DEAMER: A Deep Exposure-Aware Multimodal Content-Based Recommendation System

Yunsen Hong, Hui Li, Xiaoli Wang, and Chen Lin(✉)

School of Informatics, Xiamen University, Xiamen, China
yshong@stu.xmu.edu.cn, {hui,xlwang,chenlin}@xmu.edu.cn

Abstract. Modern content-based recommendation systems have greatly benefited from deep neural networks, which can effectively learn feature representations from item descriptions and user profiles. However, the supervision signals to guide the representation learning are generally incomplete (i.e., the majority of ratings are missing) and/or implicit (i.e., only historical interactions showing implicit preferences are available). The learned representations will be biased in this case; and consequently, the recommendations are over-specified. To alleviate this problem, we present a Deep Exposure-Aware Multimodal contEnt-based Recommender (i.e., DEAMER) in this paper. DEAMER can jointly exploit rating and interaction signals via multi-task learning. DEAMER mimics the expose-evaluate process in recommender systems where an item is evaluated only if it is exposed to the user. DEAMER generates the exposure status by matching multi-modal user and item content features. Then the rating value is predicted based on the exposure status. To verify the effectiveness of DEAMER, we conduct comprehensive experiments on a variety of e-commerce data sets. We show that DEAMER outperforms state-of-the-art shallow and deep recommendation models on recommendation tasks such as rating prediction and top-k recommendation. Furthermore, DEAMER can be adapted to extract insightful patterns of both users and items.

Keywords: Neural recommender systems · Content-based recommendation · Deep generative model · Multi-task learning · Multi-modal learning

1 Introduction

Recommender systems have been widely used in e-commerce platforms such as Amazon and eBay. Recommender systems endow e-commerce platforms with the capability to deliver personalized filtering of innumerable consumption choices. Tackling the over-choice problem in online shopping brings both better user experience and higher enterprise revenue. Therefore, during the past decades,

Y. Nah et al. (Eds.): DASFAA 2020, LNCS 12114, pp. 621–637, 2020.
https://doi.org/10.1007/978-3-030-59419-0_38

recommender systems have become an indispensable part in e-commerce platforms and enormously impacted other domains, such as social networks and media websites [1].

Abundant content information is available on e-commerce platforms, including item meta-data, user profiles, user generated reviews, and other auxiliary data sources such as videos and images. To exploit content information, content-based methods, which is a main paradigm of recommender systems, have been extensively studied in the past. Content-based methods offer recommendations based on comparisons across contents of users and items. Comparisons can be done by either finding exact keywords or computing relevance scores on handcrafted and/or learned content representations [29]. Recent breakthroughs of deep learning boost the performance of content-based methods, as deep neural networks are superior at deriving underlying representations without handcrafted features [35]. Compared to the other paradigm of recommender systems, i.e., collaborative filtering methods which solely learn user preferences from user-item historical interactions, content-based methods are able to alleviate the cold-start problem effectively [1]. When a new user has zero or only a few ratings, content-based approaches can still generate reasonable recommendations by performing content-based comparisons. However, it is reported that content-based methods lack some diversity and serendipity, which leads to inferior performance to collaborative filtering methods when sufficient ratings are provided [29].

One possible reason for the inferior performance of content-based approaches is that the supervision signals which are used to guide the representation learning of content information are incomplete or implicit in general. Traditional content-based methods require users to label item documents by assigning a relevance score [1,29]. Such approaches are labor-consuming and thus not applicable. Other content-based methods, including shallow [18] and deep models [21] use ratings to measure the content relevance. However, rating signals are generally missing for a large portion of user-item pairs in practice. Moreover, content-based methods will learn over-specialized representations on items that users already like, as well as non-distinguishable representations on items that users have never rated. To cope with aforementioned issues, some contemporary content-based approaches consider binary interactions (i.e., implicit feedback), e.g., whether a user has rated, clicked, tagged or consumed an item. In particular, a positive interaction usually indicates positive preference, while a zero interaction does not indicate negative preference. Nevertheless, content-based methods using implicit feedback are still unable to learn accurate representations for user-item pairs with zero interactions.

Let us recall the expose-evaluate process which is typically adopted by users when using a recommender system. The process is shown in Fig. 1. A user will rate an item, only if the item is exposed to him/her. To further specify the process, we make the following assumptions: (1) The decision of exposure can be roughly interpreted as a quick content matching. It does not matter whether the occurrence of exposure is due to the pro-activeness of users or the intelligent decision from recommender systems. Exposure indicates preferences.

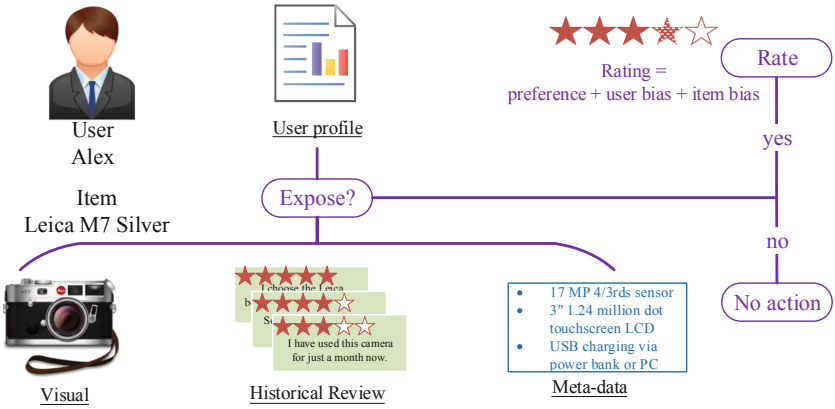


Fig. 1. Expose-evaluate process where an item is rated after it is exposed to a user

The user will interact (e.g., rate, consume or click) with an exposed item. (2) Once exposed, the item will be further evaluated (in many scenarios, it means the item will be consumed) and rated. A well-known and successful assumption is that the value of rating is an aggregation of user preference, user bias and item bias [13].

The expose-evaluate process inspires us to generate rating values based on interactions in learning content representations and improve the quality of content-based recommendation. We present a Deep Exposure-Aware Multimodal contEnt-based Recommender (i.e., DEAMER). DEAMER can jointly exploit rating and interaction signals via multi-task learning and it consists of two modules. The exposure generation module predicts the binary exposure based on representations extracted from multi-modal contents including item meta-data, item images, and historical reviews grouped by users and items. The predicted probability of an interaction, which is treated as the user preference over an item, is transmitted to the rating generation module which estimates the user-item relevance (i.e., rating) based on the user-item preference, user-specific and item-specific embeddings.

In summary, the contributions of this paper is three-fold:

- **A novel deep generative model.** Unlike previous methods which model ratings and interactions in parallel [22], DEAMER generates ratings from exposures. The content representations are first learned in the coarse-grained bottom module and then finer tuned in the top module.
- **A multi-modal multi-task framework.** Multi-modal content information (i.e., visuals and texts) and heterogeneous supervision signals (i.e., ratings and interactions) are utilized to improve recommendations in DEAMER.
- **Superior recommendation performance and a behavior analysis.** We experimentally verify, on a variety of e-commerce data sets, that DEAMER outperforms state-of-the-art shallow and deep recommendation models. Furthermore, we show that DEAMER is able to capture interesting patterns of

users and items. For example, DEAMER learns user and item embeddings that can distinguish active users and inactive users, popular items and niche items, etc.

2 Related Work

Recommendation systems can be generally classified into three categories: content-based methods, collaborative filtering methods, and hybrid methods which combine both of them [1,29]. There is a tremendous amount of work on recommender systems in the literature and we only introduce the most relevant methods to DEAMER. Readers can refer to surveys [1,29,35] for more information about recommender systems.

Content-based methods essentially make recommendations based on the degrees of resemblance between item content and user profiles. Existing works have investigated the mining of various content features [1,29] including social network [19], user grouping data [2,18], relationships in a graph [27], time-series information [17], locations [24], review text [5], just to name a few. The rapid development of deep neural network techniques have fostered modern content-based recommenders. Many deep neural networks have been leveraged to analyze different data modalities. For example, CNNs are often used for image feature extraction, CNNs and RNNs are usually necessary to deal with texts, and attention mechanism is commonly adopted for textual and tag information [35]. To learn the corresponding user and item representations over heterogeneous information, researchers recently turn to multi-modal learning [33,36] in order to incorporate multiple types of information sources (e.g., review text, product image and numerical rating) into one content-based recommendation model.

Collaborative filtering methods extract users with similar tastes as the target user from interactions/rating data only. In the fruitful literature of collaborative filtering methods, matrix factorization [13,16] and factorization machines [28] have exhibited superior performances in rating prediction task. However, it is problematic to only consider ratings, since the underlying rating matrix is *sparse*. When ratings are *Missing Not At Random* (MNAR) [26], the performance of collaborative filtering methods will further degrade. Therefore, many MNAR-targeted models have been proposed [23,26]. Recent advances of deep neural networks have also benefited collaborative filtering methods. For instance, NeuCF [9] generalizes matrix factorization with multi-layer perceptron. DeepFM [7] and xDeepFM [22] extend factorization machines by learning high-order feature interactions via a compressed interaction network. AutoRec [31] introduced autoencoder into collaborative filtering.

In many cases where a wider variety of data sources is available, one has the flexibility of using both content-based and collaborative filtering recommenders for the task of recommendation. Using **hybridization of different types of recommenders**, the various aspects from different types of recommender systems are combined to achieve the best performance. There is a surge of works on hybrid recommenders. HybA [6] is a hybrid recommender system for automatic

playlist continuation which combines Latent Dirichlet Allocation and case-based reasoning. MFC [20] is a hybrid matrix factorization model which uses both ratings and the social structure to offer recommendations. Deep neural network has also been introduced into hybrid recommender systems. For instance, CFN [32] utilizes autoencoder and incorporates side-information to construct a hybrid recommender. aSDAE [3] is a hybrid model which combines additional stacked denoising autoencoder and matrix factorization together.

To the best of our knowledge, there exists methods modeling ratings and explicit interactions in parallel [22], but the dependency between ratings values and explicit user-item interactions has not been explored in deep neural network architecture for the recommendation task. In addition to generalizing and extending the shallow MNAR and exposure-aware collaborative filtering models, DEAMER learns the ratings through multi-modal content information. Thus the contents are assembled to provide more accurate recommendations than pure collaborative filtering or content-based methods.

3 DEAMER

In this section, we will elaborate on the proposed Deep Exposure-Aware Multimodal Content-based Recommender System (i.e., DEAMER).

Suppose that we have a set of users $\mathcal{U}$ and a set of items $\mathcal{V}$. For each user $u \in \mathcal{U}$ and each item $v \in \mathcal{V}$, the inputs $\mathbf{X}_u$ and $\mathbf{X}_v$ consists of user and item contents, respectively (they will be described below); the output contain an observed interaction $y_{u,v}$ and an observed rating $r_{u,v}$. The interaction is binary (i.e., $y_{u,v} \in \{0, 1\}$, where 1 indicates that the user has rated/clicked/viewed the item, and 0 otherwise). The rating is numerical and normalized (i.e., $r_{u,v} \in [0, 1]$). We aim to estimate proper parameters which can generate observations $y_{u,v}$, $r_{u,v}$ given $\mathbf{X}_u$ and $\mathbf{X}_v$ for $\forall u \in \mathcal{U}$ and $\forall v \in \mathcal{V}$.

The user content $\mathbf{X}_u$ is constructed by the user's historic reviews on different items. That is, we aggregate u's reviews and make $\mathbf{X}_u = (x_{u,1}, \cdots, x_{u,T_{(x,u)}})$, where $x_{u,i}$ is the i-th word and $T_{(x,u)}$ is the number of words in u's content. We gather item content $\mathbf{X}_v$ in three different modalities: meta-data, textual reviews and images (i.e., $\mathbf{X}_v = \{\mathbf{w}_v, \mathbf{m}_v, \mathbf{G}_v\}$). For each item v, we aggregate its reviews from different users as its textual reviews. Item reviews for item v is a sequence of words $\mathbf{w}_v = (w_{w,1}, \cdots, w_{v,T(w,v)})$ where $w_{v,i}$ is the i-th word and $T_{(w,v)}$ is the number of words in v's reviews. We extract the name and textual description of item as its meta-data. The meta-data for item v is a sequence of words $\mathbf{m}_v = (m_{v,1}, \cdots, m_{v,T_{(m,v)}})$ where $m_{v,i}$ is the i-th word and $T_{(m,v)}$ is the number of words in v's meta-data. We use images which are associated with an item as its visual input. The image is represented as $\mathbf{G}_v \in \mathcal{R}^{N(G)\times N(G)}$ where $N(G)$ is the dimensionality of the image.

Figure 2 depicts the overall architecture of DEAMER. The inputs $\mathbf{X}_u$ and $\mathbf{X}_v = \{\mathbf{w}_v, \mathbf{m}_v, \mathbf{G}_v\}$ are first passed to the exposure generation module at the bottom which leverages real binary interaction to guide the exposure generation. Then, rating generation module on the top uses real rating value to generate

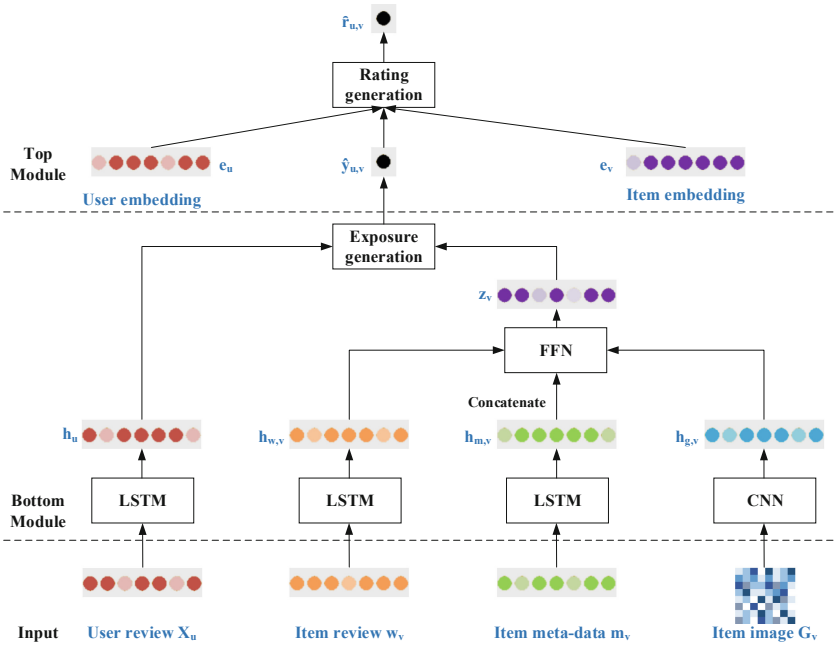


Fig. 2. Overall architecture of DEAMER

ratings. In the following, we will illustrate the exposure generation module and rating generation module of DEAMER in detail.

3.1 Exposure Generation Module

We extract feature representations from different modalities separately. For each pair of user u and item v, we use LSTM [10] to extract the features from $\mathbf{X}_u$, $\mathbf{w}_v$ and $\mathbf{m}_v$.

LSTM reads a sequence of tokens and generates hidden states one by one. For the t-th word n_t in each word sequence $(n_1, \ldots, n_T)$ ($\{n_t, T\}$ can be one of $\{x_{u,t}, T_{(x,u)}\}$, $\{w_{v,t}, T_{(w,v)}\}$ and $\{m_{v,t}, T_{(m,v)}\}$), LSTM learns a hidden state $\mathbf{h}_t$. To be specific, LSTM operates on the hidden state $\mathbf{h}_{t-1}$ of the previous token n_{t-1} and the input vector $\mathbf{n}_t$ of current token n_t to get an input gate $\mathbf{i}_t$, a forget gate $\mathbf{f}_t$, an output gate $\mathbf{o}_t$ and the cell $\mathbf{g}_t$ for the t-th input:

$$\begin{aligned} \mathbf{i}_t &= \sigma(\mathbf{W}_i[\mathbf{n}_t, \mathbf{h}_{t-1}] + \mathbf{b}_i) \\ \mathbf{f}_t &= \sigma(\mathbf{W}_f[\mathbf{n}_t, \mathbf{h}_{t-1}] + \mathbf{b}_f) \\ \mathbf{o}_t &= \sigma(\mathbf{W}_o[\mathbf{n}_t, \mathbf{h}_{t-1}] + \mathbf{b}_o) \\ \mathbf{g}_t &= \phi(\mathbf{W}_g[\mathbf{n}_t, \mathbf{h}_{t-1}] + \mathbf{b}_g) \end{aligned} \tag{1}$$

where $\mathbf{W}_i$,$\mathbf{W}_f$, $\mathbf{W}_o$ and $\mathbf{W}_g$ are the learnable weight matrices, and $\mathbf{b}_i$, $\mathbf{b}_f$, $\mathbf{b}_o$ and $\mathbf{b}_g$ are bias vectors. $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid activation function, and $\phi(x) = \frac{x}{1+|x|}$ is the softsign activation function.

Then, LSTM computes the cell state and the hidden state for the t-th input:

$$\begin{aligned} \mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \\ \mathbf{h}_t &= \mathbf{o}_t \odot \phi(\mathbf{c}_t) \end{aligned} \tag{2}$$

where $\odot$ indicates the Hadamard product.

We use the final hidden state $\mathbf{h}_T$ as the representation for user reviews, item reviews and item meta-data:

$$\mathbf{h}_u = \mathbf{h}_{T(x,u)},\ \mathbf{h}_{w,v} = \mathbf{h}_{T(w,v)},\ \mathbf{h}_{m,v} = \mathbf{h}_{T(m,v)} \tag{3}$$

where $\mathbf{h}_u$, $\mathbf{h}_{w,v}$ and $\mathbf{h}_{m,v}$ are the representations learned from user u's reviews, item v's reviews and item v's meta-data, respectively.

We utilize CNN [15] to extract features from item visual content $\mathbf{G}_v$. In CNN, each neuron j in the convolutional layer uses a filter $K_j \in \mathcal{R}^{S\times S}$ on a slide window of size $S \times S$. We obtain a feature map f_j for each neuron j:

$$b_j = ReLU(\mathbf{G}_{1:N(G),1:N(G)} \star K_j), \tag{4}$$

where $\star$ is the convolutional operator, and $ReLU(x) = \max\{0, x\}$ indicates the Rectified Linear Units (ReLU) activation function. Then we apply a max pooling operation of a slide window size $P \times P$ over the new feature map b_j, which is calculated by:

$$q_j = \max \begin{pmatrix} b_j & \cdots & b_{j+P-1} \\ \vdots & \ddots & \vdots \\ b_{j+(P-1)\cdot L} & \cdots & b_{j+(P-1)\cdot L+P-1} \end{pmatrix}, \tag{5}$$

where $L = N(G) - S + 1$ represents the length of side of the new feature map after the convolutional operation.

In this framework, multiple filters are used to gain informative feature maps, resulting $\mathbf{Q} = flat\{q_1, \ldots, q_I\}$, where I denotes the number of kernels in the convolutional layer and $flat$ is the flatten operation. Then the image representation $\mathbf{h}_{g,v}$ can be obtained by:

$$\mathbf{h}_{g,v} = ReLU(\mathbf{W}_G\mathbf{Q}), \tag{6}$$

where $\mathbf{W}_G$ is the learnable weight matrix.

Given the representations learned separately from heterogeneous content sources of item v, we first concatenate all the representations of item v:

$$\mathbf{h}_v^c = concat(\mathbf{h}_{w,v}, \mathbf{h}_{m,v}, \mathbf{h}_{g,v}), \tag{7}$$

where $concat(\cdot)$ is the concatenation operation. The concatenated representation is then passed through a fully connected layer to further reduce its dimensionality and generate the representation $\mathbf{z}_v$ of item v:

$$\mathbf{z}_v = ReLU(\mathbf{W}_z\mathbf{h}_v^c), \tag{8}$$

where $\mathbf{W}_z$ denotes the learnable weight matrix.

Finally, we use $\mathbf{h}_u$ and $\mathbf{z}_v$ to predict the probability of generating a positive interaction $\hat{y}_{u,v}$ between user u and item v:

$$\hat{y}_{u,v} = \sigma\big(\mathbf{W}_L(\mathbf{h}_u \odot \mathbf{z}_v)\big), \tag{9}$$

where $\mathbf{W}_L$ is the weight matrix in this layer.

3.2 Rating Generation Module

The rating generation module shown in the top of Fig. 2 takes the output $\hat{y}_{u,v}$ from exposure generation module, the user embedding vector $\mathbf{e}_u \in \mathcal{R}^{D_e}$ for user u and the item embedding vector $\mathbf{e}_v \in \mathcal{R}^{D_e}$ for item v as inputs where D_e is the dimensionality of embeddings.

DEAMER first concatenates the three inputs and then transform the concatenation to generate the rating. The procedure can be formally defined as:

$$\hat{r}_{u,v} = \sigma\Big(\mathbf{W}_R \cdot concate(\mathbf{e}_u, \hat{y}_{u,v}, \mathbf{e}_v)\Big), \tag{10}$$

where $\mathbf{W}_R$ denotes the weight matrix and σ is the sigmoid activation function.

3.3 Training

DEAMER optimizes its parameters by using multi-task learning and minimizing the following loss function:

$$\mathcal{L} = \sum_u \sum_v \{[y_{u,v} \log \hat{y}_{u,v} + (1 - y_{u,v}) \log(1 - \hat{y}_{u,v})] + \lambda[r_{u,v} - \hat{r}_{u,v}]^2\}, \tag{11}$$

where the first term is the interaction prediction loss, the second term is the rating prediction loss, and the loss coefficient λ is used to balance the two prediction tasks.

In the implementation, we pretrain the word embeddings of user reviews, item metadata and item reviews using Doc2vec [14] and set the pretrained embedding size for each word as 500. The pretrained item image embeddings is available from the data sets we used which will be illustrated in Sect. 4.1.

3.4 Discussion

The probabilistic matrix factorization (PMF) [30] assumes a rating is sampled from a Gaussian distribution:

$$p(r_{u,v}) = \mathcal{N}(r_{u,v}; \frac{1}{1 + \exp[-(\mathbf{uv} + b_u + b_v)]}, \beta^{-1}), \tag{12}$$

where $\mathbf{u}, \mathbf{v}$ can be considered as user and item embeddings, β^{-1} is the precision of the Gaussian distribution. Under this assumption, the likelihood $p(r_{u,v})$ is equivalent to the square loss function.

Table 1. Statistics of data sets

Data sets	# Users	# Items	# Ratings	RR (%)	MM (%)	MI (%)
Musica Instruments (MI)	1,429	900	10,261	0.798	0.22	1.00
Office Products (OP)	4,905	2,420	53,258	0.449	0.04	0.58
Digital Music (DM)	5,541	3,568	64,706	0.327	11.66	0.28
Sports and Outdoors (SO)	35,598	18,357	296,337	0.045	0.22	0.98
Health and Personal care (HP)	38,609	18,534	346,355	0.048	0.19	0.92

Given observations $r_{u,v}$ and $y_{u,v}$, the likelihood $p(r_{u,v}, y_{u,v})$ can be decomposed as:

$$p(r_{u,v}, y_{u,v}) = p(y_{u,v})p(r_{u,v}|y_{u,v}). \tag{13}$$

Therefore, we can see that DEAMER is essentially a generative model for two tasks:

1. The exposure generation module at the bottom of Fig. 2 estimates $p(y_{u,v})$ based on user and item contents. The first term of Eq. 11 is the negative likelihood of $p(y_{u,v})$ where $p(y_{u,v} = 1) = \sigma\big(\mathbf{W}_L(\mathbf{h}_u \odot \mathbf{z}_v)\big)$.
2. The rating generation module in the top of Fig. 2, which assesses $p(r_{u,v}|y_{u,v})$, is based on the assumption of PMF. But it is more expressive than standard PMF. Equation 10 generalizes Eq. 12 when $\mathbf{W}_R, \mathbf{W}_L$ are both unit matrices, $\mathbf{h}_u = \mathbf{u}, \mathbf{z}_v = \mathbf{v}$, $\mathbf{e}_u \in \mathcal{R}^{D_e}$, $\mathbf{e}_v \in \mathcal{R}^{D_e}$ and σ is the sigmoid function. The second term of Eq. 11 is the negative likelihood of PMF.
3. We can derive that the loss coefficient $\lambda = 2/\beta$ from Eqs. 11, 12 and 13.

4 Experiment

In this section, we conduct experiments to answer the following research questions:

1. Does DEAMER perform well on recommendation tasks, including rating prediction and top-k recommendation?
2. Do multi-modal learning and multi-task learning contribute to the recommendation of DEAMER?
3. How do the hyper-parameters affect the performance?
4. Can we discover interesting patterns by analyzing the user and item embeddings?

4.1 Experimental Setup

Data Sets. We adopt several public E-commerce data sets,[1] which are standard benchmarks in the recommendation community in our experiments. These data

[1] http://jmcauley.ucsd.edu/data/amazon/.

sets contain item descriptions, user reviews and ratings on the Amazon web store. We conduct experiments on five representative categories with different sizes and densities, as well as missing rate of meta-data and image. For each item, if there does not exist meta-data or an image, we regard it as missing. The statistics of the adopted data sets are listed in Table 1. We also provide the density–ratio of observed ratings (i.e., RR), the percentage of missing item meta-data (i.e., MM) and the percentage of missing item image (i.e., MI) of each data set.

Experimental Protocol. We use leave-one-out in both rating prediction and top-k recommendation tasks. In rating prediction, for each user in the data set, we holdout one rating randomly as test instance. In top-k recommendation, for each user, we use the holdout sample as the positive test instance and randomly sample 99 items that the user did not interact before as the negative test instances. The reported results are averaged over five runs.

Hyper-parameter Settings. Unless stated otherwise, we use the optimal hyper-parameters turned in the smallest MI data set for DEAMER. We leverage grid search, where the dimensionality of hidden states (i.e., $\mathbf{h}_u$, $\mathbf{h}_{w,v}$ and $\mathbf{h}_{m,v}$) D_h as well as dimensionality of embeddings D_e are searched in the range $\{32, 64, 128, 256\}$. The regularization weight and the dropout rate are tuned in the ranges of $\{0.1, 0.01, 0.001\}$ and $\{0.1, 0.25, 0.5\}$, respectively. The best performing D_h, D_e, regularization weight, and dropout rate are 256, 64, 0.01 and 0.25, respectively. In our training step, we set $\lambda = 1.2, 10, 15, 30, 40$ for data sets MI, OP, DM, SO and HP, respectively. We will report the impacts of different λ in Sect. 4.4. During training, for each user-item interaction training pair, we sample 5 items that this user has not interacted before. For CNN, we set the convolutional slide window size as 3×3 (i.e., $S = 3$) with slide step size 1×1, the max pooling size as 2×2 (i.e., $P = 2$) with step size 2×2, and the number of filters as 8. The Adam optimizer is employed with the initial learning rate, the training batch size and the max training epoch being 0.001, 256 and 80, respectively.

4.2 Analysis of Recommendation Performance

One advantage of DEAMER is that it can simultaneously predict rating values and make top-k recommendation. Thus, we conduct comparative studies on its regression performance (i.e., how close the predicted ratings are to the true ratings) and ranking performance (i.e., how close is the output recommendation list to the ranking list based on the true user feedback) in the following.

Rating Prediction. We first compare DEAMER with other state-of-the-art rating prediction models on the rating prediction task.

Table 2. Results of rating prediction with best performance in bold

Data set	UKNN	IKNN	NMF	logit-vd	AutoRec	DeepCoNN	DEAMER
MI	1.0348	1.0073	0.9558	1.0578	0.9178	0.9244	**0.9017**
OP	0.9646	0.9914	0.9308	0.9904	0.9269	0.8783	**0.8696**
DM	1.0683	1.0829	0.9973	1.0714	1.0010	1.0319	**0.9533**
SO	1.0683	1.0908	1.0087	1.8648	0.9981	0.9870	**0.9668**
HP	1.1929	1.1990	1.1208	1.8786	1.1275	1.0876	**1.0839**

Baselines. We compare with the conventional collaborative filtering models, probabilistic MNAR models, and deep neural network models:

1. UKNN [1]: the user-based collaborative filtering with Pearson correlation and $k = 50$.
2. IKNN [1]: the item-based collaborative filtering with cosine similarity and $k = 50$.
3. NMF [13]: nonnegative matrix factorization.
4. logit-vd [26]: a collaborative filtering method that assumes responses are *Missing Not At Random*.
5. AutoRec [31]: a deep model using autoencoder.
6. DeepCoNN [37]: a deep model that contains two parallel CNNs to extract latent factors from both user and item reviews.

Evaluation Metrics. We use Root Mean Square Error (RMSE) for evaluation.

Observations. We report the results of rating prediction in Table 2. We can find that DEAMER produces the best results on different data sets. DEAMER outperforms the probabilistic MNAR generative model logit-vd by more than 10% on all data sets. This is unsurprising, since DEAMER's deep architecture allows it to express the dependency structure between interactions and ratings. In addition, DEAMER achieves significantly lower RMSE than deep neural network recommendation models AutoRec and DeepCoNN . This shows the benefits of multi-modal learning and deep generative model over normal deep models.

Top-k Recommendation. We then assess the performance of DEAMER on the ranking task.

Baselines. We compare DEAMER with a wide range of state-of-the-art baselines including both shallow and deep models:

1. MostPopular: a method which outputs the most popular items as the recommendation list.

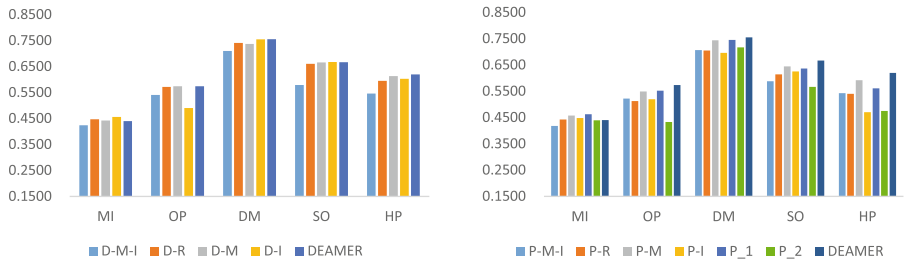
Table 3. Results of top-10 recommendation with best performance in bold

Methods		MostPopular	SVD++	FISM	NeuCF	DeepMF	ConvMF	CMN	DEAMER
MI	HR	0.3394	0.3471	0.3499	0.3478	0.3464	0.3674	0.3590	**0.4402**
	NDCG	0.1999	0.2081	0.2025	0.2058	0.2017	0.1718	0.2183	**0.2498**
	MRR	0.1571	0.1656	0.1575	0.1624	0.1573	0.1107	0.1749	**0.1924**
OP	HR	0.3321	0.4385	0.3382	0.4353	0.4112	0.4630	0.3823	**0.5739**
	NDCG	0.1725	0.2434	0.1741	0.2398	0.2280	0.2145	0.2082	**0.3350**
	MRR	0.1246	0.1842	0.1248	0.1805	0.1724	0.1374	0.1556	**0.2618**
DM	HR	0.3795	0.6979	0.3857	0.6694	0.6972	**0.7760**	0.7426	0.7553
	NDCG	0.2089	0.4525	0.2119	0.4044	0.4409	0.3956	**0.5279**	0.4813
	MRR	0.1569	0.3760	0.1591	0.3227	0.3615	0.2729	**0.4603**	0.3961
SP	HR	0.3911	0.5338	0.3957	0.4612	0.4344	0.5903	0.4722	**0.6668**
	NDCG	0.2262	0.3412	0.2284	0.2749	0.2448	0.2923	0.3057	**0.4222**
	MRR	0.1759	0.2816	0.1773	0.2176	0.1869	0.1974	0.2543	**0.3466**
HP	HR	0.3714	0.4992	0.3765	0.4427	0.4015	0.5121	0.4105	**0.6198**
	NDCG	0.2129	0.3172	0.2162	0.2675	0.2350	0.2557	0.2763	**0.4043**
	MRR	0.1647	0.2610	0.1674	0.2137	0.1841	0.1738	0.2349	**0.3376**

2. SVD++ [12]: a matrix factorization model that combines latent factor model and neighborhood model.
3. FISM [11]: an factorization matrix method that learns the item-item similarity of low dimensional latent factors.
4. NeuCF [9]: a deep collaborative filtering model that generalizes matrix factorization with multi-layer perceptron.
5. DeepMF [34]: a deep matrix factorization model that learns latent features of users and items using multi-layer perceptron.
6. ConvMF [8]: a deep model that uses an outer product to reconstruct the pairwise item correlations in the embedding space.
7. CMN [4]: a deep model that combines the global user and item embeddings and the local neighborhood-based structure with neural attention mechanism.

Evaluation Metrics. We adopt Hit Ratio (HR), Normalized Discounted Cumulative Gain (NDCG) and Mean Reciprocal Rank (MRR) as the metrics for the top-10 recommendation.

Observations. Table 3 illustrates the results of top-10 recommendation. From the results, we can conclude that DEAMER performs consistently well in terms of HR, NDCG and MRR, on different data sets. To be specific, it produces the best performances on four out of five data sets. On DM data set, although DEAMER does not produce the best results, it produces satisfying results (i.e., second best results) in terms of all evaluation metrics, while the best models ConvMF and CMN on DM data set do not perform best on other four data sets.



(a) Multi-modal (b) Multi-task

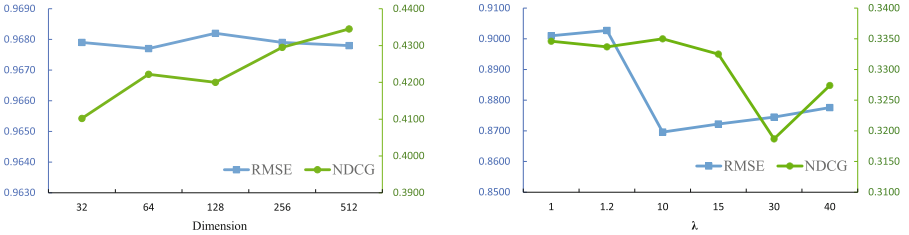
Fig. 3. Performance of DEAMER with different multi-modal and multi-task settings

4.3 Effectiveness of Multi-modal Learning and Multi-task Learning

We further conduct experiments to show the effectiveness of multi-modal learning and multi-task learning used in DEAMER.

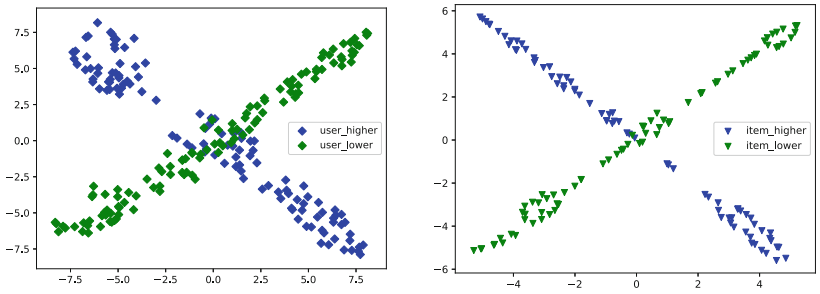
Figure 3(a) shows the impact of different modalities on the HR performance of DEAMER. User-item reviews are the major information source for content based recommendation and we denote the use of it as D. There are three additional modalities, namely item meta-data (M), item images (I) and user-item ratings (R). From Fig. 3(a), we can conclude that all the modalities contribute to the recommendation of DEAMER, as DEAMER using all modalities consistently performs better on all the data sets than the cases when some modalities are not used.

Furthermore, we compare DEAMER to an alternative multi-task learning model. DEAMER is a generative model which operates in a cascade manner. It is common to design a *parallel* multi-task learning model that uses the same representations to predict rating values and binary interactions. The parallel model implements an interaction layer and a rating prediction layer on a shared representation learning component. The representation component can be built on multi-modal data sources. Since the loss function of parallel model also consists of two parts, one is the square loss for rating values, the other is the cross-entropy loss for interactions. Similar to Eq. 11, the two parts are summated with λ. For a fair comparison, we use the same network structures. In Fig. 3(b), we compare the performance of the parallel alternative and DEAMER. The notations are similar as in Fig. 3(a). Additionally, P indicates the parallel architecture is used. For example, "P-I-M" uses LSTMs on user and item reviews, and the representations flow to a rating prediction layer and an interaction prediction layer parallely. We report the parallel multi-task model on all modalities in P_1 with $\lambda = 1.0$ for all data sets; and in P_2 with λ follows DEAMER's setting. From Fig. 3(b), we can observe that DEAMER outperform its parallel alternative which illustrates the power of its architecture.



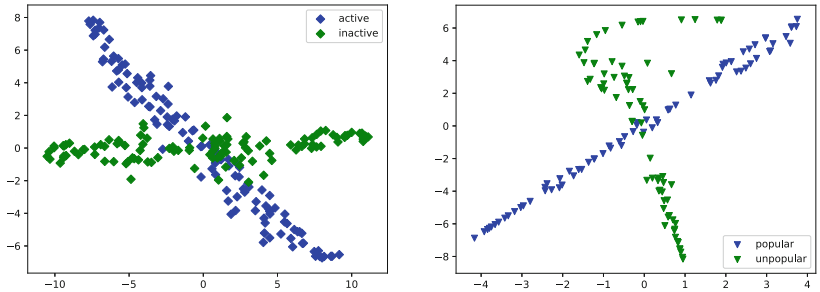
(a) Vary D_h on SO (b) Vary λ on OP

Fig. 4. Performance of DEAMER with different parameter settings



(a) User average rating (b) Item average rating

Fig. 5. Distinguishing user and item embeddings for users and items with different average ratings



(a) User activity (b) Item popularity

Fig. 6. Distinguishing user and item embeddings for different levels of user activity and item popularity

4.4 Impact of Hyper-parameters

In the following, we investigate how the hyper-parameters affect the performance of DEAMER. We report the change of RMSE and NDCG on the data set SO by varying $D_h \in \{32, 64, 128, 256, 512\}$ in Fig. 4(a) and on the data set OP by varying $\lambda \in \{1, 1.2, 10, 15, 30, 40\}$ in Fig. 4(b).

From Fig. 4(a), we can see that the dimensionality D_h of hidden states does not have a large impact on rating prediction. However, when D_h increases, the performance of top-k recommendation will be improved correspondingly. As the loss coefficient λ changes which is shown in Fig. 4(b), both RMSE and NDCG will be affected. The best overall performance of DEAMER can be achieved using $\lambda \in \{10, 15\}$ to balance rating prediction and top-k recommendation.

4.5 Visualization

After investigating the recommendation results of DEAMER, we discover that DEAMER is able to provide some insights on useful patterns of users and items.

We first compute the average rating for each user and item on data set MI and select 8% of the users/items with the highest average rating and 8% of the users/items with the lowest average rating. We project the embeddings of users and items into a two-dimensional surface with t-SNE [25] and plot the projection in Fig. 5. DEAMER clearly reveals user bias, i.e., a user tends to rate higher or lower in his/her preference, and item bias, i.e., items that are likely to receive higher or lower ratings.

We also plot the projected embeddings for 8% of the users/items with the most ratings and 8% of the users/items with the fewest ratings. Again, as depicted in Fig. 6, DEAMER can distinguish active users and inactive users, popular items and niche items.

5 Conclusion

In this work, we present DEAMER, a novel deep generative model that not only leans representations to generate the exposure, but also predict ratings simultaneously with the help of exposure. By using multi-modal learning and multi-task learning, DEAMER shows that it is possible to fully use heterogeneous information sources and achieve a better performance compared to the state-of-the-art approaches. In the future, we plan to explore the possibility of expanding the user modality used in DEAMER and making it more multivariate like the item modalities used in DEAMER. We also plan to replace the blocks for representation learning in DEAMER with other neural architectures to further improve its recommendation results.

Acknowledgement. This work was supported by the National Natural Science Foundation of China (no. 61702432, 61772209, 61972328), the Fundamental Research Funds for Central Universities of China (20720180070) and the international Cooperation Projects of Fujian in China (201810016).

References

1. Aggarwal, C.C.: Recommender Systems - The Textbook. Springer (2016)
2. Ding, D., Li, H., Huang, Z., Mamoulis, N.: Efficient fault-tolerant group recommendation using alpha-beta-core. In: CIKM. pp. 2047–2050 (2017)
3. Dong, X., Yu, L., Wu, Z., Sun, Y., Yuan, L., Zhang, F.: A hybrid collaborative filtering model with deep structure for recommender systems. In: AAAI. pp. 1309–1315 (2017)
4. Ebesu, T., Shen, B., Fang, Y.: Collaborative memory network for recommendation systems. In: SIGIR. pp. 515–524 (2018)
5. García-Durán, A., Gonzalez, R., Oñoro-Rubio, D., Niepert, M., Li, H.: Transrev: Modeling reviews as translations from users to items. arXiv Preprint (2018), https://arxiv.org/abs/1801.10095
6. Gatzioura, A., Vinagre, J., Jorge, A.M., Sànchez-Marrè, M.: A hybrid recommender system for improving automatic playlist continuation. In: IEEE Transactions on Knowledge and Data Engineering (2019). https://doi.org/10.1109/TKDE.2019.2952099
7. Guo, H., Tang, R., Ye, Y., Li, Z., He, X.: Deepfm: A factorization-machine based neural network for CTR prediction. In: IJCAI. pp. 1725–1731 (2017)
8. He, X., Du, X., Wang, X., Tian, F., Tang, J., Chua, T.: Outer product-based neural collaborative filtering. In: IJCAI. pp. 2227–2233 (2018)
9. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.: Neural collaborative filtering. In: WWW. pp. 173–182 (2017)
10. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997)
11. Kabbur, S., Ning, X., Karypis, G.: FISM: factored item similarity models for top-n recommender systems. In: KDD. pp. 659–667 (2013)
12. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: KDD. pp. 426–434 (2008)
13. Koren, Y., Bell, R.M., Volinsky, C.: Matrix factorization techniques for recommender systems. IEEE Computer **42**(8), 30–37 (2009)
14. Lau, J.H., Baldwin, T.: An empirical evaluation of doc2vec with practical insights into document embedding generation. In: Rep4NLP@ACL. pp. 78–86 (2016)
15. LeCun, Y., Haffner, P., Bottou, L., Bengio, Y.: Object recognition with gradient-based learning. In: Shape, Contour and Grouping in Computer Vision. vol. 1681, p. 319 (1999)
16. Li, H., Chan, T.N., Yiu, M.L., Mamoulis, N.: FEXIPRO: fast and exact inner product retrieval in recommender systems. In: SIGMOD Conference. pp. 835–850 (2017)
17. Li, H., Liu, Y., Mamoulis, N., Rosenblum, D.S.: Translation-based sequential recommendation for complex users on sparse data. IEEE Trans. Knowl, Data Eng (2019)
18. Li, H., Liu, Y., Qian, Y., Mamoulis, N., Tu, W., Cheung, D.W.: HHMF: hidden hierarchical matrix factorization for recommender systems. Data Min. Knowl. Discov. **33**(6), 1548–1582 (2019)
19. Li, H., Wu, D., Mamoulis, N.: A revisit to social network-based recommender systems. In: SIGIR. pp. 1239–1242 (2014)
20. Li, H., Wu, D., Tang, W., Mamoulis, N.: Overlapping community regularization for rating prediction in social recommender systems. In: RecSys. pp. 27–34 (2015)

21. Lian, J., Zhang, F., Xie, X., Sun, G.: Cccfnet: A content-boosted collaborative filtering neural network for cross domain recommender systems. In: WWW. pp. 817–818 (2017)
22. Lian, J., Zhou, X., Zhang, F., Chen, Z., Xie, X., Sun, G.: xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In: KDD. pp. 1754–1763 (2018)
23. Liang, D., Charlin, L., McInerney, J., Blei, D.M.: Modeling user exposure in recommendation. In: WWW. pp. 951–961 (2016)
24. Lu, Z., Li, H., Mamoulis, N., Cheung, D.W.: HBGG: a hierarchical bayesian geographical model for group recommendation. In: SDM. pp. 372–380 (2017)
25. van der Maaten, L., Hinton, G.: Visualizing data using t-sne. JMLR **9**, 2579–2625 (2008)
26. Marlin, B.M., Zemel, R.S.: Collaborative prediction and ranking with non-random missing data. In: RecSys. pp. 5–12 (2009)
27. Qian, Y., Li, H., Mamoulis, N., Liu, Y., Cheung, D.W.: Reverse k-ranks queries on large graphs. In: EDBT. pp. 37–48 (2017)
28. Rendle, S.: Factorization machines with libfm. ACM TIST 3(3), 57:1–57:22 (2012)
29. Ricci, F., Rokach, L., Shapira, B. (eds.): Recommender Systems Handbook. Springer (2015)
30. Salakhutdinov, R., Mnih, A.: Probabilistic matrix factorization. In: NIPS. pp. 1257–1264 (2007)
31. Sedhain, S., Menon, A.K., Sanner, S., Xie, L.: Autorec: Autoencoders meet collaborative filtering. In: WWW. pp. 111–112 (2015)
32. Strub, F., Gaudel, R., Mary, J.: Hybrid recommender system based on autoencoders. In: DLRS@RecSys. pp. 11–16 (2016)
33. Wang, C., Niepert, M., Li, H.: LRMM: learning to recommend with missing modalities. In: EMNLP. pp. 3360–3370 (2018)
34. Xue, H., Dai, X., Zhang, J., Huang, S., Chen, J.: Deep matrix factorization models for recommender systems. In: IJCAI. pp. 3203–3209 (2017)
35. Zhang, S., Yao, L., Sun, A., Tay, Y.: Deep learning based recommender system: A survey and new perspectives. ACM Comput. Surv. 52(1), 5:1–5:38 (2019)
36. Zhang, Y., Ai, Q., Chen, X., Croft, W.B.: Joint representation learning for top-n recommendation with heterogeneous information sources. In: CIKM. pp. 1449–1458 (2017)
37. Zheng, L., Noroozi, V., Yu, P.S.: Joint deep modeling of users and items using reviews for recommendation. In: WSDM. pp. 425–434 (2017)