applied sciences

MDPI

Article

TRAL: A Tag-Aware Recommendation Algorithm Based on Attention Learning

Yi Zuo, Shengzong Liu *, Yun Zhou and Huanhua Liu

School of Information Technology and Management, Hunan University of Finance and Economics, Changsha 410205, China

* Correspondence: lsz@hufe.edu.cn

Abstract: A social tagging system improves recommendation performance by introducing tags as auxiliary information. These tags are text descriptions of target items provided by individual users, which can be arbitrary words or phrases, so they can provide more abundant information about user interests and item characteristics. However, there are many problems to be solved in tag information, such as data sparsity, ambiguity, and redundancy. In addition, it is difficult to capture multi-aspect user interests and item characteristics from these tags, which is essential to the recommendation performance. In the view of these situations, we propose a tag-aware recommendation model based on attention learning, which can capture diverse tag-based potential features for users and items. The proposed model adopts the embedding method to produce dense tag-based feature vectors for each user and each item. To compress these vectors into a fixed-length feature vector, we construct an attention pooling layer that can automatically allocate different weights to different features according to their importance. We concatenate the feature vectors of users and items as the input of a multi-layer fully connected network to learn non-linear high-level interaction features. In addition, a generalized linear model is also conducted to extract low-level interaction features. By integrating these features of different types, the proposed model can provide more accurate recommendations. We establish extensive experiments on two real-world datasets to validate the effect of the proposed model. Comparable results show that our model perform better than several state-of-the-art tag-aware recommendation methods in terms of HR and NDCG metrics. Further ablation studies also demonstrate the effectiveness of attention learning.

Keywords: attention learning; tag information; tag-aware recommendation

Citation: Zuo, Y.; Liu, S.; Zhou, Y.; Liu, H. TRAL: A Tag-Aware Recommendation Algorithm Based on Attention Learning. *Appl. Sci.* **2023**, *13*, 814. https://doi.org/10.3390/app13020814

Academic Editors: Giacomo Fiumara, Pasquale De Meo, Xiaoyang Liu and Annamaria Ficara

Received: 14 December 2022
Revised: 3 January 2023
Accepted: 4 January 2023
Published: 6 January 2023

1. Introduction

A recommendation system (RS) has been considered as an extremely effective instrument to tackle the problem of information overload, because it can provide personalized services for individual users by analyzing their interests, preferences, and needs [1]. Many algorithms have been proposed to generate personalized recommendations. To enhance algorithm performance, other superior side information has been incorporated into the recommender system in recent years. In particular, tag-aware recommender systems (TRS) allow users to mark custom tags for relevant items. In this way, TRS can build the implicit relationship between users and items through a wide range of tags. These tags are generally composed of concise words or phrases defined by users, providing good supplementary information for describing user preferences and item characteristics [2]. Thus far, TRS have successfully found applications in many online business services, such as books, movies, music, videos, and social media.

Although the introduction of tags can advance the recommendation performance, some new problems will inevitably arise. For example, most users may only mark a few tags to a few items, resulting in sparse data. In addition, since users can take any word or phrase as a tag, it is easy to cause redundancy and ambiguity in the tag latent

space [3–5]. Since user-defined tags are the key factor in expressing user interests and item features, whether the tag information can be effectively processed is crucial to ensure the recommendation performance. Accordingly, the clustering techniques are introduced to extend the traditional collaborative filtering (CF) into TRS [3,4]. The goal of clustering is to aggregate redundant tags. However, it is hard to calculate the similarity of tags, especially when the tag space is extremely sparse. The tool WordNet [6], known as an online English lexical database, is also absorbed to compute the similarity for improved tag-aware recommendation [7]. Nevertheless, manually defining a valid dictionary is time-consuming and words in dictionaries are usually limited. More importantly, these methods cannot generate high-quality recommendation results. The reason may be that the used learning methods are shallow structures that are insufficient to mine the potential meaning of tags.

To obtain more abstract latent features, researchers begin to leverage the deep network model, which has been proved as its most powerful feature expression ability in many fields [8]. For instance, ACF [5] adopts the deep autoencoders to extract low-dimensional dense user features based on tags. These tag-based features are then utilized by user-based CF to generate recommendations. Experimental results show that ACF is obviously better than the clustering-based CF. DSPR [9] uses deep neural networks to obtain the abstract tag-based features of users and items and maximizes similarities between users and their associated items based on those features. TRSDL [10] employs deep neural networks and recurrent neural networks to learn the non-linear latent features of items and users, respectively. Then, the rating prediction is conducted based on these latent features.

In TRS, users may mark various tags to different items, indicating their diverse interests. Similarly, items are assigned multifarious tags that describe their various characteristics. However, in most deep network-based recommendation algorithms, user-defined tags are first transformed into multi-hot feature vectors, and then compressed into a fixed-length representation vector for a given user or a specific item by sum or average pooling, and finally concatenated together to feed into a multi-layer perceptron (MLP) to learn the non-linear relations. In other words, multi-aspect user preferences or item characteristics are compressed into a certain fixed-length feature vector. In order to represent diverse characteristics, the dimension of the feature representation should be large enough to have sufficient expression ability. However, this will significantly increase the scale of learning parameters, causing computing and storage burden.

In addition, as is known to us, the user's preference on a target item comes from the fact that certain characteristics of the item exactly match some specific interests of the user. Therefore, it is not suitable to compress all the diverse interests of a user into the same representation vector when estimating a candidate item, as not all features are equally useful. The useless features may even produce unnecessary noises and deteriorate the recommendation performance. In short, ingenious approaches that can differentiate the importance of different features are required to extract tag-based latent features. Furthermore, although deep networks can automatically learn more expressive feature representations, it is not easy to extract appropriate low-dimensional dense representations for users and items when the potential tag space is very sparse. As discussed in [11–13], both low-level and high-level feature interactions should play important roles for recommendation performance, since such interactions of features behind user preferences and item features are highly sophisticated.

To process the above-mentioned issues, we develop a tag-aware recommendation algorithm based on attention learning (TRAL), which adopts the attention mechanism to discriminate the importance of different features from tag space. Firstly, we utilize the tag embedding technique to extract low-dimensional dense features from the user-tag matrix and item-tag matrix. Secondly, to acquire more abstract and effective representation vectors for each user or item, the attention-based pooling layer is employed to compress these features to a single representation. In this way, different features are assigned different weights according to their importance. Therefore, tag-based features can make different

contributions to the final prediction. More importantly, the use of the attention mechanism means that the importance of different features can be automatically learned without any human domain knowledge. Thirdly, the extracted representation vectors for users and items are concatenated together to feed into a general MLP. The high-level feature interactions are, hence, further learned for improving the recommendation performance. In addition, to make full use of low-level feature interactions, the generalized linear model is also introduced. Finally, we combine the representation vectors obtained from the linear model and the depth model and input them into a common logic loss function for joint training.

To sum up, the main contributions of our work are listed as follows:

- We point out the limitation of using simple compression methods to obtain the fixed-length tag-based vector that represents multi-aspect user preferences or item features. To this end, we develop a new tag-aware recommendation algorithm which introduces the attention network to adaptively learn the importance of different features.
- We combine a generalized linear model and a deep neural network so as to take advantages of both low-level and high-level feature interactions.
- We perform extensive experiments on two real-world datasets, demonstrating the rationality and effectiveness of the proposed TRAL.

The rest of this paper is organized as follows. Section 2 briefly summarizes the related work. Section 3 introduces some preliminaries. We elaborate on the proposed TRAL in Section 4 and conduct experiments in Section 5. Conclusions are given in Section 6.

2. Related Work

Naturally, many traditional recommendation algorithms are extended to TRS. For example, Nakamoto et al. [14] proposed a tag-based contextual CF model which modifies the user similarity computation and the item score prediction according to the tagging information. Marinho et al. [15] projected the ternary relation of the user-item-tag to a lower-dimensional space where CF can be applied to provide recommendations. Zhen et al. [16] incorporated tagging information seamlessly into the model-based CF method by regularizing the matrix factorization procedure. Chen et al. [17] developed a tag-based CF model that adopts topic modeling to capture the semantic information of tags for users and items, respectively. Wang et al. [18] devised a robust and efficient probabilistic model based on Bayesian principle for tag-aware recommendation.

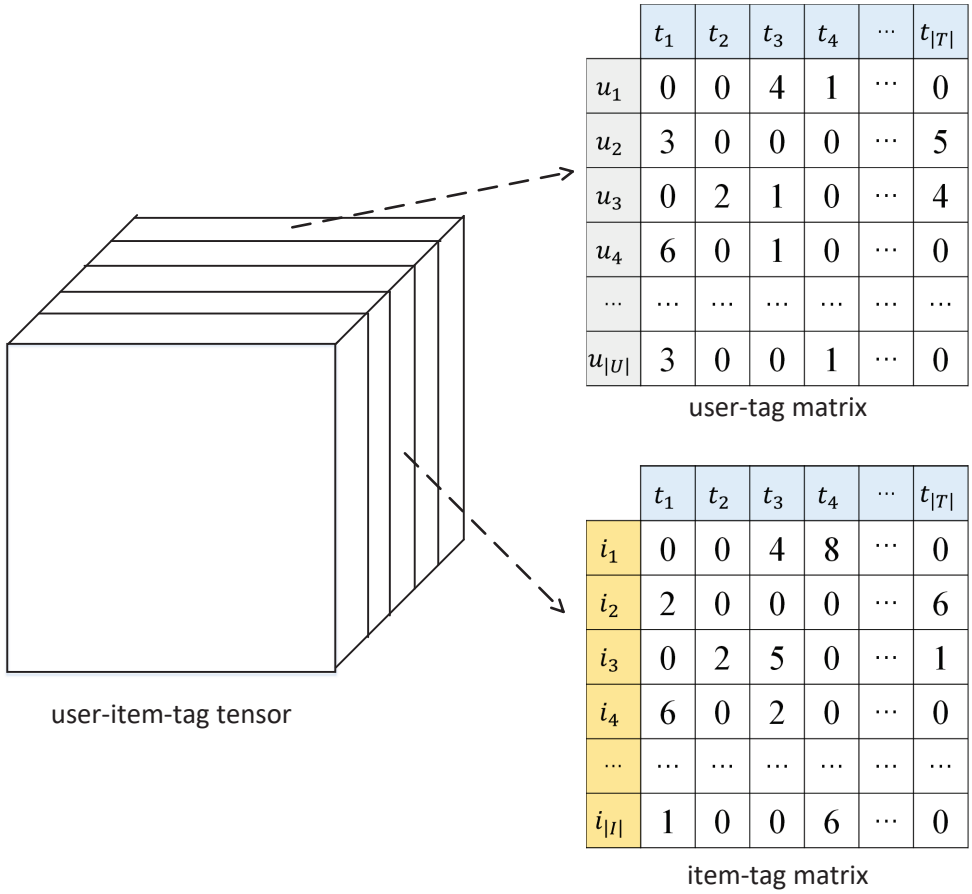
To tackle the problem of ambiguity and redundancy in tag information, other kinds of methods have been widely investigated. Shepitsen et al. [3] designed a personalized tag-aware recommendation algorithm based on hierarchical clustering. Through the clustering method, redundant tags can be aggregated, and the user's preferences can be better understood by measuring the importance of associated tag clusters. Symeonidis et al. [19] developed a general tag-aware framework to model the three types of entities: user, items, and tags. The modeled 3-dimensional data is first represented by a 3-order tensor and the dimension reduction is then performed via a higher-order singular-value decomposition. To address the problem of high dimension and sparsity of tagging information, Li et al. [20] developed a novel tag-aware recommendation framework based on Bayesian personalized ranking (BPR) with matrix factorization, where the tag mapping scheme was designed to capture low-dimensional dense features for users and items. Different from the method based on dimension reduction, Zhang et al. [21] developed an integrated diffusion-based recommendation model directly based on user-item-tag tripartite graphs. In the recent work, Pan et al. [22] designed a social tag expansion model to alleviate the tag sparsity problem. The model can explore relations among tags and assign proper weights to the expanded tags. By updating the user profile dynamically through the assigned weights, better recommendation performance can be gained. In [23], the topic optimization was introduced into CF to further enhance both the effectiveness and the efficiency of tag-aware recommendations. In the proposed method, the tags' topic model is established and then used to find the latent preference of users and the latent affiliation of items on topics.

Due to its powerful ability for feature extraction, deep learning has been widely employed in TRS recently. Zuo et al. [5] developed a tag-aware deep model, where tag-based latent features for users are learned by the deep autoencoders. Xu et al. [24] developed a novel tag-aware recommendation model which adopts deep-semantic similarity-based neural networks to extract tag-based representations for users and items. In addition, negative sampling technique is applied so as to enhance the efficiency of the training process. Based on this model, autoencoders are integrated to further accelerate the learning process in [9]. Liang et al. [10] proposed a hybrid tag-aware model by combining deep neural networks and recurrent neural networks for rating prediction. The task of deep neural networks is to capture abstract representation of item characteristics, while the aim of recurrent neural networks is to model user dynamic preferences. Huang et al. [25] proposed a novel tag-based recommendation model that combines the attention network, the stacked autoencoder and MLP to provide recommendations. In the proposed model, a neural attention network is conducted to overcome the difficulty of assigning tag weights for personalized users. Chen et al. [26] designed an attentive intersection model which integrates the neural attention network and factorization machine. The proposed model fully utilizes the intersection between user and item tags to learn conjunct features. Recently, Ahmadian et al. [27] proposed a new tag-aware algorithm that employs deep neural networks to model the representation of trust relationships and tag information.

3. Preliminaries

Generally, users are allowed to assign certain items with personalized tags in TRS. These different tags can indicate user interests and item characteristics from several angles. By fully exploring the rich tagging information, TRS can further capture the connotation of tags, abstract features of items and predict preferences of users, thereby improving the quality of recommendations. Suppose the size of user set U, item set I, and tag set T are $|U|$,$|I|$, and $|T|$, respectively. The user tagging behavior can be formally defined as a tuple $F = (U, T, I, Y)$, in which Y indicates the internal relations between users, items and tags. More specifically, we can use the following 3-order tensor to represent Y: $Y = y_{(u,i,t)} \in \mathbb{R}^{|U|\times|I|\times|T|}$. If a given user u labels an item i with tag t, the corresponding $y_{(u,i,t)} = 1$, otherwise $y_{(u,i,t)} = 0$.

Given a user u and a tag t, we can compute the number of times that u has marked items with t. Analogously, the number of times that the item i has been labeled with tag t is also calculated. In this way, the user–item–tag tensor is decomposed into two adjacent matrices: user–tag matrix and item–tag matrix, as shown in Figure 1. Each row of the user–tag matrix represents the tag-based feature for one user, while each row of the item–tag matrix indicates the tag-based feature for one item. Note that each user often utilizes many tags, and each item is usually annotated by several tags. Consequently, tag-based multi-valued discrete vectors are obtained for users and items, respectively. The aim of the proposed model is to generate the personalized ranked item list for each user based on these tag-based features, also known as the top-n recommendation.



	t_1	t_2	t_3	t_4	⋯	$t_{\|T\|}$
u_1	0	0	4	1	⋯	0
u_2	3	0	0	0	⋯	5
u_3	0	2	1	0	⋯	4
u_4	6	0	1	0	⋯	0
⋯	⋯	⋯	⋯	⋯	⋯	⋯
$u_{\|U\|}$	3	0	0	1	⋯	0

	t_1	t_2	t_3	t_4	⋯	$t_{\|T\|}$
i_1	0	0	4	8	⋯	0
i_2	2	0	0	0	⋯	6
i_3	0	2	5	0	⋯	1
i_4	6	0	2	0	⋯	0
⋯	⋯	⋯	⋯	⋯	⋯	⋯
$i_{\|I\|}$	1	0	0	6	⋯	0

Figure 1. An example of obtaining the user–tag matrix and the item–tag matrix by decomposing the user–item–tag tensor.

4. Method

In this section, we describe the proposed TRAL in detail, the overall structure of which is presented in Figure 2. There are three main modules in the framework: (1) the deep component that integrates a neural attention network and fully connected layers to capture higher-order tag-based features for users and items; (2) the wide component that conducts the generalized linear model to learn low-order features; and (3) the predict layer that combines high-order and low-order features and leverages joint optimization for generating personalized recommendations. More specifically, we first obtain the tag-based representation vectors for users and items from the user–item–tag tensor. These vectors are then used as the input of the deep component and the wide component, respectively, to capture high-order and low-order interaction features. Finally, we integrate these features of different types in the predict layer to provide high-quality tag-aware recommendation.

4.1. The Deep Component

In TRS, user interests and item characteristics are hidden in tagging behavior data. It is remarkable to capture latent tag-based features for users and items, which is the key to advance the performance of recommendations. Consequently, we design a deep component to make the best of deep learning in representation and combination. As presented in Figure 2, the deep component is composed of three main layers: embedding layer, attention layer, and interaction layer.

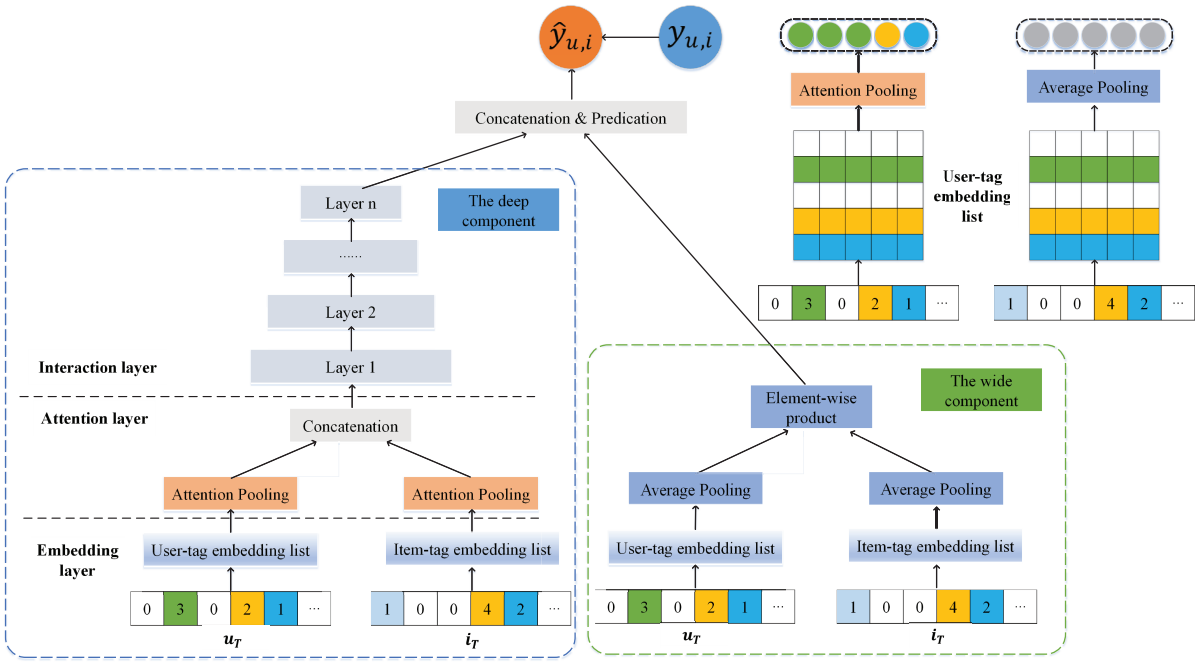


Figure 2. The overall structure of the proposed model. The upper right corner illustrates the difference between average pooling and attention pooling. The weights of different feature vectors are set to the same value in average pooling, while these weights will be automatically learned according to their importance in attention pooling.

4.1.1. Embedding Layer

In order to transform the high-dimensional sparse binary features into low-dimensional dense vectors, we introduce the widely used embedding method, which is inspired by representations for words and phrases [28]. Concretely, we first construct a tag-based embedding dictionary for users:

$$E^u = [e_1^u, \cdots, e_j^u, \cdots, e_{|T|}^u] \in \mathbb{R}^{d \times |T|} \tag{1}$$

where $e_j^u \in \mathbb{R}^d$ represents the embedding vector for tag j that is relative to users and d is the embedding size. Then, for a given user, we extract the corresponding row of the user-tag matrix to produce its user-tag vector u_T, which is apparently a multi-valued discrete vector. Assuming that $u_T[x] >= 1$ for $x \in \{j_1, j_2, \cdots, j_k\}$, we can acquire the tag-based embedding representation according to the table lookup mechanism, which is a list of embedding vectors: $h_t^U = \{e_{j_1}^U, e_{j_2}^U, \cdots, e_{j_k}^U\}$. In particular, in order to capture more accurate potential features based on tags, we also embed tag frequencies, since they can reveal the degree of user preference and item properties [29]. Finally, the resulting embedding list is as follows:

$$h_t^U = \{e_{j_1}^U || e_{j_1}^F, e_{j_2}^U || e_{j_2}^F, \cdots, e_{j_k}^U || e_{j_k}^F\} \tag{2}$$

where $||$ denotes the concatenation operation and $e_{j_1}^F$ represents the embedding of tag frequency which is divided into discrete buckets in the pre-processing. For a target item, a similar method can be conducted to generate its embedding representation h_t^I. Although tags describing users and items belong to the same set, the latent feature of users and items are obviously different. For this reason, we use two different embedding dictionaries for users and items, respectively.

4.1.2. Attention Layer

In a real-life scenario, there are obvious differences in users' behavior habits and cultural backgrounds. Therefore, the number and the content of tags for the same item

marked by different users will be significantly different. Note that the input dimensions of full connected neural networks are required to be consistent. Additional operators should be taken to convert the variable list of embedding vectors to a fixed-length vector. A common approach is to construct a pooling layer, where the list of embedding vectors is compressed by a weighted sum operator. Suppose that the list of embedding vectors for a given user is $h_t^U = \{e_1, e_2, \cdots, e_L\}$, the compressed feature vector is obtained by the following weighted sum operation:

$$v_t^U = \sum_{j=1}^{L} a_j e_j \tag{3}$$

where a_j is the weight indicating the importance of the corresponding tag-based embedding vector e_j. Two widely adopted strategies are sum pooling and average pooling, which treat each embedding vector equally and set all weights to the same value.

Clearly, user tagging behaviors play critical role in modeling user preferences in TRS. For a certain user, the compressed fix-length feature vector by sum or average pooling remains constant no matter what the predicted item is. It requires that the tag-based feature vector is capable to express multiple interests of users. To this end, the embedding size should be expanded to large enough, resulting in the increase in computation burden. Moreover, we argue that not all user tagging information are equally important and effective when predicting a target item. Those tag-based features that are less useful should naturally be given a lower weight, which is the main limitation of sum or average pooling.

Based on the above considerations, we resort to the attention mechanism, which allows the weight to be calculated automatically from data. The rationale is that the contributions of different embedding vectors should be taken into consideration when compressing them into a single representation vector. In this work, we construct a two-layer neural network to realize the attention mechanism. Specifically, the weight of each tagging feature vector is first calculated by:

$$\hat{a_j} = q^T ReLU(W_a e_j + b) \tag{4}$$

where $W_a \in \mathbb{R}^{m \times d}$, $b \in \mathbb{R}^m$, $q \in \mathbb{R}^m$ are learnable parameters, and m represents the number of hidden layer neurons in the attention network, which is called attention factor. The final attention weights are then normalized by a softmax function:

$$a_j = \frac{\exp(\hat{a_j})}{\sum_{j=1}^{L} \exp(\hat{a_j})} \tag{5}$$

In this way, a fixed-length user representation v_t^U can be adaptively derived by discriminating the importance of different embedding feature vectors. An attention network with similar structures is created so as to generate a fixed-length item representation v_t^I.

4.1.3. Interaction Layer

To capture the high-order interaction features, the obtained tag-based user and item representations from the attention layer are concatenated and further fed into a fully connected neural network with multiple layers. In this way, we can enhance the flexibility and non-linearity of our model to learn the interactions between user v_t^U and item v_t^I, compared with the simple element-wise product operations. Formally, given the concatenated feature vector $Z_0 = [v_t^U || v_t^I]$, the update rule of the neural network in the k-th layer can be defined as:

$$Z_k = \sigma_k(W_k Z_{k-1} + b_k) \tag{6}$$

where W_k, b_k and σ_k represent the weight matrix, the bias vector, and the activation function for the k-th layer, respectively. For the activation function, we apply ReLU (Rectifier), which is proved to be non-saturated and well-suited for sparse data [30]. It is worth mentioning

that the network is established by using a classic tower structure, in which the number of neurons in each layer gradually decreases from bottom to top. Consequently, after performing the computation layer by layer, we can obtain more abstract representation from the tag-based user-item interactions.

4.2. The Wide Component

Low-order feature interactions are beneficial to the recommendation results and can be used as an effective supplement to high-order features. Therefore, the wide component adopts a generalized matrix factorization method [11]. Suppose the latent vectors for user u and item i are v_u and v_i, respectively. The representation vector for low order feature interactions is calculated by:

$$Z_w = v_u \odot v_i \tag{7}$$

where $\odot$ means the element-wise product of two vectors. To obtain the tag-based latent vectors for users and items, we adopt a similar approach as used in the embedded layer of the deep component. Specifically, two embedding dictionaries for users and items are first established, respectively. Then, a list of embedding vectors representing a given user or item is obtained by extracting the corresponding rows from the embedding dictionary. To obtain a fixed-length latent vector, we finally perform a simple average pooling. It should be noted that we do not use the attention network in the wide component. The reason for this lies in two aspects. On the one hand, the wide component focuses on learning low order features by making full use of the generalized linear model, so the attention network is unnecessary. On the other hand, it will bring more parameters, making training more difficult, and increase the possibility of over-fitting. Specifically, in order to further improve the training efficiency, we share the tag embeddings used in the deep component.

4.3. Joint Optimization

In our work, two different components are established for capturing high-order and low-order feature interactions, respectively. To further combine the advantages of both, we perform the joint optimization. More specifically, the outputs of the two components are concatenated and then input into the predict layer for joint training. Let Z_w and Z_d denote the tag-based interaction feature from the wide and deep component, respectively. The prediction of the combined model is formally defined as:

$$\hat{y_{u,i}} = \sigma(W^T[Z_w||Z_d] + b) \tag{8}$$

where $\sigma(\cdot)$ is the sigmoid function, W is the weight matrix, b is the bias term, and $\hat{y_{u,i}}$ is the predicted score that measures how much users u like item i.

The commonly used negative log-likelihood function is taken as the loss function, which can be defined as:

$$L = -\sum_{(u,i)\in S^+\cup S^-} (y_{u,i} \log \hat{y_{u,i}} + (1 - y_{u,i}) \log(1 - \hat{y_{u,i}})) \tag{9}$$

where S^+ and S^- denote the positive and negative sample set, respectively. If the sample $(u, i) \in S^+$, $y_{u,i}$ is set to 1. Otherwise, $y_{u,i}$ is set to 0. For a given user u, the positive sample (u, i^+) can be easily obtained from the observed interactions, while the negative sample (u, i^-) is selected from the non-interacted items. If all unobserved interactions are treated as negative samples, the amount of calculation will inevitably increase dramatically. To cope with this problem, we adopt the negative sampling technique [11,24,31], which generates negative instances by randomly sampling from the unobserved interactions based on a uniform distribution. More concretely, for each positive sample, we randomly select a certain number of negative samples.

Moreover, we utilize the L_2 regularization to avoid over-fitting. The final objective function can be defined as:

$$L = -\sum_{(u,i)\in S^+\cup S^-} (y_{u,i}\log \hat{y_{u,i}} + (1-y_{u,i})\log(1-\hat{y_{u,i}})) + \lambda\|W\|_2 \quad (10)$$

where λ indicates the strength of regularization. Joint optimization is finally achieved by back-propagating the gradients from the output to the wide and deep components of the proposed model simultaneously with the help of mini-batch stochastic optimization. To further reduce the computational load, we introduce the mini-batch aware regularization [32], which only computes the L_2 norm on the parameters of non-zero sparse features in the current mini-batch.

5. Experiments

In this section, we elaborate on our experiments, including datasets, evaluation metrics, baselines, parameter settings, comparison results, and related analysis.

5.1. Dataset Description and Evaluation Metrics

To measure the proposed tag-aware model, we conduct a series of experiments on the following two public datasets: Delicious and LastFM, which are both published on the website of HetRec [33].

- Delicious is a dataset obtained from the Delicious social bookmarking system, which allows users to annotate web bookmarks with various tags. In this dataset, bookmarks are regarded as items to be recommended.
- LastFM is a dataset collected from Last.fm online music system, where users are encouraged to tag music artists they have listened. In this dataset, artists are treated as items to be recommended.

Note that user–item interactions are established by the tagging behaviors. Following the same assumption as in [5,24], we consider that an item is liked by those users which have tagged this item. In addition, those infrequent tags used less than 5 times in Last.Fm and 15 times in Delicious are eliminated to alleviate sparsity of tagging information [5]. Specific statistics of the two processed datasets are summarized in Table 1. The task of the proposed model is to provide recommendations based on user–item interactions and tagging information.

Table 1. Statistics of the two datasets.

Dataset	#Users	#Items	#Tags	#Assignments
Delicious	1843	65,877	3508	339,744
LastFM	1808	12,212	2305	175,641

To measure the results of the recommendation, we perform the common leave-one-out evaluation [11,34]. More specifically, for each user, we take the last interacted item as the positive test instance and leave the remaining interactions for training. Moreover, we sample 99 items as negative instances for each user randomly from the item set that are not interacted by this user. Adding the positive instance, a test set of 100 items is obtained. Instead of using all the non-interactive items as negative instances, the random sampling strategy can dramatically reduce the amount of calculation [35,36]. After predicting the relevant scores of each item in the test set, the recommendation model will provide a top-n ranked list for each user. The performance of the ranked list is finally estimated by Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG) [37]. HR considers whether the positive item appears in the top-n list, while NDCG measures the quality of ranking by computing the position of the positive item in the list. The higher score of HR and NDCG indicate better recommendation results.

5.2. Baselines and Parameter Settings

To show the effectiveness of the proposed TRAL, we compare the recommendation performance with the following baselines.

- CCF: CCF uses hierarchical clustering to obtain different tag clusters, each of which can be viewed as the representation of a certain topic area [3]. Cluster-based feature vectors for users and items are generated and the relevance relation between them can be estimated.
- ACF: ACF introduces the deep autoencoders to derive tag-based user latent features, on which user-based CF is performed to provide recommendations [5].
- NCF: It is a general framework to employ neural network architectures for CF [11]. By replacing the inner product with a MLP, NCF can learn non-linear interactions between users and items.
- DSPR: DSPR adopts MLPs with shared parameters to extract latent features for users and items based on tagging information [24]. Deep-semantic similarities between target users and their relative items can be computed to generate the ranked recommendation list.
- TRSDL: It is a tag-aware recommendation method, which introduces pre-trained word embeddings to represent tag information and learns latent features of users and items via deep structures [10].
- BPR-T: It is a ranking-based collaborative filtering model which incorporates the tag mapping scheme and the Bayesian ranking optimization [20].
- STEM: STEM establishes a new social tag expansion model to tackle the problem of tag sparsity, thereby improving the recommendation accuracy [22].

To guarantee the recommendation performance, we randomly select one interaction for each user as the validate set to determine hyper-parameters of each model. For the sake of fairness, each model is optimized by the mini-batch Adam [38]. The learning rate of each model is tuned in {0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05}, while the batch size is searched from {128, 256, 512, 1024}. The number of negative samples is search from 1 to 10. The maximum number of iterations for optimization is fixed to 300 for all the models. Early stopping strategy is also applied in the light of the performance on the validation set. For the proposed TRAL, the embedding dimension of tags and the attention factor are set as 32 and 16, respectively. The embedding dimension of tag frequency is fixed to 8. In addition, we construct a tower network architecture with three layers. The c is searched from $\{1 \times 10^{-5}, 1 \times 10^{-4}, 1 \times 10^{-3}, 1 \times 10^{-2}, 1 \times 10^{-1}\}$. The specific parameters of our model are listed in Table 2. The proposed model is implemented with Pytorch and all experiments are conducted on a PC configured with an Intel Core I9-10900X @3.40GHz CPU with 32 GB memory, and an Nvidia GeForce RTX 3080 Ti GPU with 12 GB memory.

Table 2. Specific parameters of the proposed TRAL used in experiment.

MLP and Attention Learning	embedding size of tags	32
	embedding size of tag frequency	8
	size of hidden layers	[80, 40, 20]
	attention factor	16
Training Process	optimizer	Adam
	learning rate	0.001
	maximum number of iterations	300
	batch size	256
	regularization	L_2 norm
	number of negative samples	4

5.3. Performance Comparison

Experimental results of eight recommendation models on two public datasets are presented in Table 3, where we show the best results in boldface and best baseline results in underline. Additionally, we calculate the improvement (imp.) of the proposed model compared to the best baseline. It is clear that the proposed TRAL consistently surpasses other approaches in all evaluation metrics. For example, the performance of TRAL achieves 5.9% and 2.6% improvement over the best baseline in the light of HR@10 and NDCG@10 on Delicious.

Table 3. Overall performance comparison on two datasets in terms of HR and NDCG metrics. Boldface represents the highest score and underline denotes the best result of the baselines. The improvement (imp.) of the proposed model compared to the best baseline is calculated.

Dataset	**Delicious**				**LastFM**			
Metrics	**HR@10**	**NDCG@10**	**HR@20**	**NDCG@20**	**HR@10**	**NDCG@10**	**HR@20**	**NDCG@20**
CCF	0.6103	0.3851	0.6346	0.4123	0.5420	0.2548	0.5618	0.2652
ACF	0.6524	0.4216	0.6812	0.4520	0.5624	0.2651	0.5816	0.2764
NCF	0.6836	0.4435	0.6970	0.4712	0.5961	0.2872	0.6108	0.3056
DSPR	0.8041	0.6182	0.8166	0.6315	0.6854	0.3125	0.7012	0.3204
TRSDL	0.7925	0.6012	0.8104	0.6214	0.7052	0.3356	0.7126	0.3468
BPR-T	0.7123	0.5556	0.7492	0.5725	0.6532	0.3173	0.6750	0.3325
STEM	0.7458	0.5423	0.7643	0.5680	0.6726	0.3027	0.6953	0.3252
TRAL	**0.8518**	**0.6345**	**0.8618**	**0.6505**	**0.7336**	**0.3621**	**0.7582**	**0.3820**
Imp.	5.9%	2.6%	5.5%	3.0%	4.0%	7.9%	6.4%	10.2%

It is worth noting that CCF performs worst in most metrics among these baselines, which adequately reveals that the clustering method is insufficient to capture accurate abstract representation for users or items compared to deep learning strategies. In addition, the performance of DSPR, TRSDL, and TRAL are significantly better than ACF. The reason is that ACF only employs the autoencoder to capture low dimensional feature representations of users by constantly optimizing the reconstruction error. In contrast, other deep learning models directly optimize the correlation between users and items so as to extract more accurate feature representation. Furthermore, we can see that NCF behaves marginally better than ACF, but it is obviously worse than other deep learning recommendation methods with the help of tagging information. This convincingly proves the important role of tags as auxiliary information in improving the recommendation performance. The performance of BPR-T and STEM is obviously better than that of NCF, but worse than the proposed TRAL, which indicates that the deep learning model should be well designed for current problems to bring competitive results.

To summarize, the main reasons why the proposed model is superior to other baselines are as follows: (1) constructing the deep architecture to capture effective tag-based features; (2) exploiting the attention mechanism to distinguish the importance of different tag-based features adaptively; and (3) combining the deep and the wide component to learn the high-order and low-order interactions between user and item latent features.

5.4. Ablation Studies

In this section, we carry out several ablation studies on the proposed components or strategies in our model, including the effect of the attention network and the combination of the two components.

5.4.1. Effect of the Attention Network

To investigate the effect of the attention network, we replace it with average pooling to generate a variant method called TRAL-no-A. Figure 3 presents the recommendation results on two datasets in terms of HR and NDCG metrics. It can be observed that TRAL performs

significantly better than TRAL-no-A on both metrics, indicating that the recommendation performance will be seriously degraded without the attention network. Benefiting from the attention mechanism, different tag-based features are automatically compressed into a fixed-length vector with different weights. More accurate representations of users and items are thus derived to facilitate subsequent recommendations.

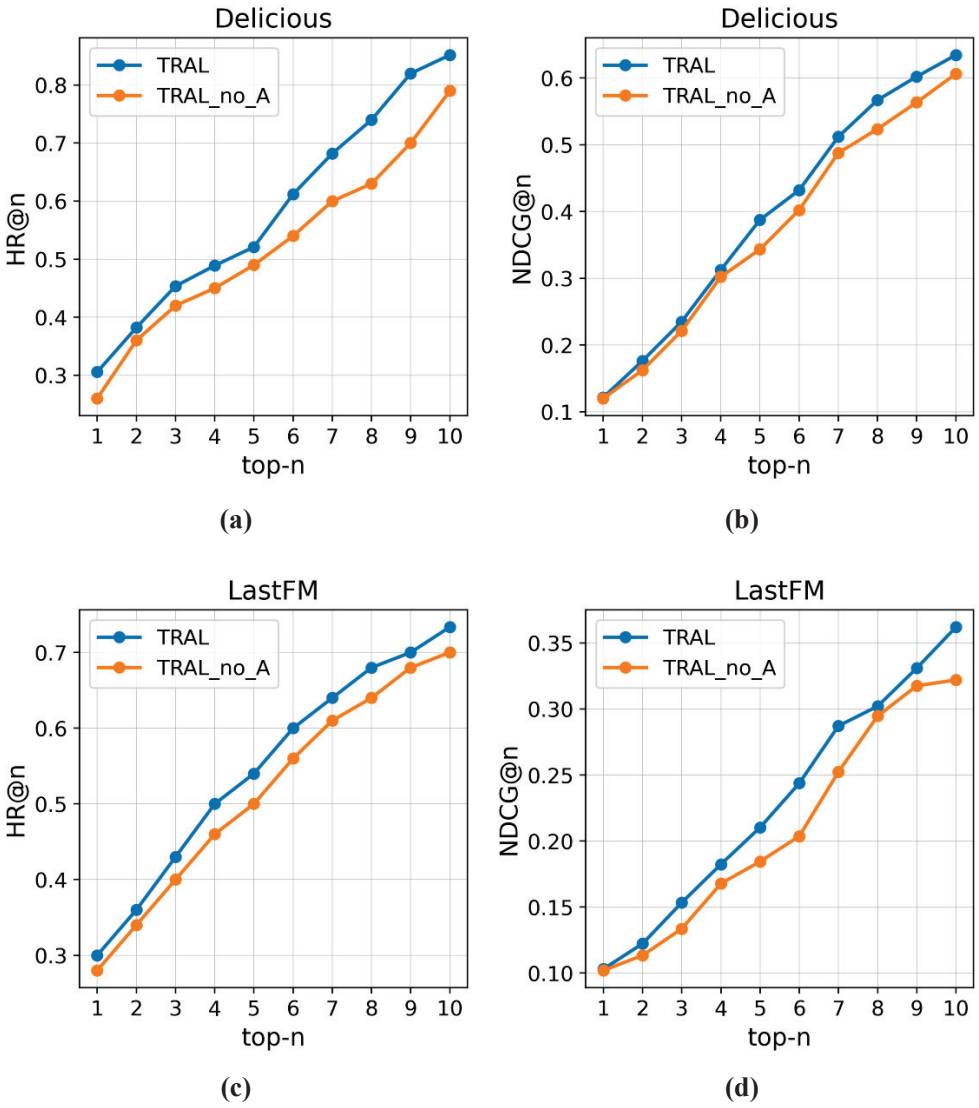


Figure 3. Performance comparison in terms of HR@n and NDCG@n on the two datasets. (**a**) HR@n on Delicious. (**b**) NDCG@n on Delicious. (**c**) HR@n on LastFM. (**d**) NDCG@n on LastFM.

As the model capability of the attention network is affected by the attention factor, we conducted several experiments to further study the impact of the attention mechanism. Figure 4 displays the results of different attention factors on the two datasets in terms of NDCG@10. Note that for Delicious and LastFM, the range of NDCG values under different attention factors falls within [0.62, 0.64] and [0.35, 0.37], respectively. The results show that the performance of TRAL is relatively stable across different attention factors on both datasets. It demonstrates that the design of attention network can make the model have strong robustness while improving the algorithm performance.

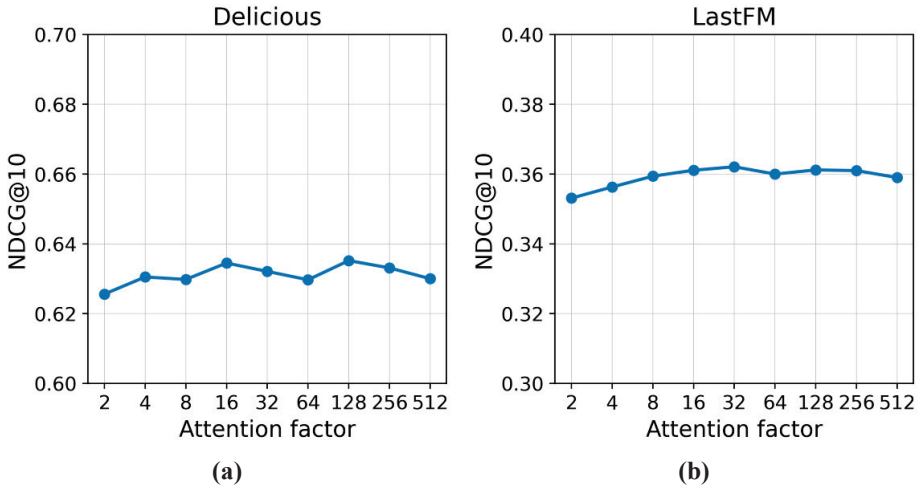


Figure 4. Recommendation results of different attention factors. (**a**) NDCG@10 on Delicious. (**b**) NDCG@10 on LastFM.

5.4.2. Effect of Combining the Two Components

To verify the effectiveness of combining the wide and the deep component, we compare the performance of the proposed model with that of its two variants, in which only the wide or the deep component is employed, named TRS-w and TRS-d, respectively. In addition, we also compare the results with the model using the wide component and the attention network, termed it as TRS-w-a. Table 4 presents the comparison results. It is clear that removing either the wide or the deep component will lead to a significant decline in algorithm performance, revealing the rationality of combining the two components. Among the three variants, TRS-d achieve the best results, which is due to the better expression ability of deep learning. Note that TRS-w performs slightly worse than TRS-w-a, indicating the positive effect of the attention mechanism.

Table 4. Ablation results on the two datasets.

Dataset	**Delicious**		**LastFM**	
Metrics	**HR@10**	**NDCG@10**	**HR@10**	**NDCG@10**
TRS-w	0.6125	0.3420	0.5671	0.2683
TRS-d	0.8093	0.5052	0.6924	0.3458
TRS-w-a	0.6340	0.3654	0.6021	0.2735
TRAL	0.8518	0.6345	0.7336	0.3621

5.5. Parameter Analysis

5.5.1. Number of Negative Samples

To examine the influence of the number of negative samples on recommendation performance, we search the number ranging from 1 to 10. Figure 5 displays the experiment results on Delicious and LastFM in terms of NDCG@10. In addition, the results of NCF and DSPR are also plotted. We can see that the performance of the proposed TRAL is significantly better than NCF and DSPR for different numbers of negative samples. It is worth noting that there is no fixed optimal value for all datasets or all models. When only one negative sample is used for each positive sample, the recommended performance is obviously not good enough, while too many samples will lead to performance degradation. A suitable number of negative samples is around 3 to 6. In our work, we set the number to 4, which is also used in the previous experiments [11].

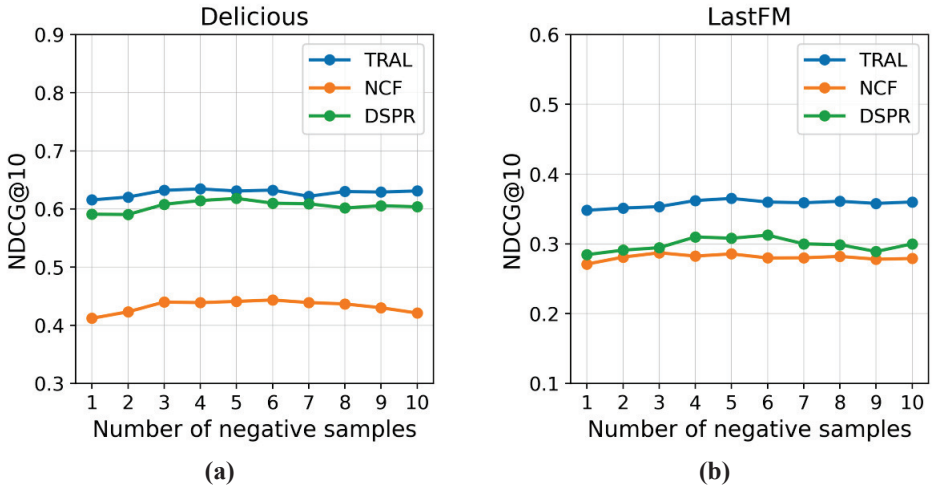


Figure 5. Recommendation results of different number of negative samples. (**a**) NDCG@10 on Delicious. (**b**) NDCG@10 on LastFM.

5.5.2. Embedding Size

To investigate the impact of the embedding size, we establish experiments by varying its value in the range of {8, 16, 32, 64, and 128}. Experimental results are summarized in Figure 6. From the results, we can given several observations. When the embedded dimension is relatively small, the expression ability is insufficient to model the interactions between users and items. With the increase in the embedding size, the performance of the proposed model is gradually improved. However, after the dimension increases to a certain value, the model cannot achieve significant improvement. In particular, we even find slight performance degradation on LastFM when using a large value of embedding size. To balance the performance and the computational cost, we set the embedding size to 32 in our work.

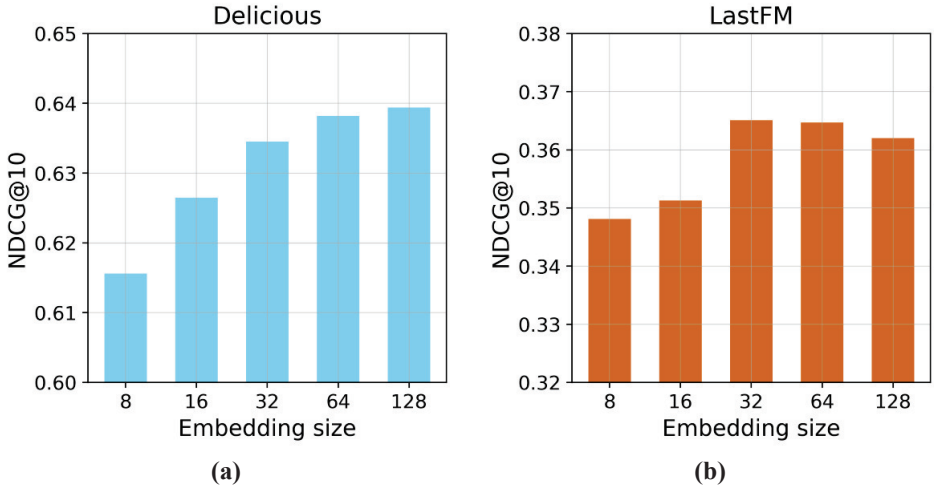


Figure 6. Recommendation results of different embedding sizes. (**a**) NDCG@10 on Delicious. (**b**) NDCG@10 on LastFM.

6. Conclusions

In social tagging systems, a great variety of tags are utilized to describe item characteristics and user preferences. In order to effectively handle tagging information, we propose a novel tag-aware recommendation model based on attention learning. The proposed model constructs a deep component to extract high-level interaction features by integrating an attention network and a multi-layer fully connected network. The aim of the attention network is to make different features contribute differently according to their

importance. Furthermore, we establish a wide component to capture low-level interaction features. By combining these different features, we can obtain more accurate representation for users and items, thus improving the recommendation performance. Experimental results demonstrated that the proposed model performs significantly better than other comparison algorithms.

However, there may be some limitations. Firstly, the proposed model is capable of addressing tagging information effectively when these tags are sufficient to express the accurate user interests and item characteristics. Unfortunately, the tagging behavior data in the real recommendation scenario is rather sparse. Secondly, the proposed model cannot deal with the cold-start problem, which refers to how to provide recommendations for new users or fresh items. The obvious reason is that these new users or items have no tagging information at all. Thirdly, the proposed model ignores the sequence information of users, which can indicate the drift of user interests.

To further improve the proposed model, future research directions thus focus on the following aspects. (1) We will investigate some new techniques achieve appropriate data augmentation, such as tag expansion [22] and graph data augmentation [39]. (2) To solve the cold-start problem, we can combine other side information, including images, texts and social relations. Moreover, the introduction of additional information will boost the recommendation performance if the hybrid algorithm is well designed. (3) In order to accurately capture the changes of users' interests over time, we need to design new component which can extract sequence information.

Author Contributions: The overall study supervised by S.L.; Methodology, hardware, software, and preparing the original draft by Y.Z. (Yi Zuo); Review and editing by Y.Z. (Yun Zhou) and H.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by National Natural Science Foundation of China (Grant no. 61902117 and Grant no. 72073041), the National Natural Science Foundation of Hunan Province (Grant No. 2020JJ5010), and Scientific research project of Hunan Provincial Department of Education (Grant no. 22A0667).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data that support the findings of this study are available from the corresponding author, upon reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bobadilla, J.; Ortega, F.; Hernando, A.; Gutiérrez, A. Recommender systems survey. *Knowl.-Based Syst.* **2013**, *46*, 109–132. [CrossRef]
2. Zhang, Z.K.; Zhou, T.; Zhang, Y.C. Tag-aware recommender systems: A state-of-the-art survey. *J. Comput. Sci. Technol.* **2011**, *26*, 767–777. [CrossRef]
3. Shepitsen, A.; Gemmell, J.; Mobasher, B.; Burke, R. Personalized recommendation in social tagging systems using hierarchical clustering. In Proceedings of the 2008 ACM Conference on Recommender Systems, Lausanne, Switzerland, 23–25 October 2008; pp. 259–266.
4. Tso-Sutter, K.H.; Marinho, L.B.; Schmidt-Thieme, L. Tag-aware recommender systems by fusion of collaborative filtering algorithms. In Proceedings of the 2008 ACM Symposium on Applied Computing, Fortaleza, Brazil, 16–20 March 2008; pp. 1995–1999.
5. Zuo, Y.; Zeng, J.; Gong, M.; Jiao, L. Tag-aware recommender systems based on deep neural networks. *Neurocomputing* **2016**, *204*, 51–60. [CrossRef]
6. Miller, G.A. WordNet: A lexical database for English. *Commun. ACM* **1995**, *38*, 39–41. [CrossRef]
7. Zhao, S.; Du, N.; Nauerz, A.; Zhang, X.; Yuan, Q.; Fu, R. Improved recommendation based on collaborative tagging behaviors. In Proceedings of the 13th International Conference on Intelligent User Interfaces, Gran Canaria, Spain, 13–16 January 2008; pp. 413–416.
8. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Netw.* **2015**, *61*, 85–117.

9. Xu, Z.; Lukasiewicz, T.; Chen, C.; Miao, Y.; Meng, X. Tag-aware personalized recommendation using a hybrid deep model. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, Melbourne, Australia, 19–25 August 2017.
10. Liang, N.; Zheng, H.T.; Chen, J.Y.; Sangaiah, A.K.; Zhao, C.Z. Trsdl: Tag-aware recommender system based on deep learning–intelligent computing systems. *Appl. Sci.* **2018**, *8*, 799. [CrossRef]
11. He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; Chua, T.S. Neural collaborative filtering. In Proceedings of the 26th International Conference on World Wide Web, Perth, Australia, 3–7 April 2017; pp. 173–182.
12. Cheng, H.T.; Koc, L.; Harmsen, J.; Shaked, T.; Chandra, T.; Aradhye, H.; Anderson, G.; Corrado, G.; Chai, W.; Ispir, M.; et al. Wide & deep learning for recommender systems. In Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, Boston, MA, USA, 15 September 2016; pp. 7–10.
13. Guo, H.; Tang, R.; Ye, Y.; Li, Z.; He, X. DeepFM: A factorization-machine based neural network for CTR prediction. *arXiv* **2017**, arXiv:1703.04247.
14. Nakamoto, R.; Nakajima, S.; Miyazaki, J.; Uemura, S. Tag-based contextual collaborative filtering. *IAENG Int. J. Comput. Sci.* **2007**, *34*, 2.
15. Marinho, L.B.; Schmidt-Thieme, L. Collaborative tag recommendations. In *Data Analysis, Machine Learning and Applications*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 533–540.
16. Zhen, Y.; Li, W.J.; Yeung, D.Y. TagiCoFi: Tag informed collaborative filtering. In Proceedings of the Third ACM Conference on Recommender Systems, New York, NY, USA, 23–25 October 2009; pp. 69–76.
17. Chen, C.; Zheng, X.; Wang, Y.; Hong, F.; Chen, D. Capturing semantic correlation for item recommendation in tagging systems. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, Arizona, 12–17 February 2016; Volume 30.
18. Wang, Z.; Deng, Z. Tag recommendation based on bayesian principle. In Proceedings of the International Conference on Advanced Data Mining and Applications, Chongqing, China, 19–21 November 2010; Springer: Berlin/Heidelberg, Germany, 2010; pp. 191–201.
19. Symeonidis, P.; Nanopoulos, A.; Manolopoulos, Y. Tag recommendations based on tensor dimensionality reduction. In Proceedings of the 2008 ACM Conference on Recommender Systems, Lausanne, Switzerland, 23–25 October 2008; pp. 43–50.
20. Li, H.; Diao, X.; Cao, J.; Zhang, L.; Feng, Q. Tag-aware recommendation based on Bayesian personalized ranking and feature mapping. *Intell. Data Anal.* **2019**, *23*, 641–659. [CrossRef]
21. Zhang, Z.K.; Zhou, T.; Zhang, Y.C. Personalized recommendation via integrated diffusion on user–item–tag tripartite graphs. *Phys. A Stat. Mech. Its Appl.* **2010**, *389*, 179–186. [CrossRef]
22. Pan, Y.; Huo, Y.; Tang, J.; Zeng, Y.; Chen, B. Exploiting relational tag expansion for dynamic user profile in a tag-aware ranking recommender system. *Inf. Sci.* **2021**, *545*, 448–464. [CrossRef]
23. Pan, X.; Zeng, X.; Ding, L. Topic optimization–incorporated collaborative recommendation for social tagging. *Data Technol. Appl.* **2022**, 1–20. [CrossRef]
24. Xu, Z.; Chen, C.; Lukasiewicz, T.; Miao, Y.; Meng, X. Tag-aware personalized recommendation using a deep-semantic similarity model with negative sampling. In Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, Indianapolis, IN, USA, 24–28 October 2016; pp. 1921–1924.
25. Huang, R.; Wang, N.; Han, C.; Yu, F.; Cui, L. TNAM: A tag-aware neural attention model for Top-N recommendation. *Neurocomputing* **2020**, *385*, 1–12. [CrossRef]
26. Chen, B.; Ding, Y.; Xin, X.; Li, Y.; Wang, Y.; Wang, D. AIRec: Attentive intersection model for tag-aware recommendation. *Neurocomputing* **2021**, *421*, 105–114. [CrossRef]
27. Ahmadian, S.; Ahmadian, M.; Jalili, M. A deep learning based trust-and tag-aware recommender system. *Neurocomputing* **2022**, *488*, 557–571. [CrossRef]
28. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed representations of words and phrases and their compositionality. *arXiv* **2013**, arXiv:1310.4546. https://doi.org/10.48550/arXiv.1310.4546.
29. Liu, H. Resource recommendation via user tagging behavior analysis. *Clust. Comput.* **2019**, *22*, 885–894. [CrossRef]
30. Glorot, X.; Bordes, A.; Bengio, Y. Deep sparse rectifier neural networks. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics. JMLR Workshop and Conference Proceedings, Fort Lauderdale, FL, USA, 11–13 April 2011; pp. 315–323.
31. Xiao, J.; Ye, H.; He, X.; Zhang, H.; Wu, F.; Chua, T.S. Attentional factorization machines: Learning the weight of feature interactions via attention networks. *arXiv* **2017**, arXiv:1708.04617.
32. Zhou, G.; Zhu, X.; Song, C.; Fan, Y.; Zhu, H.; Ma, X.; Yan, Y.; Jin, J.; Li, H.; Gai, K. Deep interest network for click-through rate prediction. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 1059–1068.
33. Cantador, I.; Brusilovsky, P.; Kuflik, T. Second workshop on information heterogeneity and fusion in recommender systems (HetRec2011). In Proceedings of the Fifth ACM Conference on Recommender Systems, Chicago, IL, USA, 23–27 October 2011; pp. 387–388.
34. Bayer, I.; He, X.; Kanagal, B.; Rendle, S. A generic coordinate descent framework for learning from implicit feedback. In Proceedings of the 26th International Conference on World Wide Web, Perth, Australia, 3–7 April 2017; pp. 1341–1350.
35. Elkahky, A.M.; Song, Y.; He, X. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In Proceedings of the 24th International Conference on World Wide Web, Florence, Italy, 18–22 May 2015; pp. 278–288.

36. Koren, Y. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, NV, USA, 24–27 August 2008; pp. 426–434.
37. Fayyaz, Z.; Ebrahimian, M.; Nawara, D.; Ibrahim, A.; Kashef, R. Recommendation systems: Algorithms, challenges, metrics, and business opportunities. *Appl. Sci.* **2020**, *10*, 7748. [CrossRef]
38. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
39. Han, X.; Jiang, Z.; Liu, N.; Hu, X. G-Mixup: Graph Data Augmentation for Graph Classification. *arXiv* **2022**, arXiv:2202.07179.