an overview of the proposed approach (Section 3.1). Before we describe the final solution, we explain a naive method for incorporating the relational knowledge in the distillation process (Section 3.2). Then, we shed light on the drawbacks of the method when applying it for KD. Motivated by the analysis, we present a new method, named HTD, which distills the relational knowledge in multi-levels to effectively cope with a large capacity gap between the teacher and the student (Section 3.3). The pseudocodes of the proposed methods are provided in the appendix.

3.1 Overview of Topology Distillation

The proposed topology distillation approach guides the learning of the student by the topological structure built upon the relational knowledge in the teacher representation space. The relational knowledge refers to all the information on how the representations are correlated in the space; those sharing similar preferences are strongly correlated, whereas those with different preferences are weakly correlated. We build a (weighted) topology of a graph where the nodes are the representations and the edges encode the relatedness of the representations. Then, we distill the relational knowledge by making the student preserve the teacher's topological structure in its representation space. With the proposed approach, the student is trained by minimizing the following loss:

$$\mathcal{L} = \mathcal{L}_{Base} + \lambda_{TD}\mathcal{L}_{TD} \tag{3}$$

where λ_{TD} is a hyperparameter controlling the effects of topology distillation. The base model can be any existing recommender,

[1] [8] proposes two KD methods: one by prediction and the other by latent knowledge.

[2] We use the term 'entity' to denote the subject of each representation.

and $\mathcal{L}_{Base}$ corresponds to its loss function. $\mathcal{L}_{TD}$ is defined on the topology of the representations in the same batch used for the base model training.

3.2 Full Topology Distillation (FTD)

As a straightforward method, we distill the knowledge of the entire relations in the teacher space. Given a batch, we first generate a fully connected graph in the teacher representation space. The graph is characterized by the adjacency matrix $\mathbf{A}^t \in \mathbb{R}^{b\times b}$ where b is the number of representations in the batch. Each element a^t_{ij} is the weight of an edge between entities i and j representing their similarity and is parameterized as follows:

$$a^t_{ij} = \rho(\mathbf{e}^t_i, \mathbf{e}^t_j) \tag{4}$$

where $\rho(\cdot,\cdot)$ is a similarity score such as the cosine similarity or negative Euclidean distance, in this work we use the former. Analogously, we generate a graph, characterized by the adjacency matrix $\mathbf{A}^s \in \mathbb{R}^{b\times b}$, in the student representation space, i.e., $a^s_{ij} = \rho(\mathbf{e}^s_i, \mathbf{e}^s_j)$.

After obtaining the topological structures $\mathbf{A}^t$ and $\mathbf{A}^s$ from the representation space of the teacher and student, respectively, we train the student to preserve the topology discovered by the teacher by the topology-preserving distillation loss as follows:

$$\mathcal{L}_{FTD} = \text{Dist}(\mathbf{A}^t, \mathbf{A}^s) = \|\mathbf{A}^t - \mathbf{A}^s\|_F^2, \tag{5}$$

where $\text{Dist}(\cdot,\cdot)$ is the distance between the topological structures, in this work, we compute it with the Frobenius norm. By minimizing $\mathcal{L}_{FTD}$, parameters in the student are updated. As this method utilizes the full topology as supervision to guide the student, we call it Full Topology Distillation (FTD). Substituting the distillation loss $\mathcal{L}_{TD}$ in Equation 3 derives the final loss for training the student.

Issues: Although FTD directly transfers the relational knowledge which is ignored in the previous work, it still has some clear drawbacks. Because the student has a very limited capacity compared to the teacher, it is often daunting for the student to learn all the relational knowledge in the teacher. Indeed, we observe that sometimes FTD even hinders the learning of the student and degrades the recommendation performance (reported in Section 4.2). Therefore, the relational knowledge should be distilled with consideration of the huge capacity gap.

3.3 Hierarchical Topology Distillation (HTD)

Our key idea to tackle the issues is to decompose the whole topology hierarchically so as to be effectively transferred to the student. We argue that the student should focus on learning the relations among the strongly correlated entities that share similar preferences and accordingly have a direct impact on top-N recommendation performance. To this end, we summarize the numerous relations among the weakly correlated entities, enabling the student to better focus on the important relations.

During the training, HTD adaptively finds *preference groups* of strongly correlated entities. Then, the topology is hierarchically structured in group-level and entity-level: 1) *group-level topology* includes the summarized relations across the groups, providing the overview of the entire topology. 2) *entity-level topology* includes the relations of entities belonging to the same group. This provides fine-grained supervision on important relations of the entities having similar preferences. By compressing the complex individual relations across the groups, HTD relaxes the daunting supervision, effectively distills the relational knowledge to the student.

3.3.1 *Preference Group Assignment.*

To find the groups of entities having a similar preference in an end-to-end manner considering both the teacher and the student, we borrow the idea of DE [8]. Formally, let there exist K preference groups in the teacher space. We use a small network $v : \mathbb{R}^{d^t} \rightarrow \mathbb{R}^K$ with Softmax output to compute the assignment probability vector $\boldsymbol{\alpha}_i \in \mathbb{R}^K$ for each entity i as follows:

$$\boldsymbol{\alpha}_i = v(\mathbf{e}^t_i), \tag{6}$$

where each element α_{ik} encodes the probability of the entity i to be assigned to k-th preference group. Let $\mathbf{z}_i$ be a K-dimensional one-hot assignment vector whose element $z_{ik} = 1$ if entity i belongs to the corresponding k-th group. We assign a group for each entity by sampling the assignment vector from a categorical distribution parameterized by $\{\alpha_{ik}\}$ i.e., $p(z_{ik} = 1 \mid v, \mathbf{e}^t_i) = \alpha_{ik}$. To make the sampling process differentiable, we adopt Gumbel-Softmax [6] which is a continuous distribution on the simplex that can approximate samples from a categorical distribution.

$$z_{ik} = \frac{\exp\left(\left(\alpha_{ik} + g_k\right)/\tau\right)}{\sum_{j=1}^{K}\exp\left(\left(\alpha_{ij} + g_j\right)/\tau\right)} \quad \text{for} \quad k = 1, ..., K, \tag{7}$$

where g_j is the gumbel noise drawn from $\text{Gumbel}(0,1)$ distribution [6] and τ is the temperature parameter. We set a small value on τ so that samples from the Gumbel-Softmax distribution become one-hot vector [6]. This group assignment is evolved during the training via backpropagation [8]. This will be explained in Section 3.3.4.

With the assignment process, for a given batch, HTD obtains the grouping information summarized by a $b \times K$ assignment matrix $\mathbf{Z}$ where each row corresponds to the one-hot assignment vector for each entity. We also denote the set of entities belonging to each group by $G_k = \{i | z_{ik} = 1\}$. Note that the group assignment is based on the teacher's knowledge. Based on the assignment, we decompose the topology hierarchically.

3.3.2 *Group-level topology.*

HTD introduces a *prototype* representing the entities in each preference group, then use it to summarize the relations across the groups. Let $\mathbf{E}^t \in \mathbb{R}^{b\times d^t}$ and $\mathbf{E}^s \in \mathbb{R}^{b\times d^s}$ denote the representation matrix in the teacher space and the student space, respectively. The prototypes $\mathbf{P}^t \in \mathbb{R}^{K\times d^t}$ and $\mathbf{P}^s \in \mathbb{R}^{K\times d^s}$ are defined as follows:

$$\mathbf{P}^t = \tilde{\mathbf{Z}}^\top \mathbf{E}^t \quad \text{and} \quad \mathbf{P}^s = \tilde{\mathbf{Z}}^\top \mathbf{E}^s, \tag{8}$$

where $\tilde{\mathbf{Z}}$ is normalized assignment matrix by the number of entities in each group (i.e., $\tilde{\mathbf{Z}}_{[:,i]} = \mathbf{Z}_{[:,i]}/\sum_i \mathbf{Z}_{[:,i]}$). For $\mathbf{P}$, each row $\mathbf{P}_{[k,:]}$ corresponds to the average representation for the entities belonging to each group k, and we use it as a prototype representing the group.

With the prototypes, we consider two design choices with different degrees of relaxation. In the first choice, we distill the relations between the prototypes. We build the topology characterized by the $K \times K$ matrix $\mathbf{H}^t$ which contains the relations as:

$$h^t_{km} = \rho(\mathbf{P}^t_{[k,:]}, \mathbf{P}^t_{[m,:]}), \tag{9}$$

where $k, m \in \{1, ..., K\}$.

In the second choice, we distill the relations between each prototype and entities belonging to the other groups. We build the topology characterized by the $K \times b$ matrix $\mathbf{H}^t$ which contains the relations as:

$$h^t_{kj} = \rho(\mathbf{P}^t_{[k,:]}, \mathbf{e}^t_j), \tag{10}$$

where $k \in \{1, ..., K\}$, $j \in \{1, ..., b\}$. It is worth noting that we only distill the relations across the groups (i.e., $j \notin G_k$). Using one of the choices, we build the group-level topological structure $\mathbf{H}^t$ in the teacher space, and analogously, we build $\mathbf{H}^s$ in the student space.

The first choice puts a much higher degree of relaxation compared to the second choice. For instance, assume that there are two groups of entities (i.e., G_1 and G_2). Without the hierarchical approach (as done in FTD), there exists $|G_1| \times |G_2|$ relations across the groups. With the first choice, they are summarized to a single relation between two prototypes, and with the second choice, they are summarized to $|G_1| + |G_2|$ relations. We call the first choice as Group(P,P) and the second choice as Group(P,e) and provide results with each choice in Section 4.2.

3.3.3 *Entity-level topology.*

HTD distills the full relations among the strongly correlated entities in the same group. In the teacher space, the entity-level topology contains the following relations:

$$\{\rho(\mathbf{e}^t_i, \mathbf{e}^t_j) \mid (i, j) \in G_k \times G_k\}, \quad \text{for } k \in \{1, ..., K\}, \tag{11}$$

and analogously, we build the entity-level topology in the student space. For an efficient computation on matrix form, we introduce the $b \times b$ binary indicator matrix $\mathbf{M} = \mathbf{Z}\mathbf{Z}^\top$ indicating whether each relation is contained in the topology or not. Intuitively, each element $m_{ij} = 1$ if entity i and j are assigned to the same group, otherwise $m_{ij} = 0$. Then, the entity-level topology is defined by $\mathbf{A}^t$ with $\mathbf{M}$ in the teacher space and also defined by $\mathbf{A}^s$ with $\mathbf{M}$ in the student space. The distance between two topological structures is simply computed by $\|\mathbf{M} \odot (\mathbf{A}^t - \mathbf{A}^s)\|^2_F$ where $\odot$ is the Hadamard product.

3.3.4 *Optimization*.

HTD guides the student with the decomposed topological structures. The loss function is defined as follows:

$$\begin{aligned}\mathcal{L}_{HTD} &= \gamma \left(\|\mathbf{H}^t - \mathbf{H}^s\|^2_F + \|\mathbf{M} \odot (\mathbf{A}^t - \mathbf{A}^s)\|^2_F\right) \\ &+ (1-\gamma)\left(\sum_{i=1}^{b} \|\mathbf{e}^t_i - \sum_{k=1}^{K} z_{ik} f_k(\mathbf{e}^s_i)\|^2_2\right),\end{aligned} \tag{12}$$

where the first term corresponds to the topology-preserving loss, the second term corresponds to the hint regression loss adopted in DE [8] that makes the group assignment process differentiable. We put a network f_k for each k-th group, then train each network to reconstruct the representations belonging to the corresponding group, which makes the entities having strong correlations get distilled by the same network [8]. γ is a hyperparameter balancing the two terms. In this work, we set 0.5 to γ. By minimizing $\mathcal{L}_{HTD}$, parameters in the student, v and f_* are updated. Note that v and f_* are not used in the inference phase, they are only utilized for the distillation in the offline training phase. Substituting the distillation loss $\mathcal{L}_{TD}$ in Equation 3 derives the final loss for training the student.

Effects of HTD. For more intuitive understanding, we provide a visualization of the relational knowledge distilled to the student in Figure 2. We randomly choose a preference group and visualize the relations *w.r.t.* the entities belonging to the group. Note that there exist the same number of intra-group relations (red) in both figures. Without the hierarchical approach (as done in FTD), the student is forced to learn a huge number of relations with the entities belonging to the other groups (blue). On the other hand, HTD summarizes the numerous relations, enabling the student to better focus on learning the detailed intra-group relations, which directly affects in improving the recommendation performance.

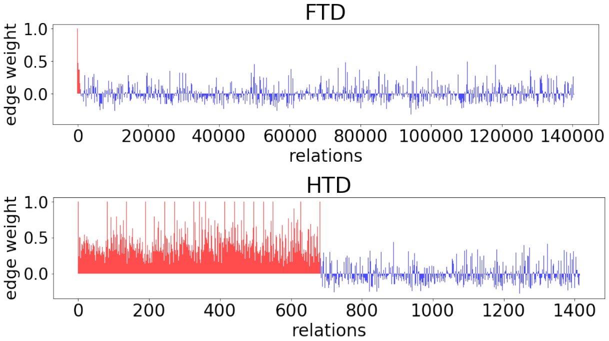


Figure 2: The relational knowledge distilled from teacher to student by FTD and HTD (with Group(P,e)). Red/Blue corresponds to relations of the entities belonging to the same/different preference group(s) (BPR on CiteULike).

Discussions on Group Assignment. Note that HTD is not limited to a specific group assignment method, i.e., DE. Any method that clusters the teacher representation space or prior knowledge of user/item groups (e.g., item category, user demographic features) can be utilized for more sophisticated topology decomposition, which further improves the effectiveness of HTD. The comparison with another assignment method is provided in the appendix.

4 EXPERIMENTS

We validate the proposed approach on **18** experiment settings: 2 real-world datasets × 3 base models × 3 different student model sizes (Section 4.1). We first present comparison results with the state-of-the-art competitor and a detailed ablation study (Section 4.2). We also provide in-depth analyses to verify the benefit of distilling the topological structure (Section 4.3). Lastly, we provide a hyperparameter study (Section 4.4).

4.1 Experimental Setup

We closely follow the experiment setup of DE [8]. However, for a thorough evaluation, we make two changes in the setup. **1)** we add LightGCN [3], which is the state-of-the-art top-N recommendation method, as a base model. **2)** unlike [8] that samples negative items for evaluation, we adopt the full-ranking evaluation which enables more rigorous evaluation. Refer to the appendix for more detail.

4.1.1 *Datasets.*

We use two real-world datasets: CiteULike and Foursquare. CiteULike contains tag information for each item, and Foursquare contains GPS coordinates for each item. We use the side information to evaluate the quality of representations induced by each KD method. More details of the datasets are provided in the appendix.

4.1.2 Base Models. We evaluate the proposed approach on base models having different architectures and learning strategies, which are widely used for top-N recommendation task.

- **BPR [19]**: A learning-to-rank model that models user-item interaction with Matrix Factorization (MF).
- **NeuMF [4]**: A deep model that combines MF and Multi-Layer Perceptron (MLP) to learn the user-item interaction.
- **LightGCN [3]**: The state-of-the-art model which adopts simplified Graph Convolution Network (GCN) to capture the information of multi-hop neighbors.

4.1.3 Teacher/Student. For each setting, we increase the model size until the recommendation performance is no longer improved and adopt the model with the best performance as **Teacher** model. Then, we build three student models by limiting the model size, i.e., $\phi \in \{0.1, 0.5, 1.0\}$. We call the student model trained without distillation as **Student** in this section. Figure 3 summarizes the model size and inference time. The inferences are made using PyTorch with CUDA from TITAN Xp GPU and Xeon on Gold 6130 CPU. It shows that the smaller model has lower inference latency.

4.1.4 Compared Methods. We compare the following KD methods distilling the latent knowledge from the intermediate layer of the teacher recommender.

- **FitNet [20]**: A KD method utilizing the original hint regression.
- **Distillation Experts (DE) [8]**: The state-of-the-art KD method distilling the latent knowledge. DE elaborates the hint regression.
- **Full Topology Distillation (FTD)**: A proposed method that distills the full topology (Section 3.2).
- **Hierarchical Topology Distillation (HTD)**: A proposed method that distills the hierarchical topology (Section 3.3).

Note that we do not include the methods distilling the predictions (e.g., RD [22], CD [12], and RRD [8]) in the competitors, because they are not competing with the methods distilling the latent knowledge [8]. Instead, we provide experiment results when they are combined with the proposed approach in Section 4.2.

4.2 Performance Analysis

Table 1 presents top-N recommendation performance of the methods compared ($\phi = 0.1$), and Figure 4 presents results with three different students sizes. For the group-level topology of HTD, we choose Group(P,e), since it consistently shows better results than Group(P,P). The detailed comparisons along with other various ablations are reported in Table 2. Lastly, results with prediction-based KD method are presented in Figure 5 and Figure 6.

Overall Evaluation. In Table 1, we observe that HTD achieves significant performance gains compared to the main competitor, i.e., DE. This result shows that distilling the relational knowledge provides better guidance than only distilling the knowledge of individual representation. We also observe that FTD is not always effective and sometimes even degrades the student's recommendation performance (e.g., LightGCN on CiteULike). As the student's size is highly limited compared to the teacher in the KD scenario, learning all the relational knowledge is daunting for the student, which leads to degrade the effects of distillation. This result supports our claim that the relational knowledge should be distilled considering the huge capacity gap. Also, the results show that HTD successfully copes with the issue, enabling the student to effectively learn the relational knowledge.

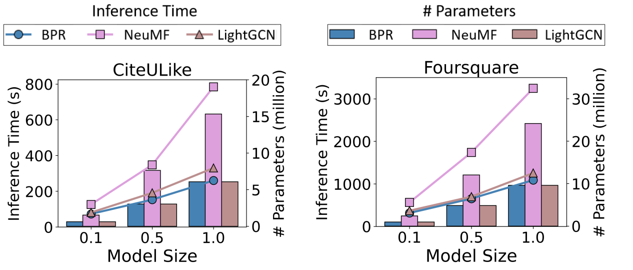
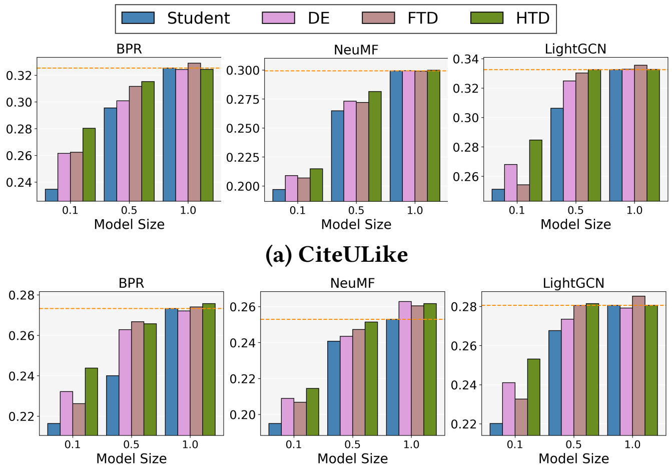


Figure 3: Inference time (s) and model size (ϕ). Inference time denotes the wall time used for generating recommendation list for every user.



(a) CiteULike

(b) Foursquare

Figure 4: Recall@50 across three different student sizes (Dotted line: Teacher)

In Figure 4, we observe that as the model size increases, the performance gap between FTD and HTD decreases. FTD achieves comparable and even higher performance than HTD when the student has enough capacity (e.g., LightGCN on Foursquare with $\phi = 1.0$). This result again verifies that the effectiveness of the proposed topology distillation approach. It also shows that FTD can be applied to maximize the performance of recommender in the scenario where there is no constraint on the model size by self-distillation.

Comparison with ablations. In Table 2, we provide the comparison with diverse ablations. For FTD, we report the results when it is used with the state-of-the-art hint regression method (denoted as **FTD+DE**). As HTD includes DE for the group assignment, comparison with FTD+DE shows the direct impacts of topology relaxation. We observe that FTD+DE is not as effective as HTD and sometimes even achieves worse performance than DE. This shows that the power of HTD comes from the distillation strategy transferring the relational knowledge in multi-levels.

For HTD, we compare three ablations: **1) Group only** that considers only group-level topology, **2) Entity only** that considers

Table 1: Performance comparison ($\phi = 0.1$). *Gain.DE* denotes the improvement of HTD over DE, *Gain.S* denotes the improvement of HTD over Student. HTD achieves statistically significant improvements over the best baseline. We use the paired t-test with significance level at 0.05 on Recall@50.

Dataset	Base Model	Method	Recall@10	NDCG@10	Recall@20	NDCG@20	Recall@50	NDCG@50
CiteULike	BPR	Teacher	0.1533	0.0883	0.2196	0.1058	0.3253	0.1247
		Student	0.1014	0.0560	0.1506	0.0684	0.2347	0.0864
		FitNet	0.1097	0.0595	0.1610	0.0738	0.2521	0.0924
		DE	0.1165	0.0645	0.1696	0.0778	0.2615	0.0960
		FTD	0.1131	0.0630	0.1660	0.0763	0.2624	0.0953
		HTD	**0.1247**	**0.0691**	**0.1820**	**0.0836**	**0.2803**	**0.1031**
		Gain.DE	7.0%	7.3%	7.3%	7.5%	7.2%	7.4%
		Gain.S	23.0%	23.5%	20.9%	22.3%	19.4%	19.3%
	NeuMF	Teacher	0.1487	0.0844	0.2048	0.0986	0.2993	0.1155
		Student	0.0856	0.0449	0.1249	0.0553	0.1970	0.0697
		FitNet	0.0856	0.0469	0.1275	0.0576	0.2020	0.0723
		DE	0.0882	0.0475	0.1306	0.0581	0.2090	0.0736
		FTD	0.0875	0.0474	0.1291	0.0579	0.2069	0.0733
		HTD	**0.0914**	**0.0504**	**0.1416**	**0.0618**	**0.2154**	**0.0772**
		Gain.DE	3.6%	6.2%	8.4%	6.4%	3.1%	4.8%
		Gain.S	6.8%	12.2%	13.4%	11.8%	9.0%	10.8%
	LightGCN	Teacher	0.1610	0.0934	0.2274	0.1091	0.3326	0.1299
		Student	0.1125	0.0618	0.1642	0.0748	0.2512	0.0944
		FitNet	0.1151	0.0642	0.1710	0.0783	0.2653	0.0969
		DE	0.1189	0.0664	0.1733	0.0801	0.2680	0.0988
		FTD	0.1112	0.0615	0.1635	0.0747	0.2542	0.0926
		HTD	**0.1322**	**0.0742**	**0.1902**	**0.0888**	**0.2847**	**0.1075**
		Gain.DE	11.2%	11.8%	9.7%	10.9%	6.2%	8.8%
		Gain.S	17.5%	20.1%	15.8%	18.7%	13.3%	13.9%
Foursquare	BPR	Teacher	0.1187	0.0695	0.1700	0.0825	0.2732	0.1028
		Student	0.0911	0.0544	0.1333	0.0648	0.2164	0.0809
		FitNet	0.0957	0.0564	0.1386	0.0672	0.2258	0.0845
		DE	0.0979	0.0567	0.1434	0.0681	0.2322	0.0856
		FTD	0.0987	0.0582	0.1417	0.0690	0.2262	0.0857
		HTD	**0.1037**	**0.0622**	**0.1505**	**0.0740**	**0.2438**	**0.0921**
		Gain.DE	5.9%	9.7%	5.0%	8.7%	5.0%	7.6%
		Gain.S	13.8%	14.3%	12.9%	14.2%	12.7%	13.8%
	NeuMF	Teacher	0.1060	0.0590	0.1546	0.0716	0.2529	0.0910
		Student	0.0737	0.0393	0.1125	0.0490	0.1950	0.0653
		FitNet	0.0829	0.0462	0.1243	0.0564	0.2062	0.0729
		DE	0.0855	0.0476	0.1255	0.0576	0.2089	0.0741
		FTD	0.0823	0.0451	0.1233	0.0554	0.2068	0.0719
		HTD	**0.0891**	**0.0501**	**0.1294**	**0.0601**	**0.2152**	**0.0770**
		Gain.DE	4.3%	5.3%	3.1%	4.3%	3.0%	3.9%
		Gain.S	20.9%	27.2%	15.0%	22.7%	10.0%	17.9%
	LightGCN	Teacher	0.1259	0.0730	0.1779	0.0865	0.2806	0.1067
		Student	0.0951	0.0564	0.1372	0.0670	0.2202	0.0834
		FitNet	0.0993	0.0587	0.1431	0.0697	0.2315	0.0872
		DE	0.1051	0.0617	0.1503	0.0731	0.2410	0.0910
		FTD	0.1018	0.0602	0.1466	0.0714	0.2327	0.0884
		HTD	**0.1119**	**0.0652**	**0.1597**	**0.0772**	**0.2531**	**0.0956**
		Gain.DE	6.4%	5.6%	6.3%	5.6%	5.0%	5.1%
		Gain.S	17.7%	15.6%	16.4%	15.2%	14.9%	14.6%

Table 2: Performance comparison with ablations ($\phi = 0.1$).

Base Model	Method	CiteULike		Foursquare	
		Recall@50	NDCG@50	Recall@50	NDCG@50
BPR	FTD	0.2624	0.0953	0.2262	0.0857
	FTD+DE	0.2650	0.0969	0.2339	0.0867
	HTD	**0.2803**	**0.1031**	**0.2438**	**0.0921**
	Group only	0.2619	0.0948	0.2349	0.0874
	Entity only	0.2608	0.0968	0.2361	0.0871
	Group (P,P)	0.2648	0.0982	0.2399	0.0887
LightGCN	FTD	0.2542	0.0926	0.2327	0.0884
	FTD+DE	0.2572	0.0931	0.2335	0.0872
	HTD	**0.2847**	**0.1075**	**0.2531**	**0.0956**
	Group only	0.2596	0.0976	0.2432	0.0910
	Entity only	0.2709	0.1010	0.2453	0.0910
	Group (P,P)	0.2683	0.0987	0.2459	0.0909

only entity-level topology, and **3) Group (P,P)** that considers the relations of the prototypes only for the group-level topology (Section 3.3.2). We first observe that both group-level and entity-level topology are indeed necessary. Without either of them, the performance considerably drops. The group-level topology includes the summarized relations across the groups, providing the overview of the entire topology. On the other hand, the entity-level topology includes the full relations in each group, providing fine-grained supervision of how the entities should be correlated. Based on both two-level topology, HTD effectively transfers the relational knowledge. Lastly, we observe that Group (P,P) is not as effective as Group(P,e) adopted in HTD. We conjecture that summarizing numerous relations across the groups into a single relation may lose too much information and cannot effectively boost the student.

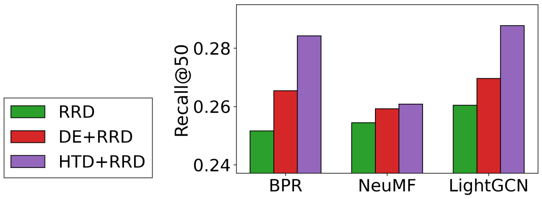


Figure 5: Performance comparison with RRD.

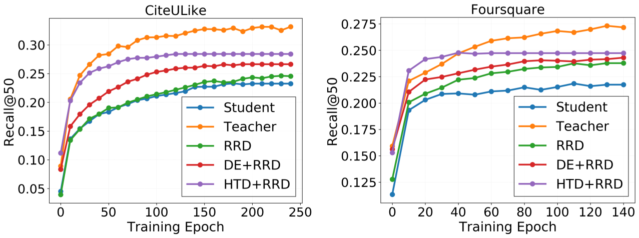


Figure 6: Training curves with RRD.

With prediction-based KD method. We report the results with the state-of-the-art prediction KD method (i.e., RRD [8]) on CiteULike with $\phi = 0.1$ in Figure 5. Also, we provide the training curves of BPR with $\phi = 0.1$ in Figure 6[3]. First, we observe that the effectiveness of RRD is considerably improved when it is applied with the KD method distilling the latent knowledge (i.e., DE and HTD). This result aligns with the results reported in [8] and shows the importance of distilling the latent knowledge. Second, we observe that the student recommender achieves the best performance with HTD. Unlike RRD, which makes the student imitate the ranking order of items in each user's recommendation list, HTD distills relations existing in the representation space via the topology matching. The topology includes much rich supervision including not only user-item relations but also *user-user* relations and *item-item* relations. By obtaining this additional information in a proper manner considering the capacity gap, the student can be fully improved with HTD.

4.3 Benefit of Topology Distillation

To further ascertain the benefit of the topology distillation, we provide in-depth analysis on representations obtained by each KD method ($\phi = 0.1$).

First, we evaluate whether the topology distillation indeed makes the student better preserve the relations in the teacher representation space than the existing method. For quantitative evaluation, we conduct the following steps: **1)** In the teacher space, for each representation, we compute the similarity distributions with 100 *most similar* representations and 100 *randomly selected* representations, respectively. **2)** In the student space, for each representation, we compute the similarity distributions with the representations chosen in the teacher space. **3)** We compute KL divergence for each distribution and report the average value in Figure 7. KL divergence of 'Most similar' indicates how well the detailed relations among the strongly correlated representations are preserved, and that of 'Random' indicates how well the overall relations in the space are preserved. We observe that HTD achieves the lowest KL divergence for both Most similar and Random, which shows that HTD indeed enables the student to better preserve the relations in the teacher space.

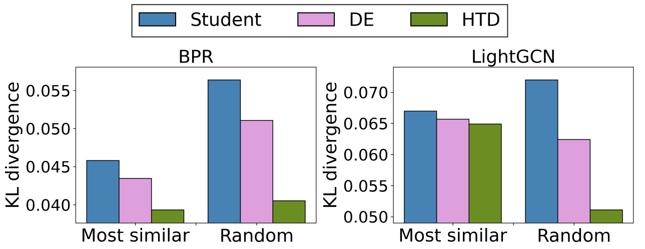


Figure 7: The average KL divergence from similarity distributions obtained in the teacher representation spaces.

Table 3: Performance comparison on downstream tasks.

Base Model	Type	Method	Tag Retrieval (Recall@10)	Region Classification (Accuracy)
BPR	linear	Teacher	0.3121±2.7e-3	0.6531±2.5e-3
		Student	0.2421±1.7e-3	0.4222±7.4e-3
		DE	0.2542±1.5e-3	0.5448±3.2e-3
		HTD	**0.2635**±1.2e-3	**0.5653**±5.0e-3
	non-linear	Teacher	0.3123±1.4e-3	0.6357±8.8e-3
		Student	0.2701±2.1e-3	0.4224±7.9e-3
		DE	0.2807±2.5e-3	0.5123±6.6e-3
		HTD	**0.2909**±2.2e-3	**0.5318**±3.5e-3
LightGCN	linear	Teacher	0.3489±0.6e-3	0.6787±6.1e-3
		Student	0.2523±0.7e-3	0.4635±3.1e-3
		DE	0.2565±1.2e-3	0.5654±6.5e-3
		HTD	**0.2650**±0.3e-3	**0.5854**±7.7e-3
	non-linear	Teacher	0.3360±1.2e-3	0.6504±5.9e-3
		Student	0.2971±0.3e-3	0.4354±6.5e-3
		DE	0.3053±1.3e-3	0.5250±3.4e-3
		HTD	**0.3138**±1.1e-3	**0.5485**±8.1e-3

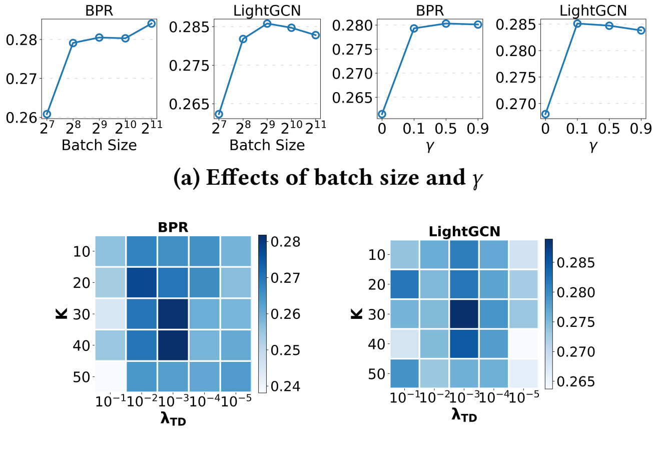
Second, we compare the performance of two downstream tasks that evaluate how well each method encodes the items' characteristics (or semantics) into the representations. We perform the tag retrieval task for CiteULike and the region classification task for Foursquare. We train a linear and a non-linear model to predict the tag/region of each item by using the fixed item representation as the input. The detailed setup is provided in the appendix. In Table 3, we observe that HTD achieves consistently higher performance than DE on both of two downstream tasks. This strongly indicates that the representation space induced by HTD more accurately captures the item's semantics compared to the space induced by DE.

In sum, with the topology distillation approach, the student can indeed better preserve the relations in the teacher space. This not only improves the recommendation performance but also allows it to better capture the semantic of entities.

4.4 Hyperparameter Analysis

We provide analyses to guide the hyperparameter selection of the topology distillation approach. For the sake of the space, we report the results of BPR and LightGCN on CiteULike dataset with $\phi = 0.1$ (Figure 8). **1)** The batch size is an important factor affecting the performance of the topology distillation. When the batch size is too small, the topology cannot include the overall relational knowledge in the representation space, leading to limited performance. For CiteULike, we observe that HTD achieves the stable performance around 2^8-2^{11}. In this work, we set the batch size to 2^{10}. **2)** γ is a hyperparameter for balancing the topology-preserving loss and the

[3] After the early stopping on the validation set, we plot the final performance.



(a) Effects of batch size and γ

(b) Effects of λ_{TD} and K

Figure 8: Effects of the hyperparameters (Recall@50).

regression loss. We observe the stable performance with a value larger than 0.1. In this work, we set γ to 0.5. Note that $\gamma = 0$ equals DE. **3)** The number of preference groups (K) is an important hyperparameter of HTD. It needs to be determined considering the dataset, the capacity gap, and the selected layer for the distillation. For this setup, the best performance is achieved when K is around 30-40 in both base models. **4)** λ_{TD} is a hyperparameter for controlling the effects of topology distillation. For this setup, the best performance is achieved when λ_{TD} is around 10^{-3} in both base models.

5 CONCLUSION AND FUTURE WORK

We develop a general topology distillation approach for RS, which guides the learning of the student by the topological structure built upon the relational knowledge in the teacher representation space. Concretely, we propose two topology distillation methods: 1) FTD that transfers the full topology. FTD is used in the scenario where the student has enough capacity to learn all the teacher's knowledge. 2) HTD that transfers the decomposed topology hierarchically. HTD is adopted in the conventional KD scenario where the student has a very limited capacity compared to the teacher. We conduct extensive experiments on real-world datasets and show that the proposed approach consistently outperforms the state-of-the-art competitor. We also provide in-depth analyses to ascertain the benefit of distilling the topology.

We believe the topology distillation approach can be advanced and extended in several directions. First, layer selection and simultaneous distillation from multiple layers are not investigated in this work. We especially expect that this can further improve the limited improvements by the topology distillation in the deep model (i.e., NeuMF). Second, topology distillation across different base models (e.g., from LightGCN to BPR) can be also considered to further improve the performance. Lastly, prior knowledge of user/item groups (e.g., item category, user demographic features) can be utilized for more sophisticated topology decomposition. We expect that this can further improve the effectiveness of the proposed method by providing better supervision on relational knowledge.

Acknowledgment: This work was supported by the NRF grant funded by the MSIT (No. 2020R1A2B5B03097210), and the IITP grant funded by the MSIT (No. 2018-0-00584, 2019-0-01906).

REFERENCES

[1] Yoram Bachrach, Yehuda Finkelstein, Ran Gilad-Bachrach, Liran Katzir, Noam Koenigstein, Nir Nice, and Ulrich Paquet. 2014. Speeding up the xbox recommender system using a euclidean transformation for inner-product spaces. In *RecSys*.

[2] Jon Louis Bentley. 1975. Multidimensional binary search trees used for associative searching. *Commun. ACM* (1975).

[3] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *SIGIR*.

[4] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*.

[5] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[6] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144* (2016).

[7] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *TOIS* 20, 4 (2002).

[8] SeongKu Kang, Junyoung Hwang, Wonbin Kweon, and Hwanjo Yu. 2020. DE-RRD: A Knowledge Distillation Framework for Recommender System. In *CIKM*.

[9] Walid Krichene and Steffen Rendle. 2020. On sampled metrics for item recommendation. In *KDD*.

[10] Wonbin Kweon, SeongKu Kang, and Hwanjo Yu. 2021. Bidirectional Distillation for Top-K Recommender System. In *WWW*.

[11] Dongha Lee, SeongKu Kang, Hyunjun Ju, Chanyoung Park, and Hwanjo Yu. 2021. Bootstrapping User and Item Representations for One-Class Collaborative Filtering. In *SIGIR*.

[12] Jaewoong Lee, Minjin Choi, Jongwuk Lee, and Hyunjung Shim. 2019. Collaborative Distillation for Top-N Recommendation. *ICDM* (2019).

[13] Hui Li, Tsz Nam Chan, Man Lung Yiu, and Nikos Mamoulis. 2017. FEXIPRO: fast and exact inner product retrieval in recommender systems. In *SIGMOD*.

[14] Huayu Li, Richang Hong, Defu Lian, Zhiang Wu, Meng Wang, and Yong Ge. 2016. A Relaxed Ranking-Based Factor Model for Recommender System from Implicit Feedback. In *IJCAI*.

[15] Xiaojie Li, Jianlong Wu, Hongyu Fang, Yue Liao, Fei Wang, and Chen Qian. 2020. Local correlation consistency for knowledge distillation. In *ECCV*. Springer.

[16] Defu Lian, Rui Liu, Yong Ge, Kai Zheng, Xing Xie, and Longbing Cao. 2017. Discrete Content-Aware Matrix Factorization. In *KDD*.

[17] Han Liu, Xiangnan He, Fuli Feng, Liqiang Nie, Rui Liu, and Hanwang Zhang. 2018. Discrete factorization machines for fast feature-based recommendation. *arXiv preprint arXiv:1805.02232* (2018).

[18] Yifan Liu, Ke Chen, Chris Liu, Zengchang Qin, Zhenbo Luo, and Jingdong Wang. 2019. Structured knowledge distillation for semantic segmentation. In *CVPR*.

[19] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*.

[20] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2014. Fitnets: Hints for thin deep nets. In *arXiv*.

[21] Anshumali Shrivastava and Ping Li. 2014. Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS). In *NeurIPS*.

[22] Jiaxi Tang and Ke Wang. 2018. Ranking distillation: Learning compact ranking models with high performance for recommender system. In *KDD*.

[23] Christina Teflioudi, Rainer Gemulla, and Olga Mykytiuk. 2015. Lemp: Fast retrieval of large entries in a matrix product. In *SIGMOD*.

[24] Frederick Tung and Greg Mori. 2019. Similarity-preserving knowledge distillation. In *ICCV*.

[25] Saúl Vargas, Craig Macdonald, and Iadh Ounis. 2015. Analysing compression techniques for in-memory collaborative filtering. (2015).

[26] Haoyu Wang, Defu Lian, and Yong Ge. 2019. Binarized collaborative filtering with distilling graph convolutional networks. *IJCAI* (2019).

[27] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. 2008. Listwise approach to learning to rank: theory and algorithm. In *ICML*.

[28] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. 2017. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *CVPR*.

[29] Sergey Zagoruyko and Nikos Komodakis. 2017. Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer. In *ICLR*.

[30] Hanwang Zhang, Fumin Shen, Wei Liu, Xiangnan He, Huanbo Luan, and Tat-Seng Chua. 2016. Discrete Collaborative Filtering. In *SIGIR*.

[31] Jieming Zhu, Jinyang Liu, Weiqi Li, Jincai Lai, Xiuqiang He, Liang Chen, and Zibin Zheng. 2020. Ensembled CTR Prediction via Knowledge Distillation. In *CIKM*.

A APPENDIX

A.1 Pseudocode of the proposed methods

The pseudocode of FTD and HTD are provided in Algorithm 1 and Algorithm 2, respectively. The base model can be any existing recommender and $\mathcal{L}_{Base}$ is its loss function. Note that the distillation is conducted in the offline training phase. At the online inference phase, the student model is used only.

All the computations of the topology distillation are efficiently computed on matrix form by parallel execution through GPU processor. In Algorithm 1 (FTD), the topological structures (line 5) are computed as follows:

$$\mathbf{A}^t = \text{Cos}(\mathbf{E}^t, \mathbf{E}^t) \ \text{ and } \ \mathbf{A}^s = \text{Cos}(\mathbf{E}^s, \mathbf{E}^s),$$

where Cos is the operation computing the cosine similarities by $\text{Cos}(\mathbf{B}, \mathbf{D}) = \hat{\mathbf{B}}\hat{\mathbf{D}}^\top$, $\hat{\mathbf{B}}_{[i,:]} = \mathbf{B}_{[i,:]}/\|\mathbf{B}_{[i,:]}\|_2$. In Algorithm 2 (HTD with Group(P,e)), the two-level topological structures (line 5-8) are computed as follows:

$$\mathbf{Z} = \text{Gumbel-Softmax}(v(\mathbf{E}^t))$$
$$\mathbf{P}^t = \tilde{\mathbf{Z}}^\top\mathbf{E}^t \ \text{ and } \ \mathbf{P}^s = \tilde{\mathbf{Z}}^\top\mathbf{E}^s$$
$$\mathbf{H}^t = \text{Cos}(\mathbf{P}^t, \mathbf{E}^t) \ \text{ and } \ \mathbf{H}^s = \text{Cos}(\mathbf{P}^s, \mathbf{E}^s)$$
$$\mathbf{A}^t = \text{Cos}(\mathbf{E}^t, \mathbf{E}^t) \ \text{ and } \ \mathbf{A}^s = \text{Cos}(\mathbf{E}^s, \mathbf{E}^s)$$
$$\mathbf{M} = \mathbf{Z}\mathbf{Z}^\top$$

The power of topology distillation comes from distilling the additional supervision in a proper manner considering the capacity gap.

A.2 Group Assignment.

Instead of DE, any clustering method or prior knowledge of user/item groups can be utilized for more sophisticated topology decomposition. The simplest method is K-means clustering. Specifically, we first conduct the clustering in the teacher space, then use the results for the group assignment. The results are summarized in Table 4. For both DE and K-means, the number of preference group (K) is set to 30, and the teacher space dimensions (d^t) is 200.

We get consistently better results with the adaptive assignment by DE. We conjecture the possible reasons as follows: 1) Accurate clustering in high-dimensional space is very challenging, 2) With the adaptive approach, the assignment process gets gradually sophisticated along with the student, and thereby it provides guidance considering the student's learning. For these reasons, we use the adaptive assignment in HTD. The performance of HTD can be further improved by adopting a more advanced assignment method and prior knowledge. We leave exploring better ways of the assignment for future study.

Table 4: Performance comparison with different group assignment methods.

	CiteULike		Foursquare	
	Recall@50	NDCG@50	Recall@50	NDCG@50
K-means	0.2661	0.0980	0.2346	0.0871
DE	0.2803	0.1031	0.2438	0.0921

Algorithm 1: Full Topology Distillation.

```
Input  : Training data $\mathcal{D}$, Trained Teacher model
Output : Student model
1  Initialize Student model
2  while not convergence do
3      for each batch $\mathcal{B} \in \mathcal{D}$ do
4          Compute $\mathcal{L}_{Base}$
           /* Topology Distillation                         */
5          Build full topology $\mathbf{A}^t$ and $\mathbf{A}^s$
6          Compute $\mathcal{L}_{FTD}$                      // Eqn. 5
7          Compute $\mathcal{L} = \mathcal{L}_{Base} + \lambda\mathcal{L}_{FTD}$     // Eqn. 3
8          Update Student model by minimizing $\mathcal{L}$
```

Algorithm 2: Hierarchical Topology Distillation.

```
Input  : Training data $\mathcal{D}$, Trained Teacher model
Output : Student model
1  Initialize Student model, $v$, and $f_*$
2  while not convergence do
3      for each batch $\mathcal{B} \in \mathcal{D}$ do
4          Compute $\mathcal{L}_{Base}$
           /* Topology Distillation                         */
5          Assign preference groups $\mathbf{Z}$ for $\mathcal{B}$
6          Compute prototypes $\mathbf{P}^t$ and $\mathbf{P}^s$
7          Build group-level topology $\mathbf{H}^t$ and $\mathbf{H}^s$
8          Build entity-level topology by $\mathbf{A}^t$, $\mathbf{A}^s$, and $\mathbf{M}$
9          Compute $\mathcal{L}_{HTD}$                      // Eqn. 12
10         Compute $\mathcal{L} = \mathcal{L}_{Base} + \lambda\mathcal{L}_{HTD}$     // Eqn. 3
11         Update Student model, $v$, and $f_*$ by minimizing $\mathcal{L}$
```

A.3 Datasets.

We use two public real-world datasets: CiteULike and Foursquare. We remove users having fewer than 5 (CiteULike) and 20 interactions (FourSquare) and remove items having fewer than 10 interactions (FourSquare) as done in [8, 11]. Table 5 summarizes the statistics of the datasets. In the case of CiteULike, each item corresponds to an article, and each article has multiple tags. In the case of Foursquare, each item corresponds to a POI (points-of-interest) such as museums and restaurants, and each POI has GPS coordinates (i.e., the latitude and longitude). We use this side information in Section 4.3. Table 6 shows the URLs from which the datasets can be downloaded.

Table 5: Statistics of the datasets.

Dataset	#Users	#Items	#Interactions	Density
CiteULike	5,220	25,182	115,142	0.09%
Foursquare	19,466	28,594	609,655	0.11%

Table 6: URL links to the datasets.

Dataset	URL link to the dataset
CiteULike	https://github.com/changun/CollMetric
Foursquare	http://spatialkeyword.sce.ntu.edu.sg/eval-vldb17/

A.4 Evaluation Protocol and Metrics

We adopt the widely used *leave-one-out* evaluation protocol, whereby two interacted items for each user are held out for testing/validation, and the rest are used for training. However, unlike [8] that samples a predefined number (e.g., 499) of unobserved items for evaluation, we adopt the full-ranking evaluation scheme that evaluates how well each method can rank the test item higher than all the unobserved items. Although it is time-consuming, it enables a more thorough evaluation compared to the sampling-based evaluation [9, 12]. We evaluate all methods by two widely used ranking metrics: Recall@N [14] and Normalized Discounted Cumulative Gain (NDCG@N) [7]. Recall@N measures whether the test item is included in the top-N list and NDCG@N assigns higher scores on the upper ranked test items. We compute the metrics for each user, then compute the average score. Lastly, we report the average value of five independent runs for all methods.

A.5 Implementation Details.

We use PyTorch to implement the proposed methods and all the competing methods. We optimize all methods with Adam optimizer. For DE and RRD, we use the public implementation provided by the authors. For each setting, hyperparameters are tuned by using grid searches on the validation set. The learning rate is searched in the range of {0.01, 0.005, 0.001, 0.0005, 0.0001}. The model regularizer is searched in the range of $\{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$. We set the total number of epochs to 500 and adopt the early stopping strategy; it terminates when Recall@50 on the validation set does not increase for 20 successive epochs.

For all base models, the number of negative samples is set to 1. For NeuMF and LightGCN, the number of layers is searched in the range of {1, 2, 3, 4}. For all the distillation methods, weight for the distillation loss (λ) searched in the range of $\{1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$.

For the hint regression-related setup, we closely follow the setup reported in DE paper [8]. Specifically, two-layer MLP with $[d^s \rightarrow (d^s + d^t)/2 \rightarrow d^t]$ is employed for f in FitNet, DE and HTD. Also, one-layer perceptron with $[d^t \rightarrow K]$ is employed for assigning group (v) in DE and HTD. For DE and HTD, the number of preference groups (K) is chosen from {5, 10, 20, 30, 40, 50}. We provide an analysis of these hyperparameters in Section 4.4.

A.6 Experiment Setup for Downstream Tasks.

We evaluate how well each method encodes the items' characteristics (or semantics) into the representations. We train a small network to predict the side information of items by using the *fixed* item representations as the input. Specifically, we use a linear and a non-linear model (i.e., a single-layer perceptron and three-layer perceptron, respectively) with Softmax output. The linear model has the shape of $[d^s \rightarrow C]$, and the non-linear model has the shape of $[d^s \rightarrow (d^s + C)/2 \rightarrow (d^s + C)/2 \rightarrow C]$ with relu, where C is the number of tags/classes. Let $\mathbf{q}$ denote the output of the model whose element q_i is a prediction score for each tag/class. Also, let $\mathbf{p}$ denote the ground-truth vector whose element $p_i = 1$ if i-th tag/class is the answer, otherwise $p_i = 0$. We train the model by minimizing the negative log-likelihood: $-\sum_i p_i \log q_i$. Note that the side-information is not utilized for training of the base model.

For CiteULike dataset, we perform **item-tag retrieval task**; by using each item representation as a query, we find a ranking list of tags that are relevant to the item. We first remove tags used less than 10 times. Then, there exist 4,153 tags and an item has 6.4 tags on average. After training, we make a ranking list of tags by sorting the prediction scores. We evaluate how many the ground-truth tags are included in the top-10 list by Recall@10. For Foursquare dataset, we perform **item-region classification task**; given each item representation, we predict the region class to which the item belongs. We first perform k-means clustering on the coordinates with $k = 200$ and use the clustering results as the class labels. After training, we evaluate the performance by Accuracy. Finally, we perform 5-fold cross-validation and report the average result and standard deviation in Table 3.