TPR: Text-aware Preference Ranking for Recommender Systems

Yu-Neng Chuang*
National Chengchi University
107753011@nccu.edu.tw

Chih-Ming Chen*†
National Chengchi University
104761501@nccu.edu.tw

Chuan-Ju Wang
Academia Sinica
cjwang@citi.sinica.edu.tw

Ming-Feng Tsai
National Chengchi University
mftsai@nccu.edu.tw

Yuan Fang
Singapore Management University
yfang@smu.edu.sg

Ee-Peng Lim
Singapore Management University
eplim@smu.edu.sg

ABSTRACT

Textual data is common and informative auxiliary information for recommender systems. Most prior art utilizes text for rating prediction, but rare work connects it to top-*N* recommendation. Moreover, although advanced recommendation models capable of incorporating auxiliary information have been developed, none of these are specifically designed to model textual information, yielding a limited usage scenario for typical user-to-item recommendation. In this work, we present a framework of text-aware preference ranking (TPR) for top-*N* recommendation, in which we comprehensively model the joint association of user-item interaction and relations between items and associated text. Using the TPR framework, we construct a joint likelihood function that explicitly describes two ranking structures: 1) item preference ranking (IPR) and 2) word relatedness ranking (WRR), where the former captures the item preference of each user and the latter captures the word relatedness of each item. As these two explicit structures are by nature mutually dependent, we propose TPR-OPT, a simple yet effective learning criterion that additionally includes implicit structures, such as relatedness between items and relatedness between words for each user for model optimization. Such a design not only successfully describes the joint association among users, words, and text comprehensively but also naturally yields powerful representations that are suitable for a range of recommendation tasks, including user-to-item, item-to-item, and user-to-word recommendation, as well as item-to-word reconstruction. In this paper, extensive experiments have been conducted on eight recommendation datasets, the results of which demonstrate that by including textual information from item descriptions, the proposed TPR model consistently outperforms state-of-the-art baselines on various recommendation tasks.

CCS CONCEPTS

• **Information systems** → *Collaborative filtering*; • **Computing methodologies** → *Learning latent representations*.

KEYWORDS

recommender systems; textual information; preference ranking

ACM Reference Format:
Yu-Neng Chuang, Chih-Ming Chen, Chuan-Ju Wang, Ming-Feng Tsai, Yuan Fang, and Ee-Peng Lim. 2020. TPR: Text-aware Preference Ranking for Recommender Systems. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3340531.3411969

1 INTRODUCTION

Recommender systems are ubiquitous, as almost every service that provides content to users is now armed with a recommender system. In general, as user-item interactions such as ratings, playing times, likes, sharing, and tags are generally available in real-world recommender systems, many systems have leveraged collaborative filtering (CF) for recommendation [9]. Despite the effectiveness and prevalence of CF, most pure CF methods (i.e., those that leverage only user-item interaction for modeling) do not incorporate auxiliary information such as item description and user profiles, thereby yielding poor performance when user-item interactions are sparse as well as under cold-start situations.

To leverage such auxiliary information to boost performance, modern recommendation algorithms have expanded their ability to integrate auxiliary context information using CF [3, 19, 22, 26]. A natural paradigm is to transform the auxiliary information into a generic feature vector, along with user and item IDs, using this as the input to train a supervised model for score prediction. Such a paradigm for recommender systems has been widely used in industry [6, 21]; representative algorithms include factorization machines (FM) [19], NFM (neural FM) [7], and Wide & Deep [5]. On the other hand, as graphs are an extremely flexible and powerful way to represent data, another paradigm is to construct a graph structure that incorporates both the auxiliary information and user-item interactions, based on which node and/or edge (or relation) embeddings are learned [1, 3, 16, 22, 25–27]; for this type of approach, recent recommendation models such as CKE [26] exploit knowledge base for better recommendation results; KGAT [22] investigate the utility of knowledge graphs (KGs) and yield state-of-the-art

* These authors contributed equally to this work.
† Social Networks and Human-Centered Computing, Taiwan International Graduate Program, Institute of Information Science, Academia Sinica, Taiwan.

CIKM '20, October 19–23, 2020, Virtual Event, Ireland

ACM ISBN 978-1-4503-6859-9/20/10…$15.00
https://doi.org/10.1145/3340531.3411969

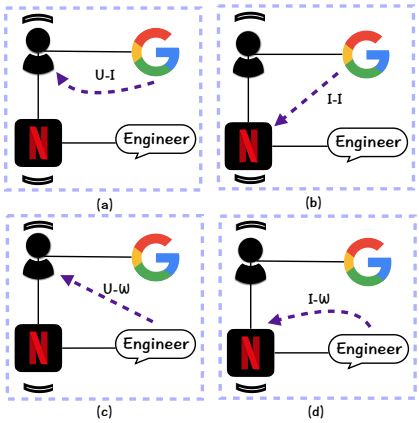

Figure 1: Four relations in TPR for job recommendation

performance on recommendation. However, as the above methods are not particularly designed to model textual information, their usage scenarios are limited to typical user-to-item recommendation only. For example, models within the supervised paradigm predict recommendation scores for a given user and an item; even though such methods can encode textual information via a feature vector (e.g., a bag-of-words representation), it is difficult to complete other recommendation tasks, including item-to-item recommendation or user-to-word recommendation.

An exception that deals solely with textual information is a thread of studies that incorporates review texts within recommendation models; instead of the commonly adopted top-N recommendation, this thread of work considers rating-based prediction only [2, 12, 13]. In contrast to prior studies, in this paper, we focus on modeling another source of textual information—the item description. Compared to user reviews, the text in item descriptions is usually easier to obtain and conveys specific and accurate information regarding the corresponding items, thereby constituting informative material for modeling user preference. It is of great help in providing explainable recommendations.

Thus, in this paper, we propose text-aware preference ranking (TPR), a framework in which we model the joint association of user-item interaction and relations between items and associated text for top-N recommendation. Motivated by preference ranking in ranking-based CF [4, 20], we construct a joint likelihood function that explicitly describes two explicit ranking structures: 1) item preference ranking (IPR) and 2) word relatedness ranking (WRR), where IPR captures the item preference of each user and WRR captures the word relatedness of each item. As these two structures are by nature mutually dependent, except for modeling them, we propose TPR-OPT, a simple yet effective learning criterion that additionally includes implicit structures such as relatedness between items and relatedness between words for a user; four types of structures are shown in Figure 1 under the scenario of job recommendation. Note that our design not only successfully describes the joint association comprehensively but also naturally yields universal and powerful representations for users, items, and texts that are suitable for a range of recommendation tasks, including user-to-item, item-to-item, and user-to-word recommendation, as well as item-to-word reconstruction. The support of such variants of recommendation tasks in a unified model indeed broadens the usage scope of our model. For example, within a job recommendation scenario, user-to-word recommendation can be of great help in providing users with related skill sets mentioned in job descriptions, which can be done naturally as the TPR-OPT simultaneously models user preference on text.

We conduct extensive experiments on eight recommendation datasets, including six publicly available Amazon datasets and two privately collected datasets. To attest the capability of the learned embeddings, we perform various types of tasks, including typical user-to-item (cold-start) recommendation, item-to-word reconstruction, and user-to-word recommendation. Experimental results show that by including textual information from item descriptions, the proposed model consistently outperforms state-of-the-art baselines on various tasks. We also discuss the efficiency of the proposed method in terms of time and memory usage empirically. In summary, the contributions of this work are listed as follows.

- We present TPR, a text-aware recommendation framework that jointly describes the association of user-item interactions and relations between items and associated text.
- Within the TPR framework, we propose TPR-OPT, an effective learning criterion that comprehensively models four types of structures among users, items, and texts.
- By optimizing TPR-OPT, the learned embeddings of users, items, and words are comparable; thus, any pair-wise similarity (e.g., user-to-item or user-to-word) can be obtained in a straightforward manner to support various types of recommendation tasks.
- We conduct extensive experiments on eight datasets, showing the superiority of the proposed method over different advanced models on various recommendation tasks.
- We provide an effective and efficient implementation, the source code is available online at a GitHub repository [1].

2 METHODOLOGY

In this section, we first introduce the problem definition of text-aware recommendation in Section 2.1 and give an overview of our proposed TPR framework that leverages both explicit and implicit ranking structures in Section 2.2. Subsequently, in Section 2.3 we introduce a joint learning objective for both explicit and implicit structures and then discuss its optimization together with regularization.

2.1 Problem Definition

Let U, I, and W denote the sets of users, items, and words, respectively. In this study we consider two types of relations between the elements in these sets: 1) interaction between users and items, denoted as $E_{u,i} = \{(u,i)|u \in U, i \in I\}$, and 2) the "has-a" relation between items and words, denoted as $E_{i,w} = \{(i,w)|i \in I, w \in W\}$, where the words for each item are extracted from its description.

The goal of our model is to learn a representation matrix $\Theta \in \mathbb{R}^{|U \cup I \cup W| \times d}$ that maps each user, item, or word to a d-dimensional embedding vector. The learned embedding vectors are thus suitable for various types of recommendation tasks, including user-to-item, item-to-item, and user-to-word recommendation, as well as item-to-word reconstruction, as motivated in Section 1.

[1] https://github.com/cnclabs/codes.tpr.rec

2.2 Proposed TPR Framework

The proposed TPR is designed to model the joint association of user-item interactions and the relations between items and associated words from their descriptions. Here, we use preference ranking [4, 20, 23] to describe such relations, for which the objective is to find an embedding matrix Θ that maximizes the joint likelihood function from observed user-item and item-word pairs:

$$O_{\text{TPR}} \equiv \max \prod_{(u,i)\in E_{u,i}} p(\overbrace{>_u}^{\text{IPR}}, \overbrace{>_i}^{\text{WRR}} |\Theta), \tag{1}$$

where $>_u$ indicates the preference structure between two items for the given user $u \in U$, $>_i$ refers to the relatedness structure between words for the given item $i \in I$. Specifically, $j >_u j'$ denotes that user u prefers item j over item j', whereas $w >_i w'$ denotes that word w is with a higher probability of being in the description of item i than w'. From Eq. (1), the joint likelihood is composed of two ranking structures: 1) $>_u$, item preference ranking (IPR) (described in Section 2.2.1) and 2) $>_i$, word relatedness ranking (WRR) (described in Section 2.2.2). Note that structures $>_u$ and $>_i$ are by nature mutually dependent as the items that the given user has interacted with overlap in these two structures. Therefore, despite only explicitly observing these two structures, structures such as relatedness between items and relatedness between words for a user should also be considered in the joint likelihood function.

To realize such preference (relatedness) ranking, we create a set of triples $D_u : U \times I \times I$ based on user-item interactions $E_{u,i}$ for $>_u$, and another set of triples $D_i : I \times W \times W$ based on the has-a relations between items and words $E_{i,w}$ for $>_i$, as $D_u = \{(u, j, j') | \forall u \in U, j \in I_u^+ \wedge j' \in I \backslash I_u^+\}$ and $D_i = \{(i, w, w') | \forall i \in I, w \in W_i^+ \wedge w' \in W \backslash W_i^+\}$ respectively, where $I_u^+ = \{i \in I | (u, i) \in E_{u,i}\}$ denotes the set of items that user u has interacted with, and $W_i^+ = \{w \in W | (i, w) \in E_{i,w}\}$ denotes the set of words in the description of item i. That is, for each triple $(u, j, j') \in D_u$, we have $j >_u j'$. Similarly, for each triple $(i, w, w') \in D_i$, we have $w >_i w'$.

Figure 2 illustrates the overall concept of the proposed TPR framework. Given a user-item interaction $(u, i) \in E_{u,i}$, we consider a joint likelihood function composed of one preference structure on items for u (i.e., IPR) and one relatedness structure on words for i (i.e., WRR), as shown in the dashed box. Due to the dependency between the explicit structures (a) and (d) shown in the figure, two additional structures should be further modeled: (b) for the similarity ranking of two other items w.r.t. the given item, and (c) for the word relatedness ranking of two words w.r.t. the given user. Below, in Sections 2.2.1 and 2.2.2, we discuss the two explicitly considered structures, IPR and WRR, respectively, after which we discuss how to design the objective to jointly consider the four types of structures in Section 2.3.

2.2.1 Item Preference Ranking (IPR). With item preference ranking (IPR), we focus on generating a personalized ranked list of items for each user based on the observed user-item interaction by leveraging the user-item-item triples D_u as training data to optimize the correct ranking of item pairs [4, 20, 23]. For such an approach, BPR [20] is a pioneering, well-known example in which the likelihood function of IPR is formulated as

$$\begin{aligned} O_{\text{IPR}} &= \max \prod_{u\in U} p(>_u |\Theta) \\ &\propto \max \ln \left(\prod_{u\in U} p(>_u |\Theta) \right) \\ &= \max \ln \left(\prod_{(u,j,j')\in D_u} p(j >_u j' |\Theta) \right) \\ &= \max \sum_{(u,j,j')\in D_u} \ln p\left(j >_u j' |\Theta\right). \end{aligned} \tag{2}$$

Recall that I_u^+ is the item preference set of the given user u, and $>_u$ denotes the pairwise item preference for user u. It is common to calculate $p(j >_u j' |\Theta)$ in Eq. (2) as

$$p(j >_u j' |\Theta) = \sigma\left(\left\langle \Theta_u, \Theta_j - \Theta_{j'} \right\rangle\right), \tag{3}$$

where $\langle \cdot, \cdot \rangle$ denotes the dot product between two vectors, Θ_u (Θ_j) denotes the d-dimensional row vector from Θ for user $u \in U$ (item $j \in I$, respectively) and $\sigma(\cdot)$ denotes the sigmoid function.

2.2.2 Word Relatedness Ranking (WRR). With the WRR component, we seek to model the relation between items and associated words from their description. Inspired by language modeling techniques that learn the word semantics by the words distributions [10, 14, 15], we propose maximizing the likelihood function of relevant (positive) item-word pairs over irrelevant (negative) item-word pairs for each item, by leveraging the item-word-word triples in D_i as training data. Thus we have

$$\begin{aligned} O_{\text{WRR}} &= \max \prod_{i\in I} p(>_i |\Theta) \\ &\propto \max \ln \left(\prod_{i\in I} p(>_i |\Theta) \right) \\ &= \max \sum_{(i,w,w')\in D_i} \ln p(w >_i w' |\Theta). \end{aligned} \tag{4}$$

Recall that $>_i$ denotes the pairwise word relatedness for item i (i.e., $w >_i w'$ indicates that word w is more related than word w' to item i). In Eq. (4), $p(w >_i w' |i)$ is calculated as

$$p(w >_i w' |i) = \sigma\left(\langle \Theta_i, \Theta_w - \Theta_{w'} \rangle\right), \tag{5}$$

where Θ_w is the d-dimensional row vector from Θ for word $w \in W$.

2.3 Joint Learning of Implicit and Explicit Ranking Structures

Next, we illustrate how to design an objective function that jointly considers the four types of explicit and implicit structures shown in Figure 2(a)–(d). To our best knowledge, although various systems have been developed that incorporate IPR concepts into their recommendation models, none leverages the concept of WRR to fuse textual information into the models. More importantly, due to the dependency between IPR and WRR, it is unreasonable to model these two structures independently, that is, to directly maximize the product of Eqs. (3) and (5).

To this end, in this paper, we propose a learning approach based on the TPR framework for text-aware recommendation, for which we

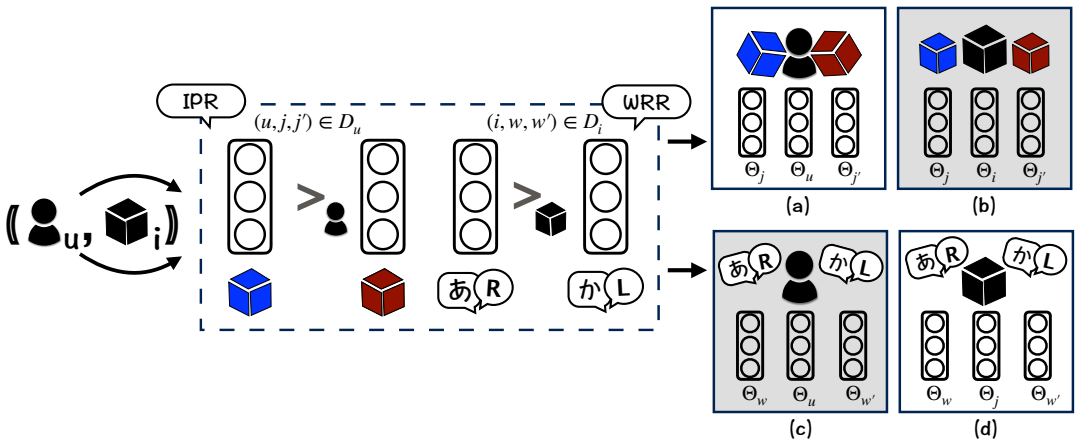

Figure 2: Overview of proposed TPR framework

define the calculation for the joint likelihood function (see Eq. (1)) for a user-item pair $(u,i) \in E_{u,i}$ as

$$\begin{aligned} & p(j >_u j', w >_i w'|\Theta) \\ &= \sigma\left(\langle \Theta_u + \Theta_i, (\Theta_j - \Theta_{j'}) + (\Theta_w - \Theta_{w'})\rangle\right) \\ &= \sigma\Big(\langle \Theta_u, (\Theta_j - \Theta_{j'})\rangle + \langle \Theta_u, (\Theta_w - \Theta_{w'})\rangle + \\ &\quad \langle \Theta_i, (\Theta_j - \Theta_{j'})\rangle + \langle \Theta_i, (\Theta_w - \Theta_{w'})\rangle\Big), \end{aligned} \tag{6}$$

where $j \in I_u^+$, $j' \in I \backslash I_u^+$, $w \in W_i^+$, and $w' \in W \backslash W_i^+$. As shown, the above joint likelihood in Eq. (6) can be decomposed into the following four components:

(a) $\langle \Theta_u, \Theta_j - \Theta_{j'}\rangle$: Modeling the item preference ranking (IPR) between j and j' for user u (see Figure 2(a) and Eq. (3));
(b) $\langle \Theta_i, \Theta_j - \Theta_{j'}\rangle$: Modeling the item similarity to item i regarding items j and j' (see Figure 2(b));
(c) $\langle \Theta_u, \Theta_w - \Theta_{w'}\rangle$: Modeling the word relatedness ranking to user u regarding words w and w' (see Figure 2(c));
(d) $\langle \Theta_i, \Theta_w - \Theta_{w'}\rangle$: Modeling the word relatedness ranking to item i regarding words w and w' (see Figure 2(d) and Eq. (5)).

Note that the item i and j in (b) are both positive items for user u (i.e., $i, j \in I^+(u)$); as a result, for this part, the model tends to cluster items that the user has interacted with in the training data in the embedding space. Moreover, word w in (c) is extracted from the description of an item i that the user has interacted with (i.e., $i \in I_u^+$); thus, this part of modeling captures user word preferences. It is worth remembering that this elegant design for the objective not only successfully describes the joint likelihood function in Eq. (1) comprehensively but also naturally yields a powerful representation matrix Θ that is suitable for a range of recommendation tasks, including user-to-item, item-to-item, and user-to-word recommendation, as well as item-to-word reconstruction.

With Eq. (6), we formulate the maximum posterior estimator to derive our optimization criterion for TPR as

$$\begin{aligned} & \text{TPR-OPT} \\ &:= \ln p(\Theta| >_u, >_i) \ \propto \ \ln p(>_u, >_i |\Theta) p(\Theta) \\ &= \ln \prod_{\substack{(u,j,j')\in D_u, \\ (i,w,w')\in D_i}} p(j >_u j', w >_i w'|\Theta) p(\Theta) \\ &= \sum_{\substack{(u,j,j')\in D_u, \\ (i,w,w')\in D_i}} \ln \sigma\left(\langle \Theta_u + \Theta_i, (\Theta_j - \Theta_{j'}) + (\Theta_w - \Theta_{w'})\rangle\right) - \lambda_\Theta \|\Theta\|^2, \end{aligned} \tag{7}$$

where λ_Θ is a model-specific regularization parameter.

2.3.1 Optimization. The most common algorithms for gradient ascent are full and stochastic gradient ascent. In the first case, in each step the full gradient over all training data is calculated after which the model parameters are updated according to learning rate α:

$$\Theta \leftarrow \Theta + \alpha \left(\frac{\partial \text{TPR-OPT}}{\partial \Theta}\right).$$

Although full gradient ascent generally leads to an ascent in the correct direction, its convergence is slow. As a result, in this paper, the objective function in Eq. (7) is instead maximized by adopting asynchronous stochastic gradient ascent—the opposite of asynchronous stochastic gradient descent (ASGD) [18]—to efficiently update parameters Θ in parallel. Specifically, for each given (u,i) pair, we randomly sample one positive item as j and one negative item as j' for user u and a positive-negative word pair as (w, w') for item i, resulting in triplets $(u,j,j') \in D_u$ and $(i,w,w') \in D_i$ for updating the parameters with the gradient defined as

$$\frac{\partial \text{TPR-OPT}}{\partial \Theta} = \frac{\partial}{\partial \Theta} \ln \sigma(\hat{x}) - \lambda_\Theta \frac{\partial}{\partial \Theta} \|\Theta\|^2 \tag{8}$$

$$\propto \frac{e^{-\hat{x}}}{1+e^{-\hat{x}}} \frac{\partial}{\partial \Theta} \hat{x} - \lambda_\Theta \Theta, \tag{9}$$

where $\hat{x} := \langle \Theta_u + \Theta_i, (\Theta_j - \Theta_{j'}) + (\Theta_w - \Theta_{w'})\rangle$.

2.3.2 Regularization. In practice, the regularization term is used to reduce model complexity and generally benefits model generalization ability. As the relations in IPR and WRR structures are essentially different, we enable TPR-OPT to have different weights for regularization. By rewriting Eq. (9), each row vector Θ_k in Θ is updated by

$$\frac{\partial \text{TPR-OPT}}{\partial \Theta_k} = \frac{e^{-\hat{x}}}{1+e^{-\hat{x}}} \frac{\partial}{\partial \Theta_k} \hat{x} - \Lambda(k)\Theta_k$$

where

$$\Lambda(k) := \begin{cases} \lambda_{\text{IPR}} & \text{if } k \text{ is an element of tuple } (u, j, j') \in D_u, \\ \lambda_{\text{WRR}} & \text{if } k \text{ is an element of tuple } (i, w, w') \in D_i. \end{cases}$$

Above, the two hyperparameters for regularization—λ_{IPR} and λ_{WRR}—allow additional flexibility in adjusting the relation modeling for different types of tasks. The effect of adopting different values of λ_{IPR} and λ_{WRR} is discussed in Section 3.

3 EXPERIMENT

3.1 Datasets

To examine the performance of the proposed model, we conducted experiments on eight datasets, including six public benchmarks and two private real-world datasets, the statistics of which are listed in Table 1. The six public benchmarks are from the Amazon review dataset [17],[2] for which we adopted user-item ratings and item descriptions in the experiments. Note that for the Amazon data, we treated items with ratings as positive feedback and the rest as negative feedback, and removed users that had rated fewer than three items. For the two private datasets, the first is the Online Professional Network data in Singapore, called SG-OPN, with about 10,000 persons with their working experience and 12,000 job postings, for which we treat each person and his/her listed jobs as the user-item interactions and the job descriptions as the item descriptions; the second one is the course-taking data from a major university in Asia, called the Course dataset, with 17,000 students with their course-taking logs and 600 courses with descriptions, for which we treat each student and his/her taken courses as the user-item interactions and the course descriptions as the item descriptions. Thus, all of our experimental datasets contain user-item interactions and textual item descriptions. For preprocessing, we converted the user-item interactions into implicit feedback, and for item descriptions, we filtered out words with term frequencies of less than five or words with a document frequency of less than ten percent of the corresponding corpus. Table 1 lists the statistics of the preprocessed data for each dataset, including the number of words, the amount of implicit feedback (U-I edges), and the number of relations between items and words (I-W edges).

3.2 Baselines

We compared our model with seven baseline methods. The first two, BPR [20] and WARP [23], leverage user-item interactions only for recommendation; the third and fourth, SINE [25] and HPE [3], are graph embedding methods; the fifth, GATE [13], is a model incorporating reviews for recommendation; the last two, CKE [26] and KGAT [22], are state-of-the-art knowledge-based recommendation methods. Below we briefly describe each method and how we adopted these methods under our settings.

- **BPR** [20] (Bayesian Personalized Ranking) adopts pairwise ranking loss for personalized recommendation and exploits direct user-item interaction to separate negative items from positive items.
- **WARP** [23] (Weighted Approximate-Rank Pairwise) improves ranking-based models based on BPR, weighing pairwise violations depending on their positions in a ranked list.
- **SINE** [25] (Scalable Incomplete Network Embedding) is an attributed network embedding algorithm, which incorporates words as item attributes.
- **HPE** [3] (Heterogeneous Preference Embedding) encodes user preferences and query intentions into low-dimensional vector spaces.
- **GATE** [13] (Gated Attentive-autoencoder) is an end-to-end recommendation algorithms that fuses hidden representations of item contents and binary ratings using a neural gating structure.
- **CKE** [26] (Collaborative Knowledge Base Embedding) is a knowledge-based recommendation method which exploits semantic knowledge derived from TransR [11] to enhance the performance of matrix factorization.
- **KGAT** [22] (Knowledge Graph Attention Network) is a recommendation model that explicitly models high-order relations in a collaborative knowledge graph under a graph neural network framework.

Note that GATE can be used for typical user-item recommendation only as it is an end-to-end recommendation model. Other than BPR and WARP, all other baselines encode textual information; however, only HPE, CKE, and KGAT can be used for tasks that involve node embeddings for words, e.g., cold-start recommendation, item-to-word reconstruction, and user-to-word recommendation. The implementations for the seven baselines are listed in the footnote.[3]

	Users	Items	Words	U-I edges	I-W edges
Amazon-Magazine	2,825	1,299	6,740	11,685	9,4381
Amazon-Beauty	4,801	4,865	4,115	11,685	159,475
Amazon-Application	11,823	5,554	9,712	42,675	410,079
Amazon-Software	13,634	9,325	11,111	57,793	766,112
Amazon-Fashion	19,875	36,080	5,076	75,596	442,136
Amazon-Kindle	363,303	356,634	36,445	3,334,521	6,794,209
Course	~17,000	~600	~3,000	~300,000	~150,000
SG-OPN	~10,000	~12,000	~10,100	~30,000	~1,100,000

Table 1: Dataset statistics

3.3 Experimental Settings

To attest the versatility of the learned embeddings of users, items, and words, we completed the following recommendation tasks in the following experiments.

[2] https://nijianmo.github.io/amazon/index.html

[3] WARP: https://github.com/lyst/lightfm; HPE: https://github.com/cnclabs/smore; SINE: https://github.com/benedekrozemberczki/karateclub; BPR, CKE, KGAT: https://github.com/xiangwang1223/knowledge_graph_attention_network; GATE: https://github.com/allenjack/GATE

(1) Recommendation tasks:
   (a) **User-to-item recommendation**: In this task, we provide a list of recommended items for each user; for each user u the items are ranked by the score $\langle \Theta_u, \Theta_j \rangle$, where $j \in I$.
   (b) **Item-to-item recommendation**: This task is the same as the previous task, except that the items are ranked by the score $\sum_{i \in I_u^+} \langle \Theta_i, \Theta_j \rangle$, where $j \in I$. Note that for this task, instead of using the user embedding to calculate the score, we calculate it using the embeddings of items that the user has interacted with in the training data.
(2) Cold-start recommendation tasks:
   (a) **User-to-item recommendation**: In this task, we recommend items that are completely new, which means that the recommended items are not included in the training data. In this scenario, we generate the embedding of a new item j by $\Theta_j = \frac{1}{|W_j^+|} \sum_{w \in W_j^+} \Theta_w$. For each user u, the recommendations are ranked by $\langle \Theta_u, \Theta_j \rangle$.
   (b) **Item-to-item recommendation**: This task is same as the above cold-start setting, but the items are ranked by the score $\sum_{i \in I_u^+} \langle \Theta_i, \Theta_j \rangle$.
(3) Word-related tasks:
   (a) **Item-to-word reconstruction**: With this task, we evaluate the text awareness of the proposed model. Given an item, this task is to restore the contained relations between the given item and the words from its description. The score of an item-word pair (j, w) is computed by $\langle \Theta_j, \Theta_w \rangle$.
   (b) **User-to-word recommendation**: In this task, we are to predict a list of words for each user; note that the recommended words are new, meaning that these words do not appear in the descriptions of items that the user has interacted with in the training data. The recommendations are then ranked by $\langle \Theta_u, \Theta_w \rangle$, where $w \in W \backslash W_i^+$ for all $i \in I_u^+$.

For all of these tasks, we focus on the performance of top-N recommendation, using two common recommendation metrics for evaluation: recall (denoted as Recall@N) and normalized discount cumulative gain (denoted as NDCG@N). For all the datasets, we randomly divided the user-item interactions into 80% and 20% as the training set and the testing set, respectively. The recommendation pool for each user was generated as the collection of positive items with 1,000 randomly selected negative items. Similar settings can be found in [8, 24]. For the cold-start recommendation tasks, the recommendation pool contained only those positive items that were not in the training set. The final reported results were calculated by averaging the results over five repetitions. We used all words in the item descriptions; each unique word w corresponds an embedding Θ_w (i.e., a unigram model). In our experiments, the dimensions of the embedding vectors were set to 128, and all the hyper-parameters of the compared models were determined via a grid search over different settings, from which the combination that lead to the best performance was chosen. The training iteration we searched for the compared methods was {50, 100, 200}. For the proposed TPR, to clearly illustrate the function of combined IPR/WRR regularization, we fixed $\lambda_{\text{IPR}} = 0.025$ and report the performance by varying λ_{WRR} in the experiments.

3.4 Experimental Results

In the following sections, we demonstrate the results of the tasks listed in Section 3.3. First, we conduct experiments on typical top-N recommendation, the results of which are shown in Sections 3.4.1 and 3.4.2. Second, typical top-N recommendation for the cold-star scenarios is considered in Sections 3.4.3 and 3.4.4. Finally, we examine whether a model successfully encodes the word knowledge into its learned representations by performing item-word reconstruction and user-to-word recommendation in Sections 3.4.5 and 3.4.6, respectively. Note that in the reported results , "†" symbol in Tables 2-7 indicates the best performing method among all the baseline methods; "*" and "Improv. (%)" denote statistical significance at p-value < 0.01 with a paired t-test and the percentage improvement of the proposed model, respectively, with respect to the best performing value in the baselines.

3.4.1 User-to-item Recommendation (Task 1-a). This is the typical recommendation task evaluated in most of the literature in recommender systems. Table 2 tabulates the results in terms of Recall@10 and NDCG@10 of the proposed TPR and the seven baseline methods, where the best results are highlighted in bold. Note that for the Amazon-Kindle dataset, we do not report results of GATE, CKE, and KGAT due to computational resource limitations.[4] Below, we itemize the findings from Table 2.

- In general, the models incorporating the textual information outperform the ones leveraging solely user-item interactions (i.e., BPR and WARP). However, as SINE and GATE are not originally proposed to handle the user-item ranking problem, these two methods obtain relative weak results even though they include the text data into their models.
- HPE, CKE, and KGAT are considered as the most competitive baselines in this task due to their top performance. Even so, the proposed TPR generally gains significantly better results than the three methods for seven datasets in terms of both Recall@10 and NDCG@10, except for the smallest dataset, Amazon-Magazine. The results suggest that our method could perform better in larger datasets with more interactions and textual information.
- It is feasible to train the proposed model on very large-scale datasets, such as the Amazon-Kindle dataset; many other advanced models however encounter either the training efficiency or memory capacity issues. Comparison on the training time and memory usage among different models are reported and disucssed in Section 3.5.2.
- For such a user-to-item recommendation task, adopting a larger regularization on the WRR structure, λ_{WRR}, benefits model generalization ability regarding word relatedness to items, more analysis for which is described in Section 3.5.1.
- In sum, the overall improvement of the proposed TPR ranges from 1.00% to 18.56% in terms of Recall@10 and from 3.70% to 24.75% in terms of NDCG@10; such improvements should be considered as measurable ones.

3.4.2 Item-to-item Recommendation (Task 1-b). For this and the following tasks in Sections 3.4.3-3.4.6, we only report results

[4]The model is unable to be loaded into the GPU with 32GB memory or the training is unable to be finished within days.

	Amazon-Magazine		Amazon-Beauty		Amazon-Applications		Amazon-Fashion	
	Recall@10	NDCG@10	Recall@10	NDCG@10	Recall@10	NDCG@10	Recall@10	NDCG@10
BPR [20]	0.3306	0.1734	0.4278	0.3468	0.3035	0.1590	0.1563	0.1223
WARP [23]	0.3435	0.1892	0.3468	0.3437	0.3016	0.1655	0.1815	0.1298
SINE [25]	0.0360	0.0083	0.0549	0.0157	0.1283	0.0280	0.0865	0.0181
HPE [3]	0.3419	0.1377	† 0.4773	† 0.3652	† 0.3552	0.1736	† 0.2126	† 0.1393
GATE [13]	0.2720	0.0489	0.3940	0.0812	0.1336	0.0225	0.0819	0.0186
CKE [26]	0.3838	0.2061	0.4208	0.3450	0.2933	0.1562	0.1581	0.1230
KGAT [22]	† **0.4156**	† **0.2156**	0.4321	0.3558	0.3213	† 0.1862	0.1862	0.1268
TPR ($\lambda_{\text{WRR}} = 0.001$)	0.3681	0.1599	***0.4950**	*0.3735	*0.3937	*0.1779	*0.2394	*0.1525
TPR ($\lambda_{\text{WRR}} = 0.005$)	0.4101	0.1880	*0.4925	*0.3783	***0.4097**	*0.1951	***0.2270**	*0.1462
TPR ($\lambda_{\text{WRR}} = 0.01$)	0.4182	0.1840	*0.4840	***0.3793**	*0.3997	***0.1971**	*0.2258	***0.1482**
Improv. (%)	−0.62%	−12.80%	+3.70%	+3.86%	+15.34%	+5.85%	+6.77%	+6.38%

	Amazon-Software		Amazon-Kindle		Course		SG-OPN	
	Recall@10	NDCG@10	Recall@10	NDCG@10	Recall@10	NDCG@10	Recall@10	NDCG@10
BPR [20]	† 0.3669	0.1779	0.4414	0.2097	0.5731	0.4129	0.1008	0.0339
WARP [23]	0.3423	0.1556	† 0.5461	† 0.3392	0.5340	0.3639	† 0.2623	† 0.1131
SINE [25]	0.0976	0.0257	0.2812	0.1394	0.0357	0.0168	0.0412	0.0150
HPE [3]	0.3658	0.1405	0.5228	0.2803	0.3391	0.2294	0.0047	0.0040
GATE [13]	0.1326	0.0202	-	-	0.4477	0.3170	0.0010	0.0035
CKE [26]	0.3448	0.1497	-	-	† 0.6094	† **0.4583**	0.1050	0.0837
KGAT [22]	0.3907	† 0.1847	-	-	0.5902	0.4294	0.1473	0.0512
TPR ($\lambda_{\text{WRR}} = 0.001$)	*0.3898	0.1615	*0.5682	*0.3448	0.5735	0.4177	***0.3110**	***0.1411**
TPR ($\lambda_{\text{WRR}} = 0.005$)	*0.4252	0.1844	*0.6065	*0.3722	0.6014	0.4422	*0.3094	*0.1392
TPR ($\lambda_{\text{WRR}} = 0.01$)	***0.4319**	***0.1956**	***0.6164**	***0.3804**	***0.6155**	0.4468	*0.3086	*0.1434
Improv. (%)	+17.71%	+5.90%	+12.87%	+12.14%	+1.00%	−2.50%	+18.56%	+24.75%

Table 2: Performance on user-to-item recommendation

	Amazon-Magazine		Amazon-Beauty		Amazon-Applications		Amazon-Fashion		Amazon-Software	
	Recall@10	NDCG@10	Recall@10	NDCG@10	Recall@10	NDCG@10	Recall@10	NDCG@10	Recall@10	NDCG@10
BPR [20]	0.3637	0.1964	0.4433	0.3689	0.3472	† 0.1871	0.1655	0.1244	0.3993	† 0.1933
WARP [23]	0.2769	0.1615	0.4248	0.3577	0.2686	0.1438	0.1435	0.1130	0.2989	0.1444
SINE [25]	0.1365	0.0877	0.2813	0.1041	0.1873	0.0857	0.1663	0.1160	0.2636	0.1065
HPE [3]	0.3584	0.1376	0.4575	0.3584	0.3380	0.1600	† 0.2091	† 0.1326	0.3552	0.1501
CKE [26]	0.3903	0.1986	0.4469	0.3570	0.3353	0.1838	0.1585	0.1245	0.3766	0.1887
KGAT [22]	† 0.3972	† 0.2049	† 0.4587	† 0.3710	† 0.3645	0.1864	0.1530	0.1155	† 0.4066	0.1699
TPR ($\lambda_{\text{WRR}} = 0.001$)	0.3911	0.1781	***0.4855**	*0.3803	*0.3786	0.1890	***0.2330**	***0.1541**	0.3992	0.1736
TPR ($\lambda_{\text{WRR}} = 0.005$)	*0.4155	0.2038	*0.4822	*0.3797	***0.3990**	*0.1960	*0.2195	*0.1487	*0.4233	0.1946
TPR ($\lambda_{\text{WRR}} = 0.01$)	***0.4201**	**0.2057**	*0.4755	***0.3819**	*0.3859	***0.2036**	*0.2133	*0.1477	***0.4245**	***0.1993**
Improv. (%)	+5.76%	+0.39%	+5.84%	+2.93%	+9.46%	+8.81%	+11.42%	+16.21%	+4.40%	+3.10%

Table 3: Performance on item-to-item recommendation

conducted on the five public Amazon datasets due to space limitations. Table 3 reports the results of item-to-item recommendation. Although many studies evaluate their methods with user-to-item recommendation, practically, item-to-item has a wider range of usage in many applications, such as "similar products" in e-commerce sites and recommendations under the heading "because you watched..." in video streaming services. Recall that GATE cannot be used for tasks other than typical user-item recommendation.

- Similar to the user-to-item recommendation, the models considering the textual information generally gain better performance than the ones leveraging purely user-item data.
- Similar to the user-to-item recommendation, HPE, CKE and KGAT still serve as strong baselines; even so, the proposed method consistently yields best performance compared to them. The overall improvement of TPR ranges from 4.40% to 11.42% in terms

	Amazon-Magazine		Amazon-Beauty		Amazon-Applications		Amazon-Fashion		Amazon-Software	
	Recall@10	NDCG@10	Recall@10	NDCG@10	Recall@10	NDCG@10	Recall@10	NDCG@10	Recall@10	NDCG@10
HPE [3]	† 0.2454	† 0.0609	† 0.1203	† 0.0419	† 0.0794	† 0.0101	† 0.1326	† 0.0344	† 0.1149	† 0.0114
CKE [26]	0.0636	0.0343	0.0093	0.0058	0.0072	0.0035	0.0090	0.0046	0.0104	0.0056
KGAT [22]	0.0363	0.0254	0.0152	0.0035	0.0033	0.0035	0.0152	0.0035	0.0083	0.0035
TPR ($\lambda_{\text{WRR}} = 0.001$)	*0.2636	*0.0875	***0.1654**	***0.0640**	***0.1569**	***0.0340**	***0.1698**	***0.0700**	***0.1511**	***0.0337**
TPR ($\lambda_{\text{WRR}} = 0.005$)	*0.2590	***0.0919**	*0.1483	*0.0570	*0.1175	*0.0200	0.1354	*0.0527	0.1176	0.0168
TPR ($\lambda_{\text{WRR}} = 0.01$)	***0.2863**	0.0609	*0.1320	*0.0501	*0.1026	*0.0167	0.1055	0.0374	0.0887	0.0106

Table 4: Performance on cold-start user-to-item recommendation

	Amazon-Magazine		Amazon-Beauty		Amazon-Applications		Amazon-Fashion		Amazon-Software	
	Recall@10	NDCG@10	Recall@10	NDCG@10	Recall@10	NDCG@10	Recall@10	NDCG@10	Recall@10	NDCG@10
HPE [3]	† 0.2090	† 0.0609	† 0.1133	† 0.0396	† 0.0794	† 0.0101	† 0.1337	† 0.0405	0.1106	0.0139
CKE [26]	0.0727	0.0343	0.0093	0.0035	0.0072	0.0035	0.0090	0.0046	0.0104	0.0056
KGAT [22]	0.0772	0.0432	0.0443	0.0117	0.0586	0.0068	0.0282	0.0080	† 0.1164	† 0.0181
TPR ($\lambda_{\text{WRR}} = 0.001$)	***0.2954**	***0.0697**	***0.1483**	***0.0617**	***0.1496**	***0.0200**	***0.1681**	***0.0546**	**0.1168**	***0.0264**
TPR ($\lambda_{\text{WRR}} = 0.005$)	0.2863	0.0520	*0.1320	*0.0454	0.1225	0.0167	0.1179	*0.0485	0.0883	0.0183
TPR ($\lambda_{\text{WRR}} = 0.01$)	0.2545	0.0520	*0.1250	*0.0431	0.1099	0.0167	0.1066	0.0400	0.0981	0.0139

Table 5: Performance on cold-start item-to-item recommendation

of Recall@10 and 0.39% to 16.21% in term of NDCG@10 for the five datasets.

3.4.3 Cold-start User-to-item Recommendation (Task 2-a). In order to recommend completely new items to users, which means that the items are not included in the training data, content-based information is crucial for such a cold-start scenario. Here we focus on the performance of recommending completely new items, the representations of which are obtained by averaging the word representations from the item descriptions, as described in Section 3.3. Recall that for tasks that involve word embeddings, i.e., for cold-start recommendation, item-to-word reconstruction, and user-to-word recommendation, only HPE, CKE, and KGAT are applicable.

- As shown in Table 4, for such a cold-start scenario, TPR outperforms all the state-of-the-art methods, including HEP, CKE, and KGAT. This superiority not only demonstrates the effectiveness of TPR exploring unseen items, but also reveals the capability of TPR modeling textual information for recommendation.
- In addition, we also notice that TPR obtains better recommendation results while adopting a smaller regularization on the WRR structure, λ_{WRR}. This is because with a small λ_{WRR}, the model strengthens the relations between items and words and ensures the words embedding quality, thereby benefiting the cold-start recommendation tasks.

3.4.4 Cold-start Item-to-item Recommendation (Task 2-b). Similar to the previous task, this task is also to recommend completely new items to users. For cold-start items, we generate their embeddings by averaging the embeddings of words contained in their descriptions.

- Table 5 shows that TPR also significantly outperforms all the compared methods for such a cold-start situation, which demonstrates the superiority of TPR modeling the word preference for recommendation.
- As CKE and KGAT are not designed for solving the cold-start problem, both methods yield poor performance for the two cold-start recommendation tasks in Sections 3.4.3 and 3.4.4,

3.4.5 Item-to-word Reconstruction (Task 3-a). In order to verify the TPR's capability of modeling textual information, we design this task of measuring how many related words can be captured by the learned item embeddings.

- Table 6 shows that, for such an item-to-word reconstruction task, TPR outperforms all the compared methods, including HPE, CKE, and KGAT. Similar to the phenomenon mentioned in Section 3.4.3, a small λ_{WRR} strengthens the item-word relations, which is clearly demonstrated in this experiment. Note that a larger word regularization allows TPR to explore missing relations and perform better in traditional recommendation tasks.
- The reason why CKE performs ineffectively is that this method has no direct modeling for relations between items and words; thus the learned embeddings cannot reflect the closeness between items and words.

3.4.6 User-to-word Recommendation (Task 3-b). Similarly, to verify the TPR's capability of modeling word preference on items, we turn to conduct the experiments from the perspective of users.

- Table 7 shows that TPR consistently yields the great performance on all the datasets in termso of both Recall@10 and NDCG@10, suggesting that TPR is able to connect unseen items by matching their descriptions via the TPR's user-word preference modeling.

3.5 Parameter Sensitivity and Memory and Time Usage

3.5.1 Parameters Sensitivity on Regularization. The results listed in Tables 2-7 suggest that adopting a larger regularization on the WRR structure (i.e., a larger λ_{WRR}), seems beneficial to the tasks that directly use user and item embeddings for score calculation,

	Amazon-Magazine		Amazon-Beauty		Amazon-Applications		Amazon-Fashion		Amazon-Software	
	Recall@10	NDCG@10	Recall@10	NDCG@10	Recall@10	NDCG@10	Recall@10	NDCG@10	Recall@10	NDCG@10
HPE [3]	† 0.7378	† 0.5910	† 0.7423	† 0.5651	† 0.7015	† 0.5383	† 0.5680	† 0.4927	† 0.5788	† 0.3682
CKE [26]	0.0441	0.0502	0.0222	0.0296	0.0435	0.0398	0.0124	0.0134	0.0635	0.0670
KGAT [22]	0.5273	0.4414	0.2884	0.2386	0.4109	0.2758	0.1869	0.1609	0.5138	0.3206
TPR ($\lambda_{\text{WRR}} = 0.001$)	***0.8371**	***0.7653**	***0.8244**	***0.7343**	***0.8530**	***0.7595**	***0.8645**	***0.8037**	***0.7159**	***0.4703**
TPR ($\lambda_{\text{WRR}} = 0.005$)	0.6294	0.5448	0.4738	0.3849	0.6109	0.5025	0.4930	0.4348	0.5476	0.3552
TPR ($\lambda_{\text{WRR}} = 0.01$)	0.5279	0.4411	0.4270	0.3580	0.4743	0.3722	0.3284	0.2825	0.4632	0.2949

Table 6: Performance on item-to-word reconstruction

	Amazon-Magazine		Amazon-Beauty		Amazon-Applications		Amazon-Fashion		Amazon-Software	
	Recall@10	NDCG@10	Recall@10	NDCG@10	Recall@10	NDCG@10	Recall@10	NDCG@10	Recall@10	NDCG@10
HPE [3]	† 0.2002	† 0.1938	† 0.2206	† 0.1684	† **0.1955**	† 0.1543	† 0.0829	† 0.0725	† 0.2414	† 0.1971
CKE [26]	0.0374	0.0576	0.0173	0.0263	0.0435	0.0398	0.0189	0.0213	0.0691	0.0723
KGAT [22]	0.0248	0.0375	0.0117	0.0161	0.0123	0.0201	0.0463	0.0644	0.0073	0.0153
TPR ($\lambda_{\text{WRR}} = 0.001$)	0.1996	0.1941	0.3411	*0.2943	0.1891	**0.1556**	*0.1053	*0.0833	***0.3018**	0.2332
TPR ($\lambda_{\text{WRR}} = 0.005$)	*0.2249	*0.2190	***0.3728**	***0.3147**	0.1818	0.1537	***0.1134**	***0.0865**	*0.2872	***0.2482**
TPR ($\lambda_{\text{WRR}} = 0.01$)	***0.2274**	***0.2198**	*0.3589	*0.2948	0.1690	0.1447	*0.1050	*0.0767	*0.2625	*0.2455

Table 7: Performance on user-to-word recommendation

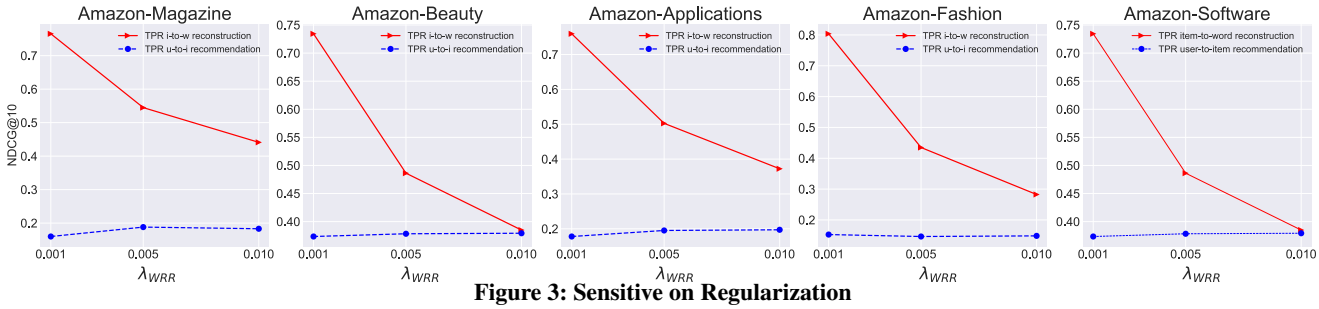

Figure 3: Sensitive on Regularization

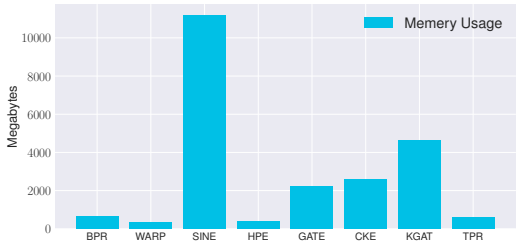

Figure 4: Memory Usage

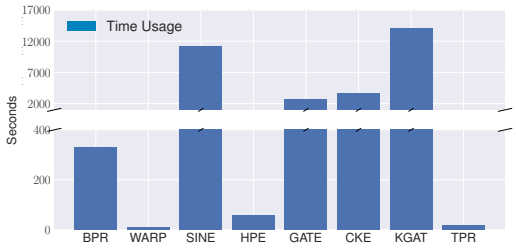

Figure 5: Execution Time Usage

i.e., user-to-item and item-to-item recommendations. In contrast, for the tasks that involve word embeddings for score calculation (see Sections 3.4.3-3.4.6), the phenomenon is inverse, especially for the task of item-to-word reconstruction.

Figure 3 illustrates this phenomenon with respect to the performance on the two tasks: user-to-item recommendation and item-to-word reconstruction. The reason for such a phenomenon is due to the fact that a small λ_{WRR} strengthens the item-word relations and ensures the words embedding quality, thereby benefiting the reconstruction and the cold-start recommendation tasks; on the contrary, a large λ_{WRR} encourages the model to explore unseen word relations for generating item embeddings with better generalization ability, thereby leading to better performance in the recommendation tasks. Hence, such a regularization weight can be treated as a trade-off parameter allowing additional flexibility in adjusting the relation modeling for different types of tasks.

3.5.2 Memory Usage and Execution Time. Figures 4 and 5 plot the time usage and the memory usage for model training on the Amazon-Fashion dataset, which is the largest dataset among all public datasets that the advanced models, GATE, CKE, and KGAT, can deal with under reasonable resource constrains. Note

that the values reported in the figure would be variant when different implementations are applied; the listed numbers are based on the implementations listed in footnote 3. With our implementation, TPR costs around 20 seconds to complete the whole training process and is much faster than the advanced model like CKE and KGAT. Moreover, our model works on only CPUs while CKE and KGAT adopt GPU for computation. On the other hand, TPR utilizes less memory than CKE and KGAT as well. Due to the simplicity of our model deign, the actual time usage of our model is near to the primitive matrix factorization models as the major additional computation regards the process of sampling the tuples for optimization. It is worth mentioning that it is hard to analyze time and memory usage in terms of complexity as the compared baselines and our method are essentially dissimilar in many aspects, which is the reason why we here compare the time and memory usage empirically.

4 CONCLUSION

In this paper, we propose TPR, a text-aware recommendation framework that models the joint association of user-item interaction and relations between items and associated text for top-N recommendation. Using the TPR framework, we design an optimization criterion that comprehensively models four types of ranking relations, yielding a unified and effective model that can be accommodated to various types of recommendation tasks. Extensive experiments for six different recommendation/reconstruction tasks are provided to attest the effectiveness of the learned embeddings of users, items, and words. The results show that TPR not only surpasses most state-of-the-art recommendation algorithms on various tasks but also achieves high modeling efficiency in terms of execution time and memory usage.

5 ACKNOWLEDGMENTS

This research is supported by the National Research Foundation, Singapore under its International Research Centres in Singapore Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

REFERENCES

[1] Ai, Q., Azizi, V., Chen, X., and Zhang, Y. Learning heterogeneous knowledge base embeddings for explainable recommendation. *Algorithms* (2018).

[2] Chen, C., Zhang, M., Liu, Y., and Ma, S. Neural attentional rating regression with review-level explanations. In *Proceedings of the 2018 World Wide Web Conference* (2018), WWW '18, International World Wide Web Conferences Steering Committee, p. 1583–1592.

[3] Chen, C.-M., Tsai, M.-F., Lin, Y.-C., and Yang, Y.-H. Query-based music recommendations via preference embedding. In *Proceedings of the 10th ACM Conference on Recommender Systems* (2016), RecSys '16, Association for Computing Machinery, p. 79–82.

[4] Chen, C.-M., Wang, C.-J., Tsai, M.-F., and Yang, Y.-H. Collaborative similarity embedding for recommender systems. In *Proceedings of the 28th International Conference on World Wide Web*, WWW '19, Association for Computing Machinery, p. 2637–2643.

[5] Cheng, H.-T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Ispir, M., Anil, R., Haque, Z., Hong, L., Jain, V., Liu, X., and Shah, H. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems* (2016), DLRS '16, Association for Computing Machinery, p. 7–10.

[6] Grbovic, M., and Cheng, H. Real-Time Personalization Using Embeddings for Search Ranking at Airbnb. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2018), KDD '18, Association for Computing Machinery, p. 311–320.

[7] He, X., and Chua., T.-S. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2017), SIGIR '17, Association for Computing Machinery, p. 355–364.

[8] He, X., Liao, L., Zhang, H., Nie, L., Hu, X., and Chua, T.-S. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web* (2017), WWW '17, International World Wide Web Conferences Steering Committee, pp. 173–182.

[9] Koren, Y., Bell, R., and Volinsky, C. Matrix factorization techniques for recommender systems. *Computer 42*, 8, 30–37.

[10] Le, Q., and Mikolov, T. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32* (2014), ICML '14, JMLR.org, p. II–1188–II–1196.

[11] Lin, Y., Liu, Z., Sun, M., Liu, Y., and Zhu, X. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence* (2015), AAAI '15, AAAI Press, p. 2181–2187.

[12] Liu, D., Li, J., Du, B., Chang, J., and Gao, R. Daml: Dual attention mutual learning between ratings and reviews for item recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2019), KDD '19, Association for Computing Machinery, p. 344–352.

[13] Ma, C., Kang, P., Wu, B., Wang, Q., and Liu, X. Gated attentive-autoencoder for content-aware recommendation. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining* (2019), WSDM '19, Association for Computing Machinery, p. 519–527.

[14] Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[15] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2* (2013), NIPS '13, Curran Associates Inc., p. 3111–3119.

[16] N. Sivaramakrishnan1, V.Subramaniyaswamy, A. V. V. V. N. S. A deep learning-based hybrid model for recommendation generation and ranking. *Neural Computing and Applications* (2020).

[17] Ni, J., Li, J., and McAuley, J. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (2019), Association for Computational Linguistics, pp. 188–197.

[18] Niu, F., Recht, B., Re, C., and Wright, S. J. Hogwild! a lock-free approach to parallelizing stochastic gradient descent. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS '11, Curran Associates Inc., p. 693–701.

[19] Rendle, S. Factorization Machines with LibFM. *ACM Transactions on Intelligent Systems and Technology 3*, 3 (2012), 1–22.

[20] Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, UAI '09, AUAI Press, p. 452–461.

[21] Wang, J., Huang, P., Zhao, H., Zhang, Z., Zhao, B., and Lee, D. L. Billion-scale commodity embedding for e-commerce recommendation in alibaba. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2018), KDD '18, Association for Computing Machinery, p. 839–848.

[22] Wang, X., He, X., Cao, Y., Liu, M., and Chua, T.-S. KGAT: Knowledge Graph Attention Network for Recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '19, Association for Computing Machinery, p. 950–958.

[23] Weston, J., Bengio, S., and Usunier, N. Wsabie: Scaling up to large vocabulary image annotation. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, IJCAI '11, AAAI Press, p. 2764–2770.

[24] Xiao, J., Ye, H., He, X., Zhang, H., Wu, F., and Chua, T.-S. Attentional factorization machines: Learning the weight of feature interactions via attention networks. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence* (2017), IJCAI '17, AAAI Press, p. 3119–3125.

[25] Zhang, D., Yin, J., Zhu, X., and Zhang, C. SINE: Scalable incomplete network embedding. In *Proceedings of the IEEE International Conference on Data Mining* (2018), ICDM '18, IEEE, pp. 737–746.

[26] Zhang, F., Yuan, N. J., Lian, D., Xie, X., and Ma, W.-Y. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016), KDD '16, Association for Computing Machinery, p. 353–362.

[27] Zhao, W. X., He, G., Yang, K., Dou, H., Huang, J., Ouyang, S., and Wen, J. Kb4rec: A data set for linking knowledge bases with recommender systems. *Data Intelligence* (2019), 121–136.