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 National Chengchi University cjwang@citi.sinica.edu.tw
Ming-Feng Tsai National Chengchi University mftsa@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 expresses 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 methodology 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.

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...
learning criterion that comprehensively models four types of structures and then discusses 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 $E_{ui} = \{(u, i) | u \in U, i \in I\}$, and 2) the "has-a" relation between items and words, denoted as $E_{iw} = \{(i, w) | i \in I, w \in W\}$, where the words for each item are extracted from its description.

The goal of our model is to learn a representation matrix $\Theta \in \mathbb{R}^{|U|+|I|+|W| \times d}$ 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 tasks.

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 recommendation, 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.

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.
item-to-item, and user-to-word recommendation, as well as item-to-word reconstruction, as motivated in Section 1.

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 $\Theta$ that maximizes the joint likelihood function from observed user-item and item-word pairs:

$$O_{\text{TPR}} = \max \prod_{u,i \in E_{ui}} p(>u, >i | \Theta), \quad (1)$$

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 $E_{ui}$ for $>u$, and another set of triples $D_i : I \times W \times W$ based on the has-a relations between items and words $E_{iw}$ for $>i$, as $D_u = \{(u, j, j’)|u \in U, j \in I \},$ $D_i = \{(i, w, w')|i \in I, w \in W \},$ and $D = \{(i, w, w')|i \in I, w \in W \},$ respectively, where $I_u = \{i \in I|(u, i) \in E_{ui}\}$ denotes the set of items that user $u$ has interacted with, and $W = \{w \in W|i, w \in E_{iw}\}$ 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 $(u, i) \in E_{ui}$, 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

$$O_{\text{IPR}} = \max \prod_{u \in U} p(>u | \Theta)$$

$$= \max \ln \left( \prod_{u \in U} p(>u | \Theta) \right)$$

$$= \max \ln \left( \sum_{(u,j,j') \in D_u} p(j > u j’ | \Theta) \right). \quad (2)$$

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

$$p(j > u j’ | \Theta) = \sigma(\langle \Theta_u, \Theta_j - \Theta_{j’} \rangle), \quad (3)$$

where $\langle \cdot, \cdot \rangle$ denotes the dot product between two vectors, $\Theta_u (\Theta_j)$ denotes the $d$-dimensional row vector from $\Theta$ 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

$$O_{\text{WRR}} = \max \prod_{i \in I} p(>i | \Theta)$$

$$= \max \ln \left( \prod_{i \in I} p(>i | \Theta) \right)$$

$$= \max \ln \left( \sum_{(i,w,w') \in D_i} p(w > i w’ | \Theta) \right). \quad (4)$$

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), $p(w > i w’ | \Theta)$ is calculated as

$$p(w > i w’ | \Theta) = \sigma(\langle \Theta_w, \Theta_i - \Theta_{w’} \rangle), \quad (5)$$

where $\Theta_w$ is the $d$-dimensional row vector from $\Theta$ 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
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 $\Theta$ 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

$$
\text{TPR-OPT} := \ln p(\Theta) > u, > i) \propto \ln p( > u, > j | \Theta)p(\Theta)
$$

$$
= \ln \prod_{(u,j,j') \in D_u, (i,w,w') \in D_i} p( > u, > j | \Theta)p(\Theta)
$$

$$
= \sum_{(u,j,j') \in D_u, (i,w,w') \in D_i} \ln \sigma \left( (\Theta_u + \Theta_i, (\Theta_j - \Theta_j') + (\Theta_w - \Theta_w')) \right) - \lambda_{\Theta} \| \Theta \|^2.
$$

(7)

where $\lambda_{\Theta}$ 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 $\alpha$:

$$
\Theta \leftarrow \Theta + \alpha \frac{\partial \text{TPR-OPT}}{\partial \Theta}.
$$

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 $\Theta$ 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

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

$$
= \frac{e^{-\hat{x}}}{1 + e^{-\hat{x}}} \frac{\partial}{\partial \Theta} \hat{x} - \lambda_{\Theta} \Theta.
$$

(8)

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

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 an embedding space. Moreover, word $w$ in (c) is extracted from the description of an item $i$ that the user has interacted with (i.e., $i \in I^+_w$); 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 $\Theta$ 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.
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 $\Theta_k$ in $\Theta$ is updated by

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

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—$\lambda_{\text{IPR}}$ and $\lambda_{\text{WRR}}$—allow additional flexibility in adjusting the relation modeling for different types of tasks. The effect of adopting different values of $\lambda_{\text{IPR}}$ and $\lambda_{\text{WRR}}$ 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], 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

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Users</th>
<th>Items</th>
<th>Words</th>
<th>U-I edges</th>
<th>I-W edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon-Magazine</td>
<td>2,825</td>
<td>1,299</td>
<td>6,740</td>
<td>11,685</td>
<td>9,4381</td>
</tr>
<tr>
<td>Amazon-Beauty</td>
<td>4,801</td>
<td>4,865</td>
<td>4,115</td>
<td>11,685</td>
<td>159,475</td>
</tr>
<tr>
<td>Amazon-Application</td>
<td>11,823</td>
<td>5,554</td>
<td>9,712</td>
<td>42,675</td>
<td>410,079</td>
</tr>
<tr>
<td>Amazon-Software</td>
<td>13,614</td>
<td>3,255</td>
<td>11,111</td>
<td>57,793</td>
<td>766,112</td>
</tr>
<tr>
<td>Amazon-Fashion</td>
<td>19,875</td>
<td>36,080</td>
<td>5,076</td>
<td>75,596</td>
<td>442,136</td>
</tr>
<tr>
<td>Amazon-Kindle</td>
<td>363,303</td>
<td>356,634</td>
<td>36,445</td>
<td>3,334,521</td>
<td>6,794,209</td>
</tr>
<tr>
<td>Course</td>
<td>~17,000</td>
<td>~600</td>
<td>~3,000</td>
<td>~300,000</td>
<td>~150,000</td>
</tr>
<tr>
<td>SG-OPN</td>
<td>~10,000</td>
<td>~12,000</td>
<td>~10,100</td>
<td>~30,000</td>
<td>~1,100,000</td>
</tr>
</tbody>
</table>

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.

---

3https://nijianmo.github.io/amazon/index.html

(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 \( \langle \Theta_u, \Theta_i \rangle \), 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 recom-
 mend 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 \( \Theta_j = \frac{1}{|W|} \sum_{w \in W} \Theta_w \). For each user \( u \), the recommenda-
tions 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 pre-
 dict 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 \( w \in W \setminus W^+_u \) 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
\( \Theta_w \) (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 \( \lambda_{\text{WRR}} \)
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 exam-
ine 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, re-
spectively. Note that in the reported results, “†” symbol in Tables 2-7
indicates the best performing method among all the baseline meth-
ds; “*” 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 recom-

mend 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. 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 ad-
anced 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
discussed in Section 3.5.2.
- For such a user-to-item recommendation task, adopting a larger
regularization on the WRR structure, \( \lambda_{\text{WRR}} \), benefits model gen-
eralization 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 fol-

lowing tasks in Sections 3.4.3-3.4.6, we only report results conducted

---

3The model is unable to be loaded into the GPU with 32GB memory or the training is
unable to be finished within days.
recommendations under the heading “because you watched...” in applications, such as “similar products” in e-commerce sites and video streaming services. Recall that GATE cannot be used for tasks other than typical user-item recommendation.

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

Table 2: Performance on user-to-item recommendation

Table 3: Performance on item-to-item recommendation

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

• 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.
averaging the embeddings of words contained in their descriptions. For cold-start items, we generate their embeddings by

- user-to-word recommendation, only HPE, CKE, and KGAT are applied for cold-start recommendation, item-to-word reconstruction, and task of measuring how many related words can be captured by the learned item embeddings.

- Cold-start User-to-item Recommendation (Task 2-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 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 $\lambda_{WRR}$ 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 terms 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 $\lambda_{WRR}$), seems beneficial to the tasks that directly use user and item embeddings for score calculation, 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 $\lambda_{WRR}$ strengthens the item-word relations and ensures the words embedding quality, thereby benefiting the reconstruction and the cold-start 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 and memory usage for model training on the Amazon-Fashion dataset, which is the largest dataset among all public datasets that those advanced models, such as KGAT, can address under reasonable resource constraints. 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 other advanced models. Moreover, our model utilizes less memory and works only on CPUs, while CKE and KGAT adopt GPU for computation with more memory usage. Due to the simplicity of our model design, 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 regard 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 to model four types of ranking relations, and yield a unified and effective model on various types of tasks. Extensive experiments for six different recommendation/reconstruction tasks are provided to attest the effectiveness of those learned embeddings. The results show that TPR not only surpasses most state-of-the-art recommendation algorithms on various tasks but also achieves high modeling efficiency regarding 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