Neurocomputing 421 (2021) 105–114

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

AIRec: Attentive intersection model for tag-aware recommendation

Bo Chen [a,1], Yue Ding [a,1], Xin Xin [b], Yunzhe Li [a], Yule Wang [a], Dong Wang [a,*]

[a] School of Software, Shanghai Jiao Tong University, Shanghai, China
[b] School of Computing Science, University of Glasgow, Glasgow, UK

ARTICLE INFO

Article history:
Received 2 May 2020
Revised 27 July 2020
Accepted 18 August 2020
Available online 11 September 2020
Communicated by H. Zhang

Keywords:
Tag-aware collaborative filtering
Neural networks
Attention mechanism

ABSTRACT

Tag-aware recommender systems (TRS) utilize rich tagging information to better depict user portraits and item features. Recently, many efforts have been done to improve TRS with neural networks. However, existing methods construct user representations through either explicit tagging behaviors or implicit interacted items, which is inadequate to capture multi-aspect user preferences. Besides, there are still lacks of investigation about the intersection between user and item tags, which is crucial for better recommendation.

In this paper, we propose AIRec, an attentive intersection model for TRS, to address the above issues. More precisely, we first project the sparse tag vectors into a latent space through multi-layer perceptron (MLP). Then, the user representations are constructed with a hierarchical attention network, where the item-level attention differentiates the contributions of interacted items and the preference-level attention discriminates the saliencies between explicit and implicit preferences. After that, the intersection between user and item tags is exploited to enhance the learning of conjunct features. Finally, the user and item representations are concatenated and fed to factorization machines (FM) for score prediction. We conduct extensive experiments on two real-world datasets, demonstrating significant improvements of AIRec over state-of-the-art methods for tag-aware top-n recommendation.

1. Introduction

Bringing user interest to item content can improve the quality of recommendation in recommender systems [1]. The reason that a user like a specific item can be attributed to only a few set of explicit or implicit features. Tags can directly associate users' preferences with item attributes. Social tagging systems, also known as folksonomies, are widely used in various websites and applications, where users can freely annotate online items (e.g., movies, artists) with arbitrary tags [2]. Generally, these tags are composed by laconic words or phrases that express users' interests in the item. Benefit from the pithiness, tags are more direct and objective than reviews, which can not only indicate user preferences [3], but also summarize features of items [4,5]. Consequently, user-defined tags can be introduced into recommender systems for alleviating the cold-start problem [6] and improving recommendation quality.

However, as users can freely choose their own vocabulary, social tags may contain many rare or even user-created vocabularies, resulting in sparse, redundant and ambiguous tag space [7]. Many efforts have been made to solve this issue. Except for the traditional solutions (such as methods based on vector space model (VSM) [8], clustering-based methods [7], graph-based methods [9], etc), deep learning brightens a new way. Neural networks-based methods extract abstract representations by converting the sparse tag space into dense latent space, such as CFA [10], DSPR-NS [11] and TRSDL [12]. Although these models have made some progress, there are three weaknesses that hinder their performance: (1) They construct user representations by either explicit tagging behaviors (CFA and DSPR-NS) or implicit interacted items (TRSDL), which is inadequate to capture multi-aspect user preferences. (2) The intersection of user and item tags reflects the diverse focuses of different users on the same item, which is the key incentive of user-item transactions (shown in Fig. 2). Unfortunately, seldom research has explored this field, as far as we know. (3) They calculate the final predicting scores by cosine similarity or inner product, which regards latent dimensions as independent, incurring significant loss of interaction information and sub-optimal performance.

In this paper, we focus on developing methods to address the above-mentioned drawbacks of existing solutions and propose an Attentive Intersection Recommendation model (AIRec) for TRS.

* Corresponding author.
E-mail addresses: chenbo.31@sjtu.edu.cn (B. Chen), dingyue@sjtu.edu.cn (Y. Ding), x.xin.1@research.gla.ac.uk (X. Xin), liyzh28@sjtu.edu.cn (Y. Li), wyl666@sjtu.edu.cn (Y. Wang), wangdong@sjtu.edu.cn (D. Wang).
[1] The co-first authors Bo Chen and Yue Ding contribute equally.

https://doi.org/10.1016/j.neucom.2020.08.018
0925-2312/

Specifically, we first leverage MLPs with shared parameters to map tag-based sparse user and item feature vectors into a dense latent space. Then, for capturing multi-aspect user preferences, we propose a **hybrid user model** with a hierarchical attention network, on which the item-level attention aims to depict user implicit preferences by differentiating contributions of interacted items and the preference-level attention dynamically discriminates the saliencies between explicit tagging behaviors and implicit preferences. Besides, we design an **intersection module** which exploits the intersection of user and item tags to further train the MLP and enhance the learning of conjunct features. Finally, the concatenated user and item representations are fed into the FM to learn the second-order interactions and predict final scores. Compared to other models in TRS, our method not only captures more accurate user portrait via hierarchical attention network, but also achieves better score prediction and makes full use of the tags intersection to improve performance.

The main contributions of this paper are summarized as follows:

- We propose a novel tag-aware recommendation model AIRec, which combines the strong expression capability of neural attention and the second-order interaction modeling of factorization machine.
- We design a hybrid user model with a hierarchical attention network, which unifies both explicit tagging behaviors and implicit user preferences for capturing multi-aspect user representations.
- We discover the importance of tags intersection in TRS and innovatively employ it to enhance the learning of conjunct features. The extensive experiments on two real-world datasets verify the superiority of our proposed model compared with state-of-the-art baseline methods.

The rest of this paper is organized as follows. Section 2 presents the related work. Section 3 demonstrates the detailed designs of our proposed AIRec model. Experiment and evaluation are described in Section 4. Finally, we conclude our work in Section 5.

2. Related work

2.1. Tag-aware recommender systems

Extensive works in TRS incorporate social tagging information into traditional collaborative filtering (CF) or content-based filtering (CBF) for personalized recommendation. TagiCoFi [13] extends the model-based CF to TRS by employing tagging information to regularize the matrix factorization (MF) procedure. Collaborative tagging system [14] projects the ternary relation of user-item-tag into user-item and user-tag, and then applies CF to perform ranking. TRCF [4] uses latent Dirichlet allocation (LDA) to mine semantic information of tags and incorporates it into probabilistic MF for factorizing rating information. De Gemmis et al. [15] present a content-based filtering algorithm by leveraging both the textual description of items and tags for modeling user preferences.

To solve the problem of sparsity, ambiguity and redundancy in tag space, a variety of solutions have been proposed, such as VSM-based methods, clustering-based methods, graph-based methods, etc. Cantador et al. [8] leverage various content-based model to map the tag-based user/item profiles into weighted vector space, and thereafter defines the corresponding similarity metrics. Shepitsen et al. [7] design a personalization recommendation model in folksonomies using hierarchical tag clustering. However, clustering-based approaches are usually time-consuming, so Xu et al. [16] develop a lightweight tag-aware recommender system using ontological similarity for solving ambiguity problem and measuring the semantic relevance. Besides, Zhang et al. [9] propose a recommendation algorithm based on an integrated diffusion on user-item-tag tripartite graphs.

Recent researches show impressive performance by using neural network-based methods to overcome the above problems in TRS. CFA [10] uses the stacked sparse autoencoder (SSAE) to obtain abstract representations of tag-based user profiles and combines user-based CF for recommendation. DSPR-NS [11] leverages the MLP to map the user and item tags into an abstract feature space and maximizes deep-semantic similarities between user and relevant items. Based on this model, HDLPR-NS [17] uses the autoencoder (AE) with reconstruction errors for further accelerating the learning progress. As for rating prediction, TRSDL [12] utilizes deep neural network (DNN) and recurrent neural network (RNN) to extract latent representations of items and users respectively, on which MF is applied. Besides, tags can be introduced into recommender systems as a kind of superior side information and combined with neural networks to make recommendation [18,19].

2.2. Recommender systems based on neural attention

Neural networks have a wide range of applications in various fields, such as image recognition, machine translation and recommender system. Due to the powerful representation capability, neural networks are extensively applied in automatic feature extraction, emerging plenty of recommendation models [20,21]. Lately, the neural attention models integrate attention mechanisms into neural networks (e.g., MLP, CNN and RNN) for identifying the importance of different parts, which has attracted a considerable amount of interest to enhance personalized recommendation. AFM [22] improves the original FM by using the attention mechanism to discriminate the importance of feature interactions. ACF [23] employs attention modeling in CF to address the challenging item- and component-level implicit feedback in multimedia recommendation. Seo et al. [24] model user preferences and item properties using convolutional neural networks (CNN) with local and global attention networks. DIN[25] proposes an activation unit that shares similar intuition with attention mechanism for click-through rate (CTR) prediction. TNAM [26] utilizes user-item interactions with tag information to construct dense input, and then combines the attention network, stacked autoencoder and MLP for making prediction.

2.3. Summary

The above works have achieved excellent results, which inspires us to incorporate deep learning and neural attention mechanism into TRS. We highlight the features of our work in difference with existing methods in Table 1.

3. The AIRec model

In this section, we first describe the architecture of our proposed AIRec model and then explain the detailed training procedure. Fig. 1 illustrates the structure of the AIRec model. The target of AIRec is to generate a ranked item list for a given user described by a tag-based feature vector, also known as top-n recommendation.

3.1. Input layer and hidden layers

3.1.1. Input layer

Before diving into the technical details, we first introduce some basic notations. Throughout the paper, we use bold uppercase

Table 1
The features of our work in difference with existing methods. Columns represent the technologies that may be used in these methods.

	Deep learning	Neural Attention	FM for prediction	Tag Intersection
TagiCoFi [13]	No	No	No	No
TRCF [4]	No	No	No	No
CFA [10]	Yes	No	No	No
DSPR-NS [11]	Yes	No	No	No
TRSDL [12]	Yes	No	No	No
AFM [22]	Yes	Yes	Yes	No
HDLPR-NS [17]	Yes	No	No	No
TNAM [26]	Yes	Yes	No	Yes
AIRec(Ours)	Yes	Yes	Yes	Yes

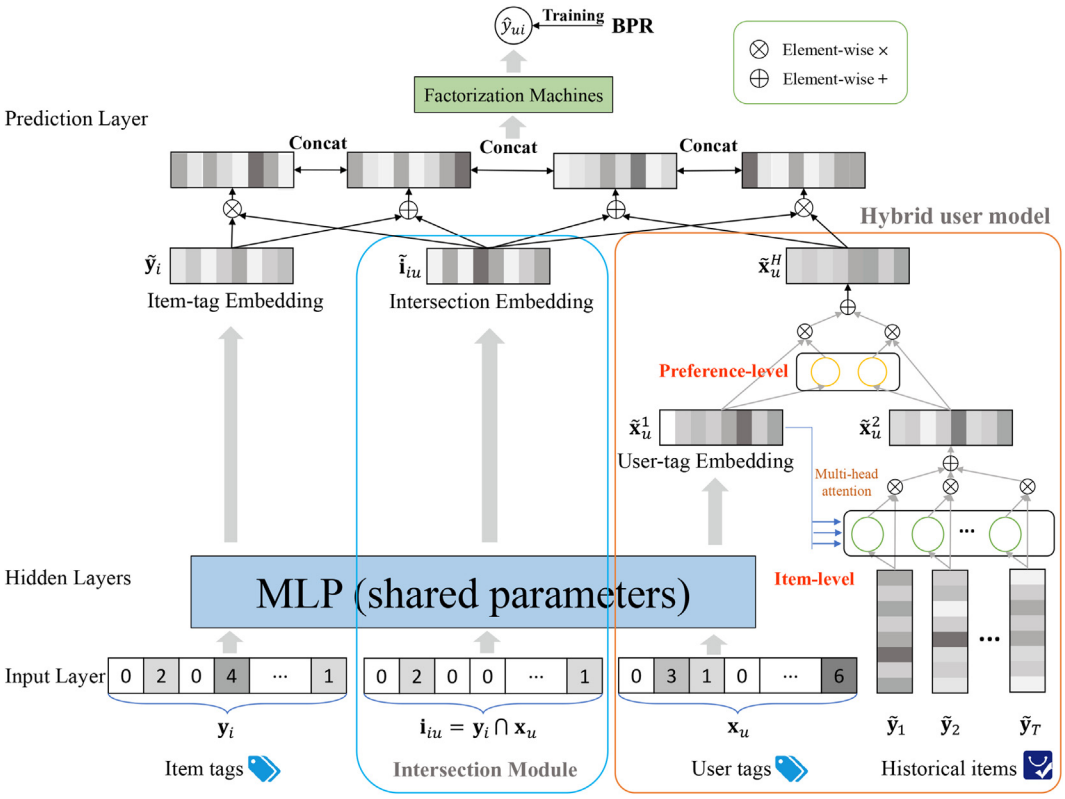


Fig. 1. The structure of AIRec model. The input layer transforms the raw tagging profiles of users or items into tag-based feature vectors. The hidden layer is a multi-layer perceptron with shared parameters to extract dense representations. User hybrid representation is calculated by the hierarchical structure with item-level attention and preference-level attention. The dense vectors of item-tag representation, item-user intersection representation and user hybrid representation are concatenated, and then be fed into the prediction layer using factorization machines. The training process is under the Bayesian Personalized Ranking (BPR) framework.

letter (e.g., $\mathbf{W}$) to denote a matrix and bold lowercase letter to denote a vector (e.g., $\mathbf{x}c$). Scalar is represented by lowercase letters (e.g., x). A folksonomy is a tuple $\mathcal{F} = (U, I, T, A)$ [2], where U, I and T are sets of users, items and tags, respectively, and $A \subseteq U \times I \times T$ is a set of assignments (u, i, t) that user u annotates tag t to item i.

The input layer aims to transform the raw tagging profiles of users/items into tag-based feature vectors. In detail, the profile of user u is represented by a sparse feature vector $\mathbf{x}_u = (p_1^u, p_2^u, \ldots, p_V^u)$, where $V = |T|$ is the size of tag vocabularies and $p_j^u = \left|\{(u, i, t_j) \in A | i \in I\}\right|$ is the number of times that user u annotates items with tag t_j. Similarly, The profile of item i can be represented as $\mathbf{y}_i = (q_1^i, q_2^i, \ldots, q_V^i)$, where $q_j^i = \left|\{(u, i, t_j) \in A | u \in U\}\right|$ is the number of times that item i is annotated with tag t_j by users. It is worth mentioning that we take into account the times that tags have been annotated since the frequencies of tags can reflect the degree of user interest and item characteristic [8,5].

3.1.2. Hidden layers

In order to solve the problem of sparsity and high-dimension in tag space, feature vectors $\mathbf{y}_i$ and $\mathbf{x}_u$ are fed into the multi-layer perceptron. Multiple stacked fully-connected layers compress highly sparse vectors into an abstract low-dimensional feature space. Here the trainable parameters of all layers in MLP are shared parameters, which means we train only one MLP for all types of inputs. The motivations of sharing parameters are twofold: (1) Since the user and item feature vectors share the same tag space, it is reasonable to share parameters among networks for better generalization capability and less computational overhead. (2) Shared parameters force neural networks to use the same abstract feature space to describe user and item, which is conducive to the subsequent modeling.

We take the user feature vector $\mathbf{x}_u$ as an example. $\mathbf{x}_u$ is fed into MLP and the output $\mathbf{h}_1$ of the first hidden layer is formulated as:

$$\mathbf{h}_1(u) = tanh(\mathbf{W}_1 \mathbf{x}_u + \mathbf{b}_1). \quad (1)$$

Similarly, the output of the jth layer $\mathbf{h}_j$ is defined as:

$$\mathbf{h}_j(u) = tanh(\mathbf{W}_j\mathbf{h}_{j-1}(u) + \mathbf{b}_j), \tag{2}$$

where $\mathbf{W}_j$ and $\mathbf{b}_j$ are the weight matrix and the bias vector for the jth layer, respectively. And we take Hyperbolic Tangent (Tanh) as the activation function.

As a result, the output of the L-layer neural network is represented as $\tilde{\mathbf{x}}_u^1 = \mathbf{h}_L(u)$, which is the latent representation of user u. Analogously, the latent representation of item i can be obtained by $\tilde{\mathbf{y}}_i = \mathbf{h}_L(i)$.

3.2. Hybrid user model

As mentioned above, the MLP exploits the tags annotated by user u for constructing user latent representation, which can be viewed as explicit preferences $\tilde{\mathbf{x}}_u^1$. However, in most cases, users tend to pick a single tag to describe an item for convenience, but that doesn't mean they only focus on this aspect. Since the latent representation of each item is constructed based on the social tagging information contributed by all users, the historical items that user u has interacted with can be leveraged to depict implicit preferences. Consequently, users' subjective tagging behaviors reflect their explicit preferences while the interacted historical items mirror their implicit preferences. In this part, we elaborate a hybrid user model with a hierarchical attention network that combines explicit and implicit preferences for capturing multi-aspect user representations.

3.2.1. Item-level attention

For seizing implicit preferences, an intuitive approach is to treat all historical items equally and obtain an average-pooling representation directly. We argue that it lacks of emphasis because different items contribute differently when modeling user profiles. Therefore, we leverage an additive attention network to alleviate the above-mentioned shortcoming. The attention network studies the similarity between user's historical rated items and user's explicit preferences $\tilde{\mathbf{x}}_u^1$, and then assembles the informative item representations with different weights to characterize implicit preferences.

Denote the historical item set of user u is $\mathcal{I}_u$, the kth item $i_k \in \mathcal{I}_u$ is represented by a dense vector $\tilde{\mathbf{y}}_k$ as described in Section 3.1.2. In order to describe user's multi-aspect representation, we apply multi-head attention mechanism to learn joint attention information from different subspaces, and then we concatenate all the multi-head output values and make projection again, resulting in the final attention score. User u's multi-head attention score for item k is defined as Eq. (3).

$$a(u,k)_{MH} = \mathbf{v}_1^T relu(\text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_\text{h})), \tag{3}$$

here for each $\text{head}_{\text{i}\in[1,\text{h}]}$,

$$\text{head}_\text{i}(\text{u},\text{k}) = \mathbf{W}_\mathbf{i}^0\tilde{\mathbf{x}}_u^1 + \mathbf{W}_\mathbf{i}^1\tilde{\mathbf{y}}_k + \mathbf{b}_\mathbf{i}^1, \tag{4}$$

The projection matrices $\mathbf{W}_\mathbf{i}^0$, $\mathbf{W}_\mathbf{i}^1$ and bias $\mathbf{b}_\mathbf{i}^1$ are the first layer parameters of head_i which project the item representation into a hidden state under the supervision of user's explicit preferences. We utilize the Rectified Linear Unit (ReLU) as the activation function to enhance nonlinear capability and speed up convergence. The vector $\mathbf{v}_1^T$ is the parameter of the second layer that projects the hidden state into the attention score, where the size of hidden state is termed as attention factor.

The attention weights $\alpha(u,k)$ are obtained by normalizing the attention scores using Softmax, which can be interpreted as the contribution of the kth item to the implicit preferences. It is worthy noting that the impacts of the same item vary from user to user, since the leaning procedure is guided by their explicit preferences. The normalized attention weight $\alpha(u,k)$ is computed as:

$$\alpha(u,k) = softmax(a(u,k)_{MH}) = \frac{exp(a(u,k))}{\sum_{i_{k'}\in\mathcal{I}_u} exp(a(u,k'))}. \tag{5}$$

Finally, the implicit preferences $\tilde{\mathbf{x}}_u^2$ can be represented as a sum of the item feature vectors weighted by the attention weights, which is shown as follows:

$$\tilde{\mathbf{x}}_u^2 = \sum_{i_k\in\mathcal{I}_u} \alpha(u,k)\tilde{\mathbf{y}}_k. \tag{6}$$

It worth to be mentioned that in practice we find that using a single head attention gets slightly better results that multi-heads.

3.2.2. Preference-level attention

After obtaining the implicit preferences $\tilde{\mathbf{x}}_u^2$, the hybrid user representation can be constructed by fusing $\tilde{\mathbf{x}}_u^2$ with explicit preferences $\tilde{\mathbf{x}}_u^1$ so as to keep consistent with the dimension of item representation. A simple but common way is to set hyper-parameter β for determining the trade-off between explicit and implicit preferences, shown as:

$$\tilde{\mathbf{x}}_u^H = \beta\tilde{\mathbf{x}}_u^1 + (1-\beta)\tilde{\mathbf{x}}_u^2. \tag{7}$$

However, this approach has two drawbacks. (1) It is difficult to tune hyper-parameter β manually; (2) Due to the individual diversity, fixing a uniform parameter for all users is sub-optimal. Accordingly, we design a self-attentive fusion mechanism for overcoming these drawbacks and discriminating the saliency for each user. The self-attention mechanism has been successfully applied in many tasks, such as sentence embedding [27] and question encoding [28], to compute the importance of each part.

Similarly, the attention score $b(u,k)$ of the kth part is calculated via a two-layer network, which is defined as:

$$b(u,k) = \mathbf{v}_2^T relu(\mathbf{W}^2\tilde{\mathbf{x}}_u^k + \mathbf{b}^2), \tag{8}$$

where $\mathbf{W}^2$, $\mathbf{b}^2$ and $\mathbf{v}_2^T$ are the trainable parameters of the self-attention network. The normalized attention weight $\beta(u,k)$ is computed as:

$$\beta(u,k) = softmax(b(u,k)) = \frac{exp(b(u,k))}{\sum_{k'=1}^2 exp(b(u,k'))}. \tag{9}$$

Finally, the hybrid user representation $\tilde{\mathbf{x}}_u^H$ is formulated as Eq. (10).

$$\tilde{\mathbf{x}}_u^H = \beta(u,1)\tilde{\mathbf{x}}_u^1 + \beta(u,2)\tilde{\mathbf{x}}_u^2. \tag{10}$$

To conclude, the hybrid user model has the following advantages: (1) It considers not only the explicit preferences reflected by user's own tagging behaviors, but also the implicit preferences conveyed by the historical items. Therefore, even for users with less tagging behaviors, their preferences can still be well depicted. (2) To capture the implicit preferences, the item-level additive attention network automatically assigns different contributions (weights) to the historical items under the supervision of user's explicit preferences, which is more efficient than max-pooling or average-pooling. (3) The preference-level self-attentive fusing mechanism effectively balances the trade-off between explicit and implicit preferences for a specific user, obtaining a superior user representation compared with the traditional manual settings.

3.3. Intersection module

As is well-known, item features are multi-dimensional. For example, a movie has the attributes of director, plot, theme, actors, etc. However, these features have diverse attractions for different

users. Fig. 2 provides a toy example of the intersection. ***John*** likes comedy, robots, cars and dogs. The tagging information of movie ***Transformers*** are robots, science fiction, cars and Michael Bay. Therefore, from ***John***'s perspective, features "robots" and "cars" are the major attractions of ***Transformers***, which are the vital dimensions when modeling this transaction. If tags are regarded as bridge linking user and item, the intersection of tags is the bridge pier undoubtedly.

Motivated by the above observation, we elaborate an intersection module to extract the intersection of user and item tags for further enhancing the recommendation performance. Firstly, we calculate the tags intersection of item i and user u by $\mathbf{i}_{iu} = \mathbf{y}_i \cap \mathbf{x}_u = (r_1^{iu}, r_2^{iu}, \ldots, r_V^{iu})$, where $r_j^{iu} = min(q_j^i, p_j^u)$ means the minimum occurrences of tag t_j. Here, we also take the times of tags into account. Then the intersection vector $\mathbf{i}_{iu}$ is fed into a MLP that shares parameters with the previous MLPs for further training the networks and obtaining latent representation $\tilde{\mathbf{i}}_{iu}$. At last, we introduce $\tilde{\mathbf{i}}_{iu}$ to obtain final user/item representations. In order to obtain a richer high-level feature expression, we get the following feature vectors, namely, $\tilde{\mathbf{y}}_i^a = \tilde{\mathbf{y}}_i \oplus \tilde{\mathbf{i}}_{iu}, \tilde{\mathbf{y}}_i^p = \tilde{\mathbf{y}}_i \otimes \tilde{\mathbf{i}}_{iu}, \tilde{\mathbf{x}}_u^{H_a} = \tilde{\mathbf{x}}_u^H \oplus \tilde{\mathbf{i}}_{iu}, \tilde{\mathbf{x}}_u^{H_p} = \tilde{\mathbf{x}}_u^H \otimes \tilde{\mathbf{i}}_{iu}$, where operation $\oplus$ means element-wise addition and $\otimes$ means element-wise product operation.

We can see from Fig. 2 that the intersection between user and item tags reveals the deep reason of user behaviors and reflects direct motivation of the user-item transaction. Due to the shared parameters, intersection module can constrain multi-layer perceptrons to focus on the conjunct features when modeling user and item representations. Besides, by introducing tags intersection, we can enhance the importance of the conjunct features in the final representations, obtaining more concrete user/item representations under a certain user-item transaction scenario. It is worth to be mentioned that in practice we find that using $\tilde{\mathbf{y}}_i^a$ and $\tilde{\mathbf{x}}_u^{H_a}$ as combined features for prediction can achieve the best results.

3.4. Prediction layer

After obtaining the final representations, feature vectors $\tilde{\mathbf{y}}_i$ and $\tilde{\mathbf{x}}_u^H$ are concatenated into a single vector $\mathbf{z} = [\tilde{\mathbf{y}}_i, \tilde{\mathbf{x}}_u^H]$. In order to predict score, $\mathbf{z}$ is passed through a prediction layer consisting of a factorization machine [29] which estimates the target by modeling all second-order interactions. The final predicting score is formulated as Eq. (11):

$$\hat{y}_{ui}(\mathbf{z}) = w_0 + \sum_{i=1}^{n} w_i z_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} z_i z_j \cdot \langle \mathbf{v_i}, \mathbf{v_j} \rangle, \tag{11}$$

where w_0 represents the global bias, w_i represents the bias factor for the ith dimension. The pairwise interaction of dimension z_i and z_j is captured by a factorized parametrization $\langle \mathbf{v_i}, \mathbf{v_j} \rangle = \sum_{f=1}^{k} v_{if} v_{jf}$, where $\langle \cdot, \cdot \rangle$ denotes the inner product of two vectors of size k.

The major advantage of using FM for prediction is that it can capture the second-order interactions in a fine-grained manner both within and between $\tilde{\mathbf{y}}_i^a$ and $\tilde{\mathbf{x}}_u^{H_a}$ (after concatenated). Consequently, FM captures more interaction signals between latent dimensions compared with cosine similarity and inner product, which assumes that latent dimensions are independent with each other. To apply FM for ranking, PRFM [30] proposes a pairwise ranking factorization machines which combines pairwise learning to ranking (LTR) with original FM. We follow their core ideas and apply FM in ranking with negative sampling.

3.5. Training details

The focus of AIRec is generating top-n recommendation other than rating prediction. As a result, we optimize the proposed AIRec model with the BPR framework [31], which is more effective to perform item ranking. The loss function is shown in Eq. (12).

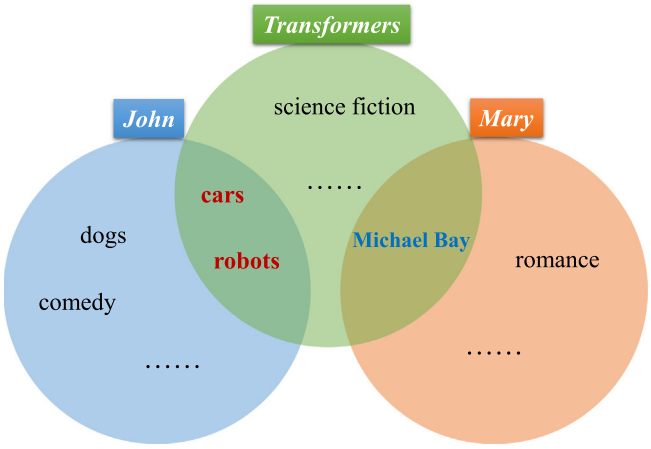


Fig. 2. The Intersection of Tags (Toy Example). ***John*** likes comedy, robots, cars and dogs. The tagging information of movie ***Transformers*** are robots, science fiction, cars and Michael Bay. Therefore, from ***John***'s perspective, features "robots" and "cars" are the major attractions of ***Transformers***, which are the vital dimensions when modeling this transaction.

$$L_\Theta = \sum_{\langle u, i_+, i_- \rangle} - \ln \sigma(\hat{y}_{ui_+} - \hat{y}_{ui_-}), \tag{12}$$

where i_+ and i_- are the positive and negative items of user u respectively, and $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function.

More specifically, we first sample a mini-batch of positive (observed) assignments (u, i_+). Thereafter, for each positive assignment, negative item is randomly sampled from a uniform distribution and negative assignment (u, i_-) is structured. Finally, both positive and negative assignments are fed to train model and minimize the loss function defined in Eq. (12). Besides, dropout [32] is also used to prevent overfitting. Recent works have demonstrated that an adaptive sampling distribution would result in better performance [33] and we leave this exploration as future work.

4. Experiments

In this section, we conduct experiments with the aim of answering the following research questions:

- **RQ1** Does AIRec model outperform state-of-the-art TRS methods for top-n recommendation?
- **RQ2** How does the hybrid user model affect the model performance?
- **RQ3** Can the intersection and FM components in AIRec improve the performance of recommendation?

4.1. Experimental settings

4.1.1. Datasets

To evaluate the performance of the proposed AIRec, we conduct comprehensive experiments on two public real-world datasets: Last.Fm and Delicious, which are both released in HetRec 2011 [34].

- **Last.Fm** is a music recommendation dataset obtained from Last.Fm online music system,[2] where users are allowed to tag artists with various tags. In this dataset, each user has a list of tag assignments to artists, and we only take artists as items to be recommended.
- **Delicious** dataset is gathered from a popular system, Del.icio.us,[3] which encourages users to tag bookmarks. In this dataset, we take web bookmarks as items to be recommended.

For fair comparison, we adopt the same preprocessing as [10,11,17] to remove infrequent tags that are used less than 5 times in Last.Fm and 15 times in Delicious respectively. Table 2 summarizes the statistics of these datasets. For each dataset, we randomly select 80% of the assignments as training set and 20% as test set, which is consistent with the state-of-the-art TRS studies [10,11,17]. The assignments in the training set are used to construct tag-based user and item profiles.

4.1.2. Evaluation protocols

In order to evaluate the experimental results in all aspects, the top-n recommendation quality is evaluated by four metrics: Precision (P), Recall (R), F1-score (F) and Mean Reciprocal Rank (MRR). $P@k$ and $R@k$ measure the ratio of correctly predicted target items in the recommendation list to the total predicted items and all target items in test set, respectively. $F@k$[35] is the weighted average of $P@k$ and $R@k$. More convincingly, we also involve MRR [36], a vital metric in top-n recommendation, which assigns higher scores to the top-ranked items.

4.1.3. Parameter settings and baselines

We train AIRec by optimizing the BPR loss with mini-batch Adagrad [37]. The Adagrad optimization method is very suitable for processing sparse data. The Adagrad can automatically adjust the learning rate during training, using a larger learning rate update for parameters with a lower frequency, while using a smaller learning rate update for parameters with a higher frequency. To fairly compare the performance of all the models, we use a grid search strategy to find the optimal hyperparameters. The learning rate is searched in [0.01, 0.02, 0.05]. We set the maximum number of iterations to 500 to ensure convergence. In order to balance the training variance and computational cost, the batch size is set to 256 which is searched in [32]. In order to prevent overfitting in deep learning based models, we turn the dropout ratio in [0.1, 0.2, $\cdots$, 0.9], and we find the result is better if the dropout ratio is between [0.4,0.6]. As stated in DeepFM [38], it is recommended to use a 3-layer MLP with 400–200 neurons in each layer. Following DeepFM, we adopt a MLP with three hidden layers and the neurons in each layer are set to 512, 256 and 128, respectively. Following FM [29], the size of latent factors in FM is usually set to 5 or 10, here we set it to 8. All the parameters are initialized with a Gaussian distribution with a mean of 0 and a variance of 1. For other parameter settings in the baseline models, we refer to the settings suggested in the original papers.

We compare the performance of AIRec with the following baselines:

- **PopRank**: PopRank recommends the top-n most popular items, which is always taken as a benchmark for top-n recommender systems.
- **FM**: The original factorization machine [29] trained by BPR loss [31]. Here we also use a uniform sampling distribution to sample negative examples.
- **CFA**: CFA uses an autoencoder to obtain latent representations of user profiles, on which user-based CF is applied for recommendation [10].
- **DSPR-NS**: A deep-semantic similarity-based model [11] that compares the cosine similarity between users and items in a latent feature space.
- **HDLPR-NS**: A hybrid deep learning based approach [17] that projects tag-based user and item profiles into a latent feature space, and maximizes deep semantic user-item similarities for personalized recommendation.
- **TRSDL**: A hybrid deep learning model that combines DNN and RNN for rating prediction [12]. This is the state-of-the-art TRS model for rating prediction and we migrate it to top-n recommendation.
- **TNAM**: A tag-aware neural attention model [26] that combines deep learning and attention mechanism to make item recommendation. This is the state-of-the-art TRS model for top-n recommendation.

4.2. Performance comparison (RQ1)

Fig. 3 illustrates the top-n recommendation performances on two datasets in terms of $P@k, R@k, F@k$ and MRR. It's obvious that AIRec achieves the best performance in all metrics. The MRRs on Last.Fm and Delicious are 80.21% and 13.47% better than the state-of-the-art methods TNSM and DSPR-NS respectively, and the improvements in other metrics are also similar. We argue that the superior performance improvement lies in the following basements: (1) The hybrid user model takes explicit and implicit preferences into consideration and the hierarchical attention network models multi-aspect user representations accurately. (2) The elaborately designed intersection module constrains neural networks to focus on the conjunct features and enhances the performance of our model. (3) The involved FM in prediction layer captures the second-order interactions, which is more effective compared with cosine similarity and inner product.

Among the baselines, we can see that DSPR-NS achieves better results than CFA where the recommendation is only based on user profiles. Besides, FM directly uses the sparse tag-based vectors as input, achieving inferior performance in comparison to our model, which demonstrates the strong expressiveness of the neural attention mechanism.

Another interesting phenomenon is that TNAM performs well on the more sparse delicious dataset, it even performs better than our proposed AIRec method on the two evaluation metrics of Recall (R) and F1-score(F). We can infer an important conclusion from this, that is, using the intersection module can greatly improve the recommendation results in TRS.

4.3. Impact of the hybrid user model (RQ2)

4.3.1. Effect of the item-level attention

We now focus on analyzing the performance of the item-level attention network. To evaluate the effectiveness of the elaborate attentive-pooling implicit preferences modeling, we conduct experiments by replacing the item-level attention network with mean-pooling and max-pooling, respectively. We can see from Table 3 and Table 4 that mean-pooling achieves better performance than max-pooling because max-pooling only considers the most important feature. By contrast, our model AIRec outperforms others. The reason is that the item-level attentive-pooling leverages a two-layer network to calculate the similarity between item representation and user's explicit preferences, and thereafter assigns different contributions (weights) to historical items. This

[2] http://www.last.fm.com.
[3] http://delicious.com.

Table 2
Datasets statistics. Columns represent the number of users, the number of items, the number of tags, and the number of times that the users have tagged the items.

Dataset	#Users	#Items	#Tags	#Assignments
Last.Fm	1,808	12,212	2,305	175,641
Delicious	1,843	65,877	3,508	339,744

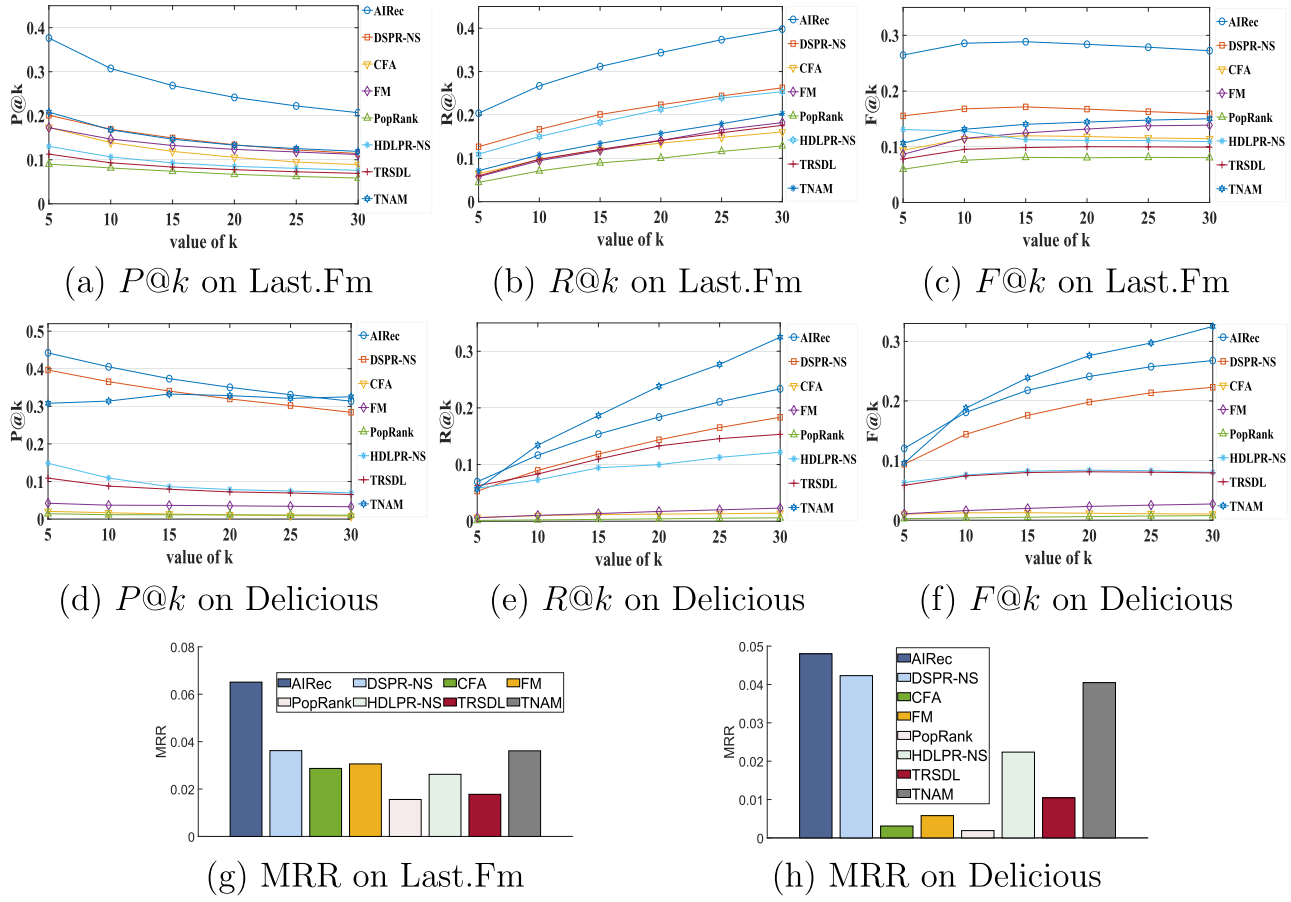


Fig. 3. The personalized recommendation performances on two datasets in terms of MRR, *P@k*, *R@k* and *F@k*.

Table 3
Effectiveness of the Item-level Attention on Lats.FM. AIRec-mean represents the item-level attention network is replaced by mean-pooling, and AIRec-max represents the item-level attention network is replaced by max-pooling.

Model	Last.Fm			
	P@10	*R*@10	*F*@10	MRR
AIRec	**0.3074**	**0.2669**	**0.2857**	**0.0651**
AIRec-mean	0.2996	0.2648	0.2811	0.0621
AIRec-max	0.3043	0.2568	0.2785	0.0609

Table 4
Effectiveness of the Item-level Attention on Delicious. AIRec-mean represents the item-level attention network is replaced by mean-pooling, and AIRec-max represents the item-level attention network is replaced by max-pooling.

Model	Delicious			
	P@10	*R*@10	*F*@10	MRR
AIRec	**0.4052**	**0.1165**	**0.1810**	**0.0480**
AIRec-mean	0.4032	0.1110	0.1741	0.0476
AIRec-max	0.3988	0.1052	0.1664	0.0472

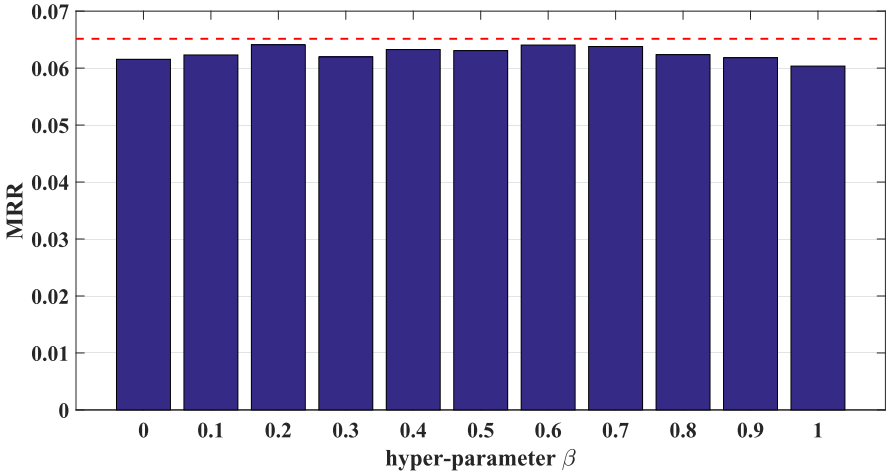


Fig. 4. Effectiveness of the Preference-level Attention. The x-axis represents the variation range of hyper-parameter β, the y-axis represent the MRR score.

approach brings more attention to items that are close to user's interest, thus capturing more precise implicit preferences.

4.3.2. Effect of the preference-level attention

To illustrate the effect of the preference-level self-attentive fusion mechanism, we compare it with the explicitly manual method shown in Eq. (7). The hyper-parameter β is tuned in [0,1]. Fig. 4 shows the results on Last.Fm dataset and we can see that our self-attentive fusion method (red line) achieves the best performance whatever the hyper-parameter β is. We argue there are two reasons. On the one hand, the trade-off parameter for discriminating the saliencies between explicit and implicit preferences can be acquired through automatic learning rather than manual assignment. On the other hand, different from fixing a uniform parameter for all users, self-attentive fusion mechanism learns suitable weight for each user according to the tagging behaviors, resulting in more personalized recommendation.

4.4. Structure investigation (RQ3)

4.4.1. Impact of the intersection module

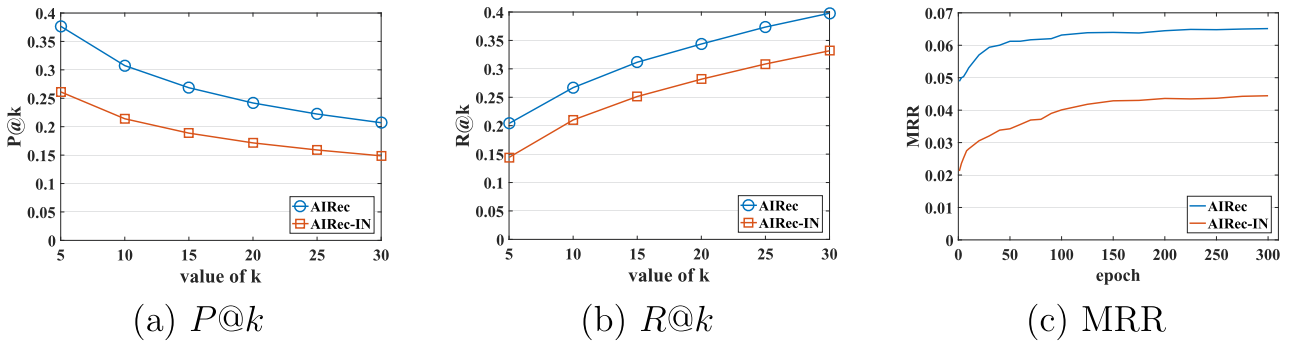
In this part, we further conduct experiments to demonstrate the impact of the elaborately introduced intersection module. We remove the intersection module from AIRec and name it as AIRec-IN. Fig. 5 shows the results on Last.Fm dataset. It's obvious that AIRec outperforms AIRec-IN significantly in all metrics. We argue that the intersection can constrain neural networks to focus on the conjunct features when modeling representations. Besides, adding $\tilde{\mathbf{i}}_{iu}$ enhances the importance of these dimensions in the final representations and benefits subsequent score prediction, resulting in better model fidelity and expressiveness.

4.4.2. Effect of the FM

To ascertain the effectiveness of the introduced FM prediction layer, we replace it with the inner product and MLP. The former uses the inner product of latent features $\tilde{\mathbf{y}}_i^a$ and $\tilde{\mathbf{x}}_u^{H_a}$ for prediction (AIRec-ip) while the latter passes the concatenated vector $\mathbf{z}$ into a MLP (AIRec-mlp). Table 5 and Table 6 illustrates the results on Last.Fm and Delicious dataset. We can see that compared with others, our AIRec achieves the best performance. The reason is that inner product assumes the independence of latent dimensions and ignores the interactions between different dimensions, which degrades model performance. Besides, although MLP learns signals with neural networks, it doesn't model the intersections among latent dimensions explicitly. However, FM takes the second-order interactions into consideration and obtains highly informative results, which is more suitable for TRS recommendation.

5. Conclusion

In this work, we propose a novel tag-aware recommendation model called AIRec which combines neural attention and factorization machine for top-n recommendation. In order to better model user features, we design a hybrid user model with a hierarchical attention network which can both differentiate the contributions of interacted items (item-level) and dynamically discriminate the saliencies between explicit and implicit preferences (preference-level). Besides, we discover the importance of tags intersection in



(a) $P@k$ (b) $R@k$ (c) MRR

Fig. 5. Impact of the Intersection Module. AIRec-In represents the intersection module is removed from original AIRec.

Table 5
Effectiveness of the FM on Last.FM. AIRec-ip represents the prediction layer of AIRec is replaced by inner product operation. AIRec-mlp represents the prediction layer fo AIRec is replaced by a MLP.

Model	Last.Fm			
	P@10	*R*@10	*F*@10	MRR
AIRec	0.3074	0.2669	0.2857	0.0651
AIRec-ip	0.2988	0.2466	0.2702	0.0619
AIRec-mlp	0.2814	0.2437	0.2612	0.0603

Table 6
Effectiveness of the FM on Delicious. AIRec-ip represents the prediction layer of AIRec is replaced by inner product operation. AIRec-mlp represents the prediction layer fo AIRec is replaced by a MLP.

Model	Delicious			
	P@10	*R*@10	*F*@10	MRR
AIRec	0.4052	0.1165	0.1810	0.0480
AIRec-ip	0.3976	0.1087	0.1707	0.0465
AIRec-mlp	0.3892	0.1096	0.1710	0.0471

the TRS and creatively leverage the intersection of user and item tags for constraining neural networks to focus on the conjunct features and enhancing the importance of these dimensions. We leverage the factorization machine to predict rating scores, which measures the second-order interactions and is more suitable for tag-aware recommendation. Extensive experiments on two real-world datasets demonstrate that AIRec significantly outperforms the state-of-the-art baselines in tag-aware top-n recommendation. Future work includes using RNN to learn the tag-based temporal representations of user preferences. Besides, we are also interested to explore explainable tag-aware recommender systems.

CRediT authorship contribution statement

Bo Chen: Conceptualization, Methodology, Software, Validation, Writing - original draft. **Yue Ding:** Methodology, Investigation, Resources, Validation, Writing - review & editing. **Xin Xin:** Methodology. **Yunzhe Li:** Validation. **Yule Wang:** Validation. **Dong Wang:** Supervision.

Declaration of Competing Interest

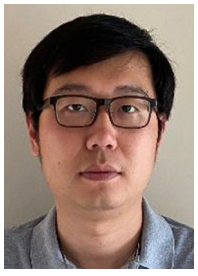
The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] H. Zhang, Y. Sun, M. Zhao, T.W. Chow, Q.J. Wu, Bridging user interest to item content for recommender systems: an optimization model, IEEE Transactions on Cybernetics (2019).
[2] J.J. Jung, Discovering community of lingual practice for matching multilingual tags from folksonomies, The Computer Journal 55 (3) (2011) 337–346.
[3] Z. Yan, J. Zhou, User recommendation with tensor factorization in social networks, in: ICASSP, IEEE, 2012, pp. 3853–3856.
[4] C. Chen, X. Zheng, Y. Wang, F. Hong, D. Chen, et al., Capturing semantic correlation for item recommendation in tagging systems, in: AAAI, 2016, pp. 108–114.
[5] H. Liu, Resource recommendation via user tagging behavior analysis, Cluster Computing (2017) 1–10.
[6] Z.-K. Zhang, C. Liu, Y.-C. Zhang, T. Zhou, Solving the cold-start problem in recommender systems with social tags, EPL (Europhysics Letters) 92 (2) (2010) 28002.
[7] A. Shepitsen, J. Gemmell, B. Mobasher, R. Burke, Personalized recommendation in social tagging systems using hierarchical clustering, in: RecSys, ACM, 2008, pp. 259–266.
[8] I. Cantador, A. Bellogín, D. Vallet, Content-based recommendation in social tagging systems, in: RecSys, ACM, 2010, pp. 237–240.
[9] Z.-K. Zhang, T. Zhou, Y.-C. Zhang, Personalized recommendation via integrated diffusion on user–item–tag tripartite graphs, Physica A: Statistical Mechanics and its Applications 389 (1) (2010) 179–186.
[10] Y. Zuo, J. Zeng, M. Gong, L. Jiao, Tag-aware recommender systems based on deep neural networks, Neurocomputing 204 (2016) 51–60.
[11] Z. Xu, C. Chen, T. Lukasiewicz, Y. Miao, X. Meng, Tag-aware personalized recommendation using a deep-semantic similarity model with negative sampling, in: CIKM, ACM, 2016, pp. 1921–1924.
[12] N. Liang, H.-T. Zheng, J.-Y. Chen, A.K. Sangaiah, C.-Z. Zhao, Trsdl: Tag-aware recommender system based on deep learning–intelligent computing systems, Applied Sciences 8 (5) (2018) 799.
[13] Y. Zhen, W.-J. Li, D.-Y. Yeung, Tagicofi: tag informed collaborative filtering, in: RecSys, ACM, 2009, pp. 69–76.
[14] L.B. Marinho, L. Schmidt-Thieme, Collaborative tag recommendations, in: Data Analysis, Machine Learning and Applications, Springer, 2008, pp. 533–540.
[15] M. De Gemmis, P. Lops, G. Semeraro, P. Basile, Integrating tags in a semantic content-based recommender, in: RecSys, ACM, 2008, pp. 163–170.
[16] Z. Xu, O. Tifrea-Marciuska, T. Lukasiewicz, M.V. Martinez, G.I. Simari, C. Chen, Lightweight tag-aware personalized recommendation on the social web using ontological similarity, IEEE Access 6 (2018) 35590–35610.
[17] Z. Xu, T. Lukasiewicz, C. Chen, Y. Miao, X. Meng, Tag-aware personalized recommendation using a hybrid deep model, in: IJCAI, 2017, pp. 3196–3202.
[18] F. Strub, R. Gaudel, J. Mary, Hybrid recommender system based on autoencoders, in: DLRS, ACM, 2016, pp. 11–16.
[19] D. Shin, S. Cetintas, K.-C. Lee, I.S. Dhillon, Tumblr blog recommendation with boosted inductive matrix completion, in: CIKM, ACM, 2015, pp. 203–212.
[20] D. Kim, C. Park, J. Oh, S. Lee, H. Yu, Convolutional matrix factorization for document context-aware recommendation, in: RecSys, ACM, 2016, pp. 233–240.
[21] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T.-S. Chua, Neural collaborative filtering, in: WWW, International World Wide Web Conferences Steering Committee, 2017, pp. 173–182.
[22] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu, T.-S. Chua, Attentional factorization machines: learning the weight of feature interactions via attention networks, arXiv preprint arXiv:1708.04617 (2017).
[23] J. Chen, H. Zhang, X. He, L. Nie, W. Liu, T.-S. Chua, Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention, in: SIGIR, ACM, 2017, pp. 335–344.
[24] S. Seo, J. Huang, H. Yang, Y. Liu, Interpretable convolutional neural networks with dual local and global attention for review rating prediction, in: RecSys, ACM, 2017, pp. 297–305.
[25] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, K. Gai, Deep interest network for click-through rate prediction, in: SIGKDD, ACM, 2018, pp. 1059–1068.
[26] R. Huang, N. Wang, C. Han, F. Yu, L. Cui, Tnam: A tag-aware neural attention model for top-n recommendation, Neurocomputing 385 (2020) 1–12.
[27] Z. Lin, M. Feng, C.N. d. Santos, M. Yu, B. Xiang, B. Zhou, Y. Bengio, A structured self-attentive sentence embedding, arXiv preprint arXiv:1703.03130 (2017).
[28] P. Li, W. Li, Z. He, X. Wang, Y. Cao, J. Zhou, W. Xu, Dataset and neural recurrent sequence labeling model for open-domain factoid question answering, arXiv preprint arXiv:1607.06275 (2016.
[29] S. Rendle, Factorization machines, in: ICDM, IEEE, 2010, pp. 995–1000.
[30] W. Guo, S. Wu, L. Wang, T. Tan, Personalized ranking with pairwise factorization machines, Neurocomputing 214 (2016) 191–200.
[31] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, Bpr: Bayesian personalized ranking from implicit feedback, in: UAI, AUAI Press, 2009, pp. 452–461.

[32] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, The Journal of Machine Learning Research 15 (1) (2014) 1929–1958.
[33] F. Yuan, G. Guo, J.M. Jose, L. Chen, H. Yu, W. Zhang, Lambdafm: learning optimal ranking with factorization machines using lambda surrogates, in: CIKM, ACM, 2016, pp. 227–236.
[34] I. Cantador, P.L. Brusilovsky, T. Kuflik, Second Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec2011), ACM, 2011.
[35] J. Bobadilla, F. Ortega, A. Hernando, A. Gutiérrez, Recommender systems survey, Knowledge-based systems 46 (2013) 109–132.
[36] E.M. Voorhees, et al., The trec-8 question answering track report., in: Trec, vol. 99, 1999, pp. 77–82.
[37] J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, Journal of Machine Learning Research 12 (Jul) (2011) 2121–2159.
[38] H. Guo, R. Tang, Y. Ye, Z. Li, X. He, Deepfm: a factorization-machine based neural network for ctr prediction, arXiv preprint arXiv:1703.04247 (2017).

Bo Chen received the B.S. in Computer Science and Engineering from South China University of Technology in 2017 and received the M.S. degree in Software Engineering from Shanghai Jiao Tong University in 2020. His current research interests include recommender systems, deep learning and automated machine learning.



Yue Ding received the Ph.D. from Shanghai Jiao Tong University in 2018. He is now a lecturer in School of Software, Shanghai Jiao Tong University. His main research interests are recommender systems, deep learning, graph neural networks and data mining.

Xin Xin now is a PhD candidate at School of Computing Science, University of Glasgow. His research mainly focuses on recommender systems, including deep & graph learning models and reinforcement learning. His research leads to publications on tire-1 conferences, like SIGIR, IJCAI, ACL and CIKM. He also serves as reviewers for top-ranked academic venues, such as TOI S, TKDE, ACL, EMNLP, SIGIR, WSDM and MM.

Yunzhe Li received the B.S. in software engineering from Sun Yat-Sen University, Guangzhou, China, in 2019. He is currently working toward the M.S. degree in computer science and technology at Shanghai JiaoTong University, Shanghai, China. His research interests involve data mining, recommendation, and information retrieval.

Yule Wang received the B.S. in software engineering and is currently working towards the master degree at Scohool of Software, Shanghai Jiao Tong University, Shanghai, China. His research focuses on recommender systems, information retrieval and data mining.

Dong Wang received B.S. from Xi'an Jiaotong University (Xi'an, China) in 1991, M.S. from Chongqing University (Chongqing, China) in 1994, and Ph.D. from Shanghai Jiao Tong University (Shanghai, China) in 2001. He is currently the director and doctoral supervisor of the Institute of RFID and Internet of Things, School of Software, Shanghai Jiao Tong University. In recent years, his research interests are RFID and Internet of Things, wireless sensing technology, data mining and recommender systems.