applied sciences

MDPI

Article

TRSDL: Tag-Aware Recommender System Based on Deep Learning–Intelligent Computing Systems

Nan Liang [1], Hai-Tao Zheng [1,*], Jin-Yuan Chen [1], Arun Kumar Sangaiah [2] and Cong-Zhi Zhao [3]

[1] Tsinghua-Southampton Web Science Laboratory, Graduate School at Shenzhen, Tsinghua University, Shenzhen 518055, China; liangn15@mails.tsinghua.edu.cn (N.L.); jy-chen13@mails.tsinghua.edu.cn (J.-Y.C.)
[2] School of Computing Science and Engineering, Vellore Institute of Technology (VIT), Vellore-632014, Tamil Nadu, India; sarunkumar@vit.ac.in
[3] Shenzhen Giiso Information Technology Co. Ltd., Sciencetific Research Building, Tsinghua Hi-Tech Park, Shenzhen 518057, China; zhaocz@giiso.com
* Correspondence: zheng.haitao@sz.tsinghua.edu.cn; Tel.: +86-180-3815-3089

Received: 7 April 2018; Accepted: 14 May 2018; Published: 16 May 2018

Abstract: In recommender systems (RS), many models are designed to predict ratings of items for the target user. To improve the performance for rating prediction, some studies have introduced tags into recommender systems. Tags benefit RS considerably, however, they are also redundant and ambiguous. In this paper, we propose a hybrid deep learning model TRSDL (tag-aware recommender system based on deep learning) to improve the performance of tag-aware recommender systems (TRS). First, TRSDL uses pre-trained word embeddings to represent user-defined tags, and constructs item and user profiles based on the items' tags set and users' tagging behaviors. Then, it utilizes deep neural networks (DNNs) and recurrent neural networks (RNNs) to extract the latent features of items and users, respectively. Finally, it predicts ratings from these latent features. The model not only addresses tag limitations and takes advantage of semantic tag information but also learns more advanced implicit features via deep structures. We evaluated our proposed approach and several baselines on MovieLens-20 m, and the experimental results demonstrate that TRSDL significantly outperforms all the baselines (including the state-of-the-art models BiasedMF and I-AutoRec). In addition, we also explore the impacts of network depth and type on model performance.

Keywords: deep learning; machine learning; neural networks; rating prediction; recommender systems; tag-aware recommendations

1. Introduction

In the era of information overload, deriving effective content from large volumes of information for Internet users has become an urgent problem. Recommender systems (RS) are regarded as integral to solving this problem, as they efficiently analyze the interests, preferences and needs of individual users and then provide personalized services to them. Many techniques for generating personalized recommendations have been proposed. For rating prediction task, one of the branches of recommendation, traditional algorithms focus on users' ratings of items but do not take item properties or user evaluations into account. In such cases, tag-aware recommender systems (TRS) are used to improve performance. Tags provide valuable supplementary information for RS, as they summarize items' properties and reflect user preferences through tagging behaviors.

While tags benefit recommender systems considerably, they also have unfavorable features. A tagging system allows users to tag items using arbitrary words or phrases, which creates two common issues: (1) redundancy, i.e., different users may use different tags to express similar concepts, such as “pretty” and “beautiful”; and (2) ambiguity, i.e., the same tags may have different meanings. For example, the word "train" has two meanings: a type of transportation or the teaching of skills through practice and instruction. These issues degrade the accuracy of TRS, as they prevent users from finding relevant information when ambiguous tags are used because common features are omitted when tags are represented in different ways.

One traditional solution utilizes clustering in the tag space [1]. However, it is difficult to compute similarities between tags when tag space is very sparse. Another traditional solution computes the similarity between two users’ tag sets by using the tool WordNet [2] to represent the similarity between the two users [3] . WordNet is an online English lexical database. It splits all entries into many groups according to their meanings. However, weaknesses of WordNet are obvious: it is time-consuming and labor-intensive to define the dictionary manually. Besides, words of the dictionary not only are limited, but also cannot keep up with the changes in language.

In our proposed model, we use Word2Vec [4], a tool for mapping words to K-dimensional distributed vectors, to represent tags with word embeddings. This method captures the semantic information of tags and simplifies the complex processing of text contents into the computing of vectors. At the same time, the weaknesses of tags are addressed because the distance of two vectors can be used to represent their semantic similarities between texts. On the other hand, deep learning has developed rapidly over the past decades and has achieved success in a variety of tasks, including those involving recommender systems [5]. Most machine learning algorithms used in traditional recommender systems are shallow structures that are limited in their use of complex data. The structures of deep neural networks render them better able to extract more abstract and representative latent features, improving the accuracy and generalization ability of learning models. In addition, consider a situation in one’s daily life: after watching the movie entitled “Cold War 1”, one is likely to search for the movie “Cold War 2” or for another Leung Ka Fai movie to watch. According to this perspective, users’ preferences evolve over time, and this evolving trajectory follows certain inner logic. Motivated by this, we organize the historical tagging records of target users by time, and then leverage recurrent neural networks (RNN) to extract users’ hidden features to take full advantage of the time-sensitive preferences of users. Recently, deep models focus on hashtag recommendations [6–8] and tag-aware top-n recommendation [9,10] are also proposed. However, as far as we know, research that apply sequential deep learning to tag-aware rating prediction remains rare, and there is room for improvement.

In summary, we propose a tag-aware recommender system based on deep learning (TRSDL) for rating prediction task. Firstly, the model uses pre-trained word embeddings to represent user-defined tags so that it not only addresses problems caused by tags but also makes full use of tags’ semantic information. Secondly, it constructs item and user profiles based on the item’s tags and the user’s tagging behaviors, and then utilizes deep neural networks (DNNs) and recurrent neural networks (RNNs) to extract the latent features of the item and the user, respectively. Finally, we use a forward neural network to simulate matrix factorization (MF) [11] to predict ratings from latent features. Relative to other tag-aware recommendation methods, the main contributions of the proposed model are as follows:

1. We use the pre-trained Word2Vec to represent tags, instead of the bag-of-words (BOW) model that is used in many TRS. By using word embeddings of tags, TRSDL alleviates problems of tag redundancy and ambiguity, and takes advantage of tags’ semantic information at the same time. Besides, it reduces the dimensionality of BOW-based representations so that it speeds up learning process and uses less memory resources.
2. Users’ preferences evolve over time and are influenced by their historical behaviors. By organizing the user’s tagging records chronologically, and using the excellent ability of the recurrent neural networks to process temporal sequence, we obtain more useful user dynamic preferences to enhance RS’s performance.

3. Interpret ability of deep structure methods is more powerful than shallow structure in the face of complex data. We model items characteristics and users preferences through deep learning algorithms, which captures the complex nonlinear relationships within the data, and extracts high-level abstract features.

The rest of this paper is organized as follows. Section 2 reviews related work. In Section 3, the proposed model is described in detail, and we present the results of our experiments in Section 4. Finally, a conclusion and future work are given in Section 5.

2. Related Work

Machine learning has a wide range of applications in various fields [12–14]. As our present paper focuses on tag-aware recommender systems based on deep learning, we introduce the development of tag-aware recommender systems and the application of deep learning to recommender systems, respectively. In the last subsection, we present the most technically relevant work of TRSDL and make a comparative analysis.

2.1. Traditional Tag-Aware Recommender Systems

Many traditional methods can be applied to TRS. Among them, collaborative filtering (CF) is the most widely used technique. Extending the CF to TRS, [15] Nakamoto et al. introduced a contextual CF model which takes tagging information into consideration when calculating the user similarity between users and predicting ratings for the user. Another work [16] was proposed by Mariho et al., who reduced the ternary relation of the *user-item-tag* to a lower-dimensional space, and then used CF to generate recommendation results. Based on this model, Ricci [17] developed a similar algorithm that finds user neighbors via the user-tag matrix and then recommends items liked by these neighbors to the target user. Karen et al. [18] proposed a generic method that allows tags to be incorporated to standard CF methods, by reducing the three-dimensional correlations to three two-dimensional correlations and then applying a fusion method to reassociate these correlations.

To solve the problem of redundancy and ambiguity caused by tags, other kinds of solutions have been applied to TRS as well, such as clustering, tensor-based methods, graph-based methods, etc. In [1], Shepitsen et al. presented a personalized recommendation that relies on hierarchical tag clusters. Through clustering technique, redundant tags are aggregated, and a cluster is more easily detected than a single tag. Wetzker et al. proposed an algorithm [19] to exploit the semantic contribution of tags. They extended the probabilistic latent semantic analysis (PLSA) approach and presented a unified recommendation model that evolving from co-occurrence theory. Szomszor et al. proposed a method based on semantic web [20], they constructed a movie knowledge base and then predict ratings by the defined rating tag-clouds. Sumeonidis et al. [21] developed a general framework by applying a semantic analysis and dimension reduction to the tensor *user-item-tag* via higher order singular value decomposition (HOSVD). Rendle [22] proposed a tensor factorization algorithm RTF (ranking with tensor factorization) that improves optimization approaches. Hotho et al. were also inspired by the Google PageRank [23] algorithm and in turn proposed the FolkRank [24] algorithm, according to which users, items and tags reinforce one another's weights through the spreading of weights, from which items with higher weights are recommended. In [25], Zhang et al. integrated tagging information into *user-item* bipartite graphs and proposed a model based on *user-item-tag* tripartite graphs. PITF [26] is optimized from the Bayesian personalized ranking [27] criterion. It models the pair-wise interactions among users, items and tags in lower dimensional matrices to decrease noise. Similarly, Wang et al. proposed a probabilistic model BTR (Bayesian-based Tag Recommendation algorithm) [28] to turn the tag recommendation task into a probability prediction problem so that a classical Bayesian method could be easily applied.

Compared with TRSDL, firstly, these models did not take advantage of word embeddings to overcome the problem caused by tags; secondly, they still use shallow structures, while the ability of interpreting complex nonlinear relationships within data is weaker than those based on deep structures. Furthermore, they did not take into account the temporal sequence of the user's tagging behaviors, which may improve the recommendation performance.

2.2. *Recommender Systems Based on Deep Learning*

Salakhutdinov et al. introduced the two-layer restricted Boltzmann machines (RBM) [29] for modeling ratings that slightly outperform carefully tuned SVD models but that still face cold start problems. To address such problem, collaborative deep learning (CDL), proposed in [30], integrates a Bayesian stack denoise auto-encoder (SDAE) [31] and a collaborative topic regression (CTR) [32] to learn items' and users' latent vectors, respectively. In terms of the ability of generating ranked recommendation lists, Collaborative Deep Ranking (CDR) [33], an improved pair-wise model, performs better than the point-wise CDL. To solve the sparsity problem of ratings, Kim et al. proposed convolutional matrix factorization (ConvMF) [34] which integrates CNN into probabilistic matrix factorization (PMF) to enhance the prediction accuracy.

Sedhain et al. presented AutoRec [35], a novel model based on AutoEncoder. AutoRec takes user or item vectors as input and reconstructs them in the output layer. The values in the reconstructed vectors are the predicted ratings of the corresponding position. Along this direction, many improved models based on AutoEncoder were proposed [36–38]. Besides computer vision, convolutional neural network is also widely used in natural language processing and other fields such as clickbait detection [39] and so on. Van et al. [40] proposed a model that applies convolutional neural networks (CNN) in music recommendations to capture advanced features from music signals. In [41], RNN was applied in a session-based recommendation system to predict the next-step behavior based on the user's behavior records. Subsequently, a series of variants of the session-based recommendation were proposed [42–44]. Then, recommendation systems for modeling the dynamic preferences of users were put forward [45,46]. Wu et al. proposed a Recurrent Recommender Networks (RRN) [47] based on the RNN, which was able to model the seasonal evolutions of items and changes of user preferences over time. Moreover, DeepFM [48] was proposed to integrate factorization machine(FM) and deep neural networks. It was able to model the high-order feature interactions via DNNs and low-order interactions via FM. IRGAN [49] is the original model which applies GAN to three information retrieval tasks: web search, item recommendation and question answering. Google proposed a general framework of wide and deep learning [32]—jointly trained wide linear models and deep neural networks—that combines the benefits of memorization and generalization for recommender systems.

The studies above are about recommender systems based on deep learning. They have achieved excellent results in their respective subdivision research areas. However, these models have not introduced tags into recommender systems. We believe that tags provide additional information for reflecting user preferences and the item characteristics, which can of further enhance the RS performance.

2.3. *Tag-Aware Recommender Systems Based on Deep Learning*

In this section, we introduce several methods similar to TRSDL and make a brief comparative analysis. In [6], Tomar et al. presented an approach to recommend hashtags for tweets. They made use of Word2Vec to represent tweets and trained a deep forward neural network to recommend corresponding hashtags for tweets. The representation of tags is similar to TRSDL, however, the model was not proposed for item recommendation or rating prediction task. Besides, the interaction of users and items profiles is not considered in their model, as well as temporal features.

Zuo [10] and Xu [9] proposed two tag-aware deep models for top-n recommendation, respectively. In both two models, tags are represented by the bag-of-words (BOW) model. Due to limitations of BOW, redundancy and ambiguity of tags is not well addressed, and abundant semantic information of tags is disregarded in the beginning. In addition, in both models, recommended items are generated by calculating the similarity of two vectors that come from the same mapping space. Zuo et al. calculated the similarity between users latent features to find nearest neighbors and then recommended their loved items, and Xu et al. calculated the similarity between users and items to find the most suitable recommend items for users. Our model uses a forward neural network to simulate MF to explore the feasibility of predicting ratings via two vectors from different deep feature space. Finally, our final task is rating prediction instead of Top-N recommendations.

3. Methodology

To model users and items using tags for rating prediction task, we propose TRSDL to learn latent features for users and items jointly using two different neural networks. The learned latent features are used to predict the corresponding ratings in a layer introduced on the top of both networks.

In this section, we describe the proposed model in more detail. Specifically, there are three main parts of this model: (1) Item Model, i.e., constructing item profiles over the item tags semantic space and extracting item latent features from deep neural networks; (2) User Model, i.e., constructing user profiles from corresponding item profiles and then using recurrent neural networks to discover the time-sensitive latent features of the user; and (3) Rating Prediction, where the learned latent features for the user and the item are used to predict the corresponding rating in a layer introduced on the top of both networks. Figure 1 provides an overview of the architecture of our model.

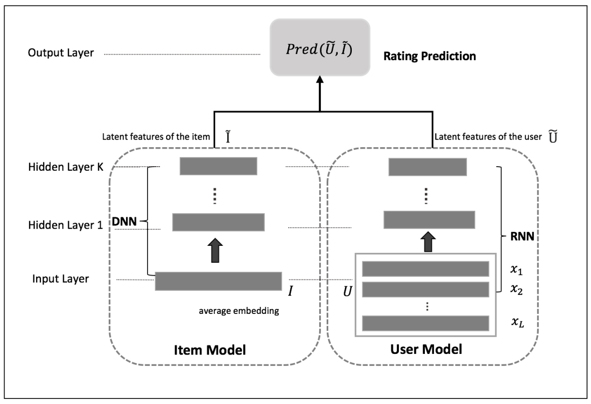

Figure 1. The overview of proposed model.

For convenience, we demonstrate types and descriptions of symbols in Table 1.

Table 1. Model notations.

Symbol	Description
UT	*User-Tag* matrix
IT	*Item-Tag* matrix
T	the tag consisting of tokens
G	the token list of Metadata
e^T_{avg}	average embeddings of tags
e^G_{avg}	average embeddings of metadata
d	dimensionality of the Word2Vec model
D	dimensionality of an item's embedding representation
M	the number of items in the training set
N	the number of users in the training set
$\tilde{I}$	latent features of the item
$\tilde{U}$	latent features of the user
L	the max RNN sequence length
z	the concatenated vector of user and item latent features
μ	the shift balance factor
β	the balance factor of L2 regularization
R	the space that users have rated on items
Y	predicted rating scores
J	the prediction loss function
Θ	the set of training parameters

3.1. Item Model

To capture semantic meaning existing in tags, TRSDL represents tags using pre-trained word embeddings. An item profile can be derived by modeling all tags and metadata (e.g., genres) of the target item with average word embeddings. The reason why we introduce metadata to the item representation is that it avoids items without any tags being denoted by all-zero vectors. Selected metadata is changing according to the dataset we used. Basically, it would be better if it is an enumerate feature (e.g., categories) that contains semantic information and can be represented with word embeddings.

Given a tag $T = \{t_1, t_2, \ldots, t_i, \ldots, t_n\}$, where n is the length of tokens after NLP preprocessing (tokenization, removal of stop words, etc.), t_i is the i-th token mentioned. Each token is converted into a d-dimensional vector e_i^T by looking up the word embedding table that can be initialized either by the pre-trained Word2Vec or by a random process, and then, the average embedding $e_{avg}^T = (\sum_{k=1}^{n} e_k^T)/n$ is used to denote the tag. If the item has not been tagged yet, we pad it with all-zero vector. Similarly, metadata G of the dataset are written as e_{avg}^G. Finally, embeddings are catenated into a single vector $X_i = [e_{avg}^T, e_{avg}^G]$ to present the i-th item. As a result, all items build a matrix of shape $[M, D]$, where M is the number of items and $D = 2 * d$ is the total length of embedding features.

A standard deep neural network is used to learn the latent features of items. Taking the item embeddings as an input, the hidden vector $\tilde{I}$ is computed as a weighted sum of input neurons as follows:

$$\tilde{I} = g(f(X_i)) = ReLU(W \cdot X_i + b) \tag{1}$$

where W is a weight matrix connecting an input layer to a hidden layer, b is a bias vector, and $g(\cdot)$ is a certain nonlinear activation function that uses $ReLU$ in the proposed model.

3.2. User Model

We build user profiles as the recurrent neural network [50] (RNN) input layer, and we then take hidden layer vectors as users' latent features.

RNN is a powerful model that learns temporal patterns from sequential data. In contrast to ordinary neural networks, the model analyzes a sequence step by step and stores the information of all previous steps in the hidden layer. This ability is dependent on the use of the special recurrent structure shown in Figure 2. As the figure shows, at step t, the hidden layer h_t is updated based on two sources: the current input x_t and its previous hidden state h_{t-1}. The formula is written as follows:

$$h_t = g(Px_t + Qh_{t-1}) \tag{2}$$

where $g(\cdot)$ is a nonlinear activation function and Q and P are the weights of the previous hidden state and the current input. All parameters of the networks are randomly initialized and tuned through training.

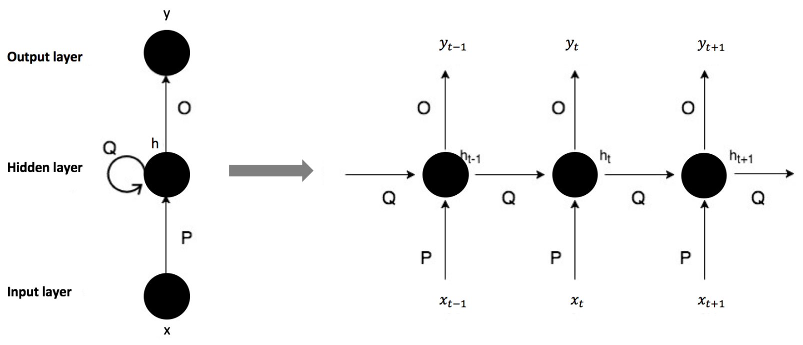

Figure 2. The recurrent structure of RNN.

In the proposed model, users' tagging behaviors can be observed as a temporal sequence. We arrange the latest L components of records according to tagging timestamps. Each item is denoted as embedding features of the Item Model. Although we use a dynamic RNN structure to address unfixed length records, we must still ensure that each user's input has a length of L by trimming longer ones and padding shorter ones with 0.

For a given user $U_i = [u_i^1, u_i^2, \dots, u_i^t, \dots, u_i^L]$, the user's representation is a matrix of size $L \times D$. Consequently, N users' embedding features form a 3-d array of $N \times L \times D$. Between the input users' profiles and the final prediction, there is a hidden layer with n_h units that stores additional information on records previously observed in the sequence. We take the last component of the valid hidden state as users' latent features $\tilde{U}$.

In the application of the user model, we use an LSTM [51] cell rather than an RNN cell to avoid missing information of a long sequence and to prevent gradients from vanishing during back propagation through time (BPTT). LSTM is an evolvement structure of RNN that addresses sequence problems in a way very similar to RNN. We apply the LSTM cell through TensorFlow.

3.3. Rating Prediction

The learned latent features for users and items are used to predict the corresponding ratings in a layer. In matrix factorization method, the inner product of user latent vectors u_i and item latent vectors v_j can approximate the rating r_{ij} [11],

$$r_{ij} \approx u_i \cdot v_j \tag{3}$$

Inspired by this, we use the interaction of two latent features learned from the hybrid deep structure to predict the main part of the rating. The rating of a target item given by a user is computed as following:

$$Y = pred(\tilde{U}, \tilde{I}) = \tilde{U} \cdot \tilde{I} + \mu * (ReLU(W_0 \cdot z + b_0)) \tag{4}$$

In the left-hand part, we use the inner product of latent features $\tilde{U}$ and $\tilde{I}$ to predict the rating. The right-hand part is a shift weighted by μ, z is a concatenated vector of $\tilde{U}$ and $\tilde{I}$, W_z is the weight used to model the strength of W_z, and b_z denotes the bias of the concatenated vector.

The loss function is the square error between real and predicted ratings. It is defined as:

$$J(\theta) = \sum_{r_{ui} \in R} (\hat{r_{ui}} - r_{ui})^2 + \beta ||\Theta||^2 \tag{5}$$

where $\hat{r_{ui}}$ is the predicted rating, r_{ui} is the real rating, Θ is the set of parameters to be trained and β is a balance factor of L2-regularization.

The training target is to minimize the loss function. Gradients are computed via back propagation through time, while training is done using an RMSProp [52] optimizer with shuffled mini-batches.

4. Experiments

4.1. Dataset

To demonstrate the effectiveness of the proposed model, we perform experiments through MovieLens (http://movielens.org).

The dataset (mL-20 m) contains 20,000,263 ratings and 465,564 tags for 27,278 movies. Each movie belongs to several genres selected from 19 categories. The data were generated by 138,493 users between 9 January 1995 and 31 March 2015. Each user has rated at least 20 movies and ratings range from 0.5 to 5.0.

To ensure the scale of the data we used is identical in all comparing experiments and reduce the interference of noise data, we selected samples created by users who have tagged items. After cutting down, the statistical descriptions of the dataset used are shown in Table 2.

We sorted samples according to the tagging timestamps and divided the dataset into two parts: 80% of the data were selected as the training set and the remaining 20% of the data constituted the testing set. The reason we choose an 80%/20% split is that the state-of-the-art TRS studies [9,10] used 80% of data as the training set, too. Meanwhile, we followed the original proportion of split (80%/20%) of mL-100 k although we used mL-20 m.

Table 2. A summary of the used dataset.

Item	Quantity
Users (u)	7159
Movies (i)	13,396
Ratings (r)	126,083
Tags (t)	34,065
Genres (g)	19

4.2. Measurements

We use the following metrics to evaluate the performance of the prediction algorithms: the root mean square error ($RMSE$) and the mean absolute error (MAE). It is preferable when indicator values are smaller. The formulas of $RMSE$ and MAE are as follows:

$$RMSE = \sqrt{\frac{\sum_{r_{ui} \in R} (\hat{r_{ui}} - r_{ui})^2}{|R|}} \tag{6}$$

$$MAE = \frac{\sum_{r_{ui} \in R} |\hat{r_{ui}} - r_{ui}|}{|R|} \tag{7}$$

where $|R|$ denotes the number of samples and $\hat{r_{ui}}$ and r_{ui} denote the predicted and real ratings of each sample, respectively.

4.3. Experimental Results and Discussions

We experiment with several variants of the model in the first part, and then evaluate TRSDL against some baselines in the second part.

4.3.1. Evaluation of Model Architectures

To evaluate the performance of choosing different network types (LSTM or DNN) and different network depths (1, 2 and 3), we employed a bottom-up approach [53] to create different models for comparison.

(1) In the first model, we learn both users' and items' latent features from two single-layer neural networks.
(2) The second one is built on Model (1)—we stack multiple hidden layers of both networks to verify whether the increase of layers can improve the model's performance.
(3) We apply a single-layer LSTM in the user model instead of the DNN of Model (1).
(4) In the last model, we stack multiple hidden layers based on the third model.

For clarity, we combined Models (1) and (2) as the first group of experiments, which is represented by "NNs", and combined Models (3) and (4) as the second group, which is denoted by "TRSDL". In the two groups, network layers are distinguished by the number shown after @. For the NNs experiments, user profiles are constructed from corresponding items' average embeddings rather than from chronological item embeddings as TRSDL.

For all baselines, we tuned the learning rate $lr \in \{0.001, 0.005, 0.01, 0.02, 0.05, 0.1\}$, the batch size $bs \in \{128, 256, 512\}$, regularization strength $\beta \in \{0, 0.001, 0.005, 0.01\}$ and the appropriate latent dimension $n_h \in \{16, 32, 64, 128, 256\}$.

The final hyper-parameters we used are listed in Table 3, where L denotes the max sequence length for LSTM, μ is the balance factor for the effects of hidden concatenated features. Furthermore, to prevent gradient explosion, we apply gradient clipping to TRSDL, and the max gradient norm is set to 5.

Table 3. Parameters of various model structures.

Structures	Settings
NNs@1	$lr = 0.01$, $n_{h1} = 64$, $L = 20$, $\mu = 0.1$, $\beta = 0$, $bs = 256$
NNs@2	$lr = 0.01$, $n_{h1} = 128$, $n_{h2} = 64$, $L = 20$, $\mu = 0.1$, $\beta = 0$, $bs = 256$
NNs@3	$lr = 0.01$, $n_{h1} = 128$, $n_{h2} = 64$, $n_{h3} = 16$, $L = 20$, $\mu = 0.1$, $\beta = 0$, $bs = 256$
TRSDL@1	$lr = 0.01$, $n_{h1} = 64$, $L = 20$, $\mu = 0.1$, $\beta = 0$, $bs = 256$
TRSDL@2	$lr = 0.01$, $n_{h1} = 128$, $n_{h2} = 64$, $L = 20$, $\mu = 0.1$, $\beta = 0$, $bs = 256$
TRSDL@3	$lr = 0.01$, $n_{h1} = 128$, $n_{h2} = 64$, $n_{h3} = 16$, $L = 20$, $\mu = 0.1$, $\beta = 0$, $bs = 256$

Experimental results of various model structures are shown in Table 4. From the table, the performances for several model architectures are quite clear and conclusive:

(a) TRSDL@3 provides the best results with an MAE of 0.658 and an $RMSE$ of 0.870, which are 5.18% and 3.87% lower than the best results of NNs, respectively. The worst performance of TRSDL group is from the single-layer structure TRSDL@1, with an MAE of 0.688 and an $RMSE$ of 0.905, which still outperforms all of the NNs' experiments.

(b) As the number of network layers increases, the MAE value of the NNs experiment decreases from 0.862 to 0.694 (the $RMSE$ value decreases from 1.104 to 0.905), and the MAE of the TRSDL group decreases from 0.688 to 0.658 (the $RMSE$ decreases from 0.905 to 0.870).

Table 4. Results of various model structures.

Structures	*MAE*	*RMSE*
NNs@1	0.862	1.104
NNs@2	0.711	0.934
NNs@3	0.694	0.905
TRSDL@1	0.688	0.905
TRSDL@2	0.670	0.881
TRSDL@3	**0.658**	**0.870**

Experiment (a) verified that the proposed TRSDL significantly outperforms all NNs model architectures. As one of the most important contributions, temporal information indeed enhances the model's performance. Features learned from tagging sequential information can be used to improve predictions, and LSTM is good at processing the sequential data.

Experiment (b) shows that the number of neural network layers plays an important role in the model behavior. In both two groups of experiments, the model based on single-layer structure is the worst performing. The performance improves when layers stack, as the lower layer captures surface characteristics of input data, while more advanced features are captured by upper layers. In other words, hidden features extracted from the shallow structure are less informative than the deep structure. Our proposed TRSDL is able to learn complex nonlinear relations within data through the deep interaction of latent features.

4.3.2. Comparative Results and Evaluations

To show the advantage of the proposed model TRSDL, we compare it with seven baselines. The baselines are: traditional algorithms based on collaborative filtering (i.e., ItemCF, UserCF and

TAG-CF), the state-of-the-art models for rating prediction (i.e., BiasedMF and I-AutoRec), and tag-aware recommender based on deep learning (i.e., DNN and BOW-TRSDL). We first describe the competing methods as follows:

- **ItemCF:** For every item of the test set, the weighted average ratings of the nearest items in the training set are returned. The cosine similarity is used to measure the degree of items' similarity.
- **UserCF:** Similar to the itemCF model, the nearest user neighbors' weighted average ratings are regarded as predictive ratings. User similarity is measured by cosine similarity.
- **TAG-CF:** This method integrates tag information to the collaborative filtering and proposes a tag-aware hybrid prediction algorithm [54]. The principle process is shown in the following steps.

  (1) Calculate item co-occurrence similarity $itemSim$ based on Item-Tag matrix T. (2) Establish a predictive rating matrix C based on item similarity. (3) Construct a pseudo-matrix M to integrate the rating matrix R and the predictive rating matrix C. (4) Calculate user similarity $userSim$ through M, the similarity is measured with Pearson correlation coefficient [55]. (5) Predict ratings based on the user similarity matrix. The rating formula is defined as:

$$\hat{r_{ui}} = \bar{r}_u + \frac{\sum_{v \in Neighbor(u)} userSim(u,v)(r_{vi} - \bar{r}_v)}{\sum_{v \in Neighbor(u)} userSim(u,v)} \tag{8}$$

- **BiasedMF:** BiasedMF is the state-of-the-art collaborative filtering technique [11]. Features of users and items are mapped to a latent factor space, and the interaction between user's latent vector p_u and item latent vector q_i is learned to predict ratings. The rating formula is defined as follows:

$$\hat{r_{ui}} = \mu + b_i + b_u + {q_i}^T p_u \tag{9}$$

  where b_i and b_u indicate the bias of the item i and the user u, respectively; and μ is the global average rating.
- **I-AutoRec:** I-AutoRec is a novel rating framework based on AutoEncoder [35]. It takes item vectors as input and reconstructs them in the output layer. The values in the reconstructed vectors are the predicted value of the corresponding position.

$$\hat{r_{ui}} = (h(r^i;\theta))_u \tag{10}$$

$$h(r;\theta) = (f(W \cdot g(Vr + \mu) + b)) \tag{11}$$

  where $h(r;\theta)$ is the reconstruction of input r, $\theta = \{W, V, \mu, b\}$ for transformations W, V, and biases μ, b. The parameters we used in experiments are: learning rate $lr = 0.001$, the number of hidden units $n_h = 500$ and the regularization strength $\lambda = 0.002$.
- **DNN:** In the DNN model, item profiles are the same as those of TRSDL, while user profiles are constructed from corresponding items' average embeddings rather than from chronological item embeddings. Ratings are predicted by latent features extracted from two parallel deep neural networks. The parameters of DNN model are the same as those of NNs@3.
- **BOW-TRSDL:** The state-of-the-art tag-aware recommenders are found in [9,10]. Unfortunately, we cannot compare with them directly because our ultimate targets are different (i.e., their application scenario is top-n recommendations). Considering the biggest difference is the way tags are represented during the profile constructions, we leverage the bag-of-words (BOW) model to build users' and items' profiles. The model is denoted as "BOW-TRSDL". In the model, tags used less than 10 times are cut down to reduce the amount of computation and the noise. Other parameters are the same as those of TRSDL.
- **TRSDL:** TRSDL is our proposed model. We utilize the DNNs to obtain items' latent features and leverage the LSTM to extract users' latent preferences from their temporal tagging sequences. Then, these hidden features are interacted to predict corresponding ratings. The parameters in the experiment are the same as those of TRSDL@3.

Table 5 depicts in detail prediction performances of TRSDL and other seven baselines on the MovieLens dataset in terms of MAE and $RMSE$. Observing the experimental results, we have the following findings and discussions.

Table 5. The comparison of MAE and $RMSE$ values for the dataset.

Models	MAE	$RMSE$
ItemCF	1.091	1.329
UserCF	1.237	1.481
TAG-CF	1.042	1.366
BiasedMF	0.673	0.904
I-AutoRec	0.703	1.057
DNN	0.735	1.011
BOW-TRSDL	0.736	0.971
TRSDL	**0.658**	**0.870**

Overall, TRSDL significantly outperforms other baselines in all metrics. The MAE and $RMSE$ of TRSDL are 0.658 and 0.870, followed by the best baseline BiasedMF, with an MAE of 0.673 and an $RMSE$ of 0.904. TRSDL reduced the prediction errors—MAE and $RMSE$ decreased by 2.23% and 3.76% compared to the suboptimal BiasedMF. The superior performance of TRSDL is mainly because that integrating tag representations based on word embeddings and users' tagging behaviors within a hybrid deep learning model improves the performance.

Table 5 demonstrates that, when comparing TRSDL to ItemCF, UserCF and TAG-CF models, TAG-CF and TRSDL perform better than ItemCF and UseCF in terms of indicator MAE. TAG-CF and TRSDL utilize additional tag information in recommender systems, which can be regarded as the good reflections of users' preferences and items' characteristics. The reasons that TRSDL achieves much higher improvement than TAG-CF may be that: On the one hand, TRSDL uses distributed word representations to encode tags instead of discrete vectors based on the bag-of-word model so that the model takes advantage of the semantic information of tags. Accordingly, not only users' preferences and opinions towards items are implicitly discovered, but also the problem of redundancy and ambiguity caused by user-defined tags is mitigated. On the other hand, TRSDL captures more general and abstract hidden features through deep networks. It enables complex and high-level relationships within data to be represented in deeper layers of the networks, and then effectively learns non-linear relationships among users, items and tags.

The results show that TRSDL outperforms two state-of-the-art algorithms, BiasedMF and I-AutoRec, proving that our proposed model is a competitive method of rating prediction studies. We also note that the performance of I-AutoRec is dramatically worse than BiasedMF, while I-AutoRec outperforms BiasedMF when it was proposed in [35]. We guess the phenomenon is related to the sparsity of the dataset. I-AutoRec works well on the MovieLens-1 m and MovieLens-10 m datasets. The sparsity of the two datasets is 95.754% and 98.692%, respectively, and the sparsity of the data we used (after preprocessing) reaches 99.551%. This also proves from the side that TRSDL is robust to the sparse data to some degree.

DNN and BOW-TRSDL models are similar to TRSDL. The difference between DNN and TRSDL is the type of networks in the user model (i.e., DNN uses a three-layer neural network to replace the three-layer LSTM of TRSDL), while the difference between BOW-TRSDL and TRSDL is the approach of tags representation (i.e., BOW-TRSDL represents tags based on the bag-of-words model instead of word embeddings). TRSDL performs better than DNN (the MAE and $RMSE$ decreased by 0.077 and 0.141, respectively), which verifies that learning the temporal features of users through the recurrent neural networks provides useful information for improving the recommendations. TRSDL outperforms BOW-TRSDL, indicating that the bag-of-words model cannot discriminate redundant tags with similar meanings or ambiguous tags with multiple meanings. In addition, the vector generated by the BOW model is very high dimensional which leads to huge resource consumption.

We analyze the variance and the two-tailed t-test between the performances of TRSDL and the three best baselines (BiasedMF, I-AutoRec and DNN) to verify whether the performance differences are statistically significant. Firstly, we analyze the statistical differences of MAE measurement. The sig value of Levene's test is 0.126 > 0.05, indicating that the variance is homogenous. After that, we conduct the one-way analysis of variance for all groups. The p-value is 8.21×10^{-14}, which means the difference is significant. Furthermore, we perform t-tests for TRSDL and baselines. The p-value of TRSDL-BaisedMF, TRSDL-I-AutoRec and TRSDL-DNN are 3.27×10^{-6}, 2.73×10^{-13} and 2.70×10^{-8}, respectively. All p-values are smaller than 0.05, illustrating the MAE differences are statically significant. p-values can verify whether the effect exists, however, they do not reveal the size of the effect. To overcome the limitation of p-values, we also report the effect size with the Cohen's d values [56]. The d values are 3.28, 10.22 and 4.72, respectively, all effect sizes are bigger than 0.8 indicating big effects. Secondly, the analysis of $RMSE$ is similar. The sig value of Leneve's test is 0.166 indicating the homogeneity of variance. The p-value from one-way variance analysis is 1.65×10^{-29}, which means the difference reaches the significant level. According to the t-tests between TRSDL and baselines, p-values obtained from TRSDL-BaisedMF, TRSDL-I-AutoRec and TRSDL-DNN are 6.61×10^{-7}, 1.10×10^{-20} and 2.75×10^{-7}, respectively. The corresponding Cohen's d values are 3.72, 30.06 and 3.98, respectively. All the analyses demonstrate that the $RMSE$ differences are statistically significant. In summary, these results show that the performance differences are statistically significant.

5. Conclusions

In the paper, we propose TRSDL as an innovative and effective hybrid deep learning model for tag-aware recommender systems. We use word embeddings to represent tags such that semantic information is utilized to construct the profiles of users and items. We also integrate forward and recurrent neural networks into a joint model to not only extract high-level latent features of users and items but also take advantage of temporal features of users' preferences.

We evaluate our proposed model on MovieLens-20m, and TRSDL significantly outperforms all the baselines. Experimental results demonstrate that TRSDL is effective and competitive for rating prediction task. It improves traditional collaborative filtering methods and performs better than the state-of-the-art models on this dataset.

We thus plan to investigate reasonable improvements to our model in the future. Future studies will involve: (1) improving the model such that it can be used without ratings data and evaluating the performance of the improved model on more complex real datasets with tags; and (2) using the improved model for top-N items recommendation and verifying that the model has a good generalization ability.

Author Contributions: N.L., H.-T.Z., J.-Y.C., A.K.S. and C.-Z.Z.; Methodology, N.L. and H.-T.Z.; Software, N.L.; Validation, N.L., H.-T.Z. and J.-Y.C.; Formal Analysis, N.L.; Investigation, N.L. and H.-T.Z.; Resources, H.-T.Z., A.K.S. and C.-Z.Z.; Data Curation, N.L. and J.-Y.C.; Writing Original Draft Preparation, N.L.; Writing Review and Editing, N.L., H.-T.Z., J.-Y.C., A.K.S. and C.-Z.Z.; Visualization, N.L.; Supervision, A.K.S.; Project Administration, C.-Z.Z.; Funding Acquisition, H.-T.Z.

Acknowledgments: This research is supported through the National Natural Science Foundation of China (Grant No. 61773229), the Shenzhen Science and Technology Project (Grant No. CYZZ20150408152315667), the Basic Scientific Research Program of Shenzhen City (Grant No. JCYJ20160331184440545), and the Graduate School Cross Fund of Shenzhen, Tsinghua University (Grant No. JC20140001).

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

References

1. 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.
2. Miller, G.A. WordNet: A lexical database for English. *Commun. ACM* **1995**, *38*, 39–41. [CrossRef]
3. 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.
4. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv* **2013**, arXiv:1301.3781. [CrossRef]
5. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
6. Tomar, A.; Godin, F.; Vandersmissen, B.; De Neve, W.; Van de Walle, R. Towards Twitter hashtag recommendation using distributed word representations and a deep feed forward neural network. In Proceedings of the 2014 International Conference on Advances in Computing, Communications and Informatics, New Delhi, India, 24–27 September 2014; pp. 362–368.
7. Zhang, Q.; Wang, J.; Huang, H.; Huang, X.; Gong, Y. Hashtag recommendation for multimodal microblog using co-attention network. In Proceedings of the 26th International Joint Conference on Artificial Intelligence, Melbourne, Australia, 19–25 August 2017; pp. 3420–3426.
8. Nguyen, H.T.; Wistuba, M.; Grabocka, J.; Drumond, L.R.; Schmidt-Thieme, L. Personalized Deep Learning for Tag Recommendation. In Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, Jeju, South Korea, 23–26 May 2017; pp. 186–197.
9. 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.
10. Zuo, Y.; Zeng, J.; Gong, M.; Jiao, L. Tag-aware recommender systems based on deep neural networks. *Neurocomputing* **2016**, *204*, 51–60. [CrossRef]
11. Koren, Y.; Bell, R.; Volinsky, C. Matrix factorization techniques for recommender systems. *Computer* **2009**, *42*, 30–37. [CrossRef]
12. Liu, S.; Pan, Z.; Cheng, X. A novel fast fractal image compression method based on distance clustering in high dimensional sphere surface. *Fractals* **2017**, *25*, 1740004. [CrossRef]
13. Lu, M.; Liu, S.; Kumarsangaiah, A.; Zhou, Y.; Pan, Z.; Zuo, Y. Nucleosome Positioning with Fractal Entropy Increment of Diversity in Telemedicine. *IEEE Access* **2017**. [CrossRef]
14. Liu, G.; Liu, S.; Muhammad, K. Object Tracking in Vary Lighting Conditions for Fog based Intelligent Surveillance of Public Spaces. *IEEE Access* **2018**. [CrossRef]
15. Nakamoto, R.; Nakajima, S.; Miyazaki, J.; Uemura, S. Tag-based contextual collaborative filtering. *IAENG Int. J. Comput. Sci.* **2007**, *34*, 214–219.
16. Marinho, L.B.; Schmidt-Thieme, L. Collaborative tag recommendations. In *Data Analysis, Machine Learning and Applications*; Springer: Heidelberg/Berlin, Germany, 2008; pp. 533–540.
17. Ricci, F.; Rokach, L.; Shapira, B. *Introduction to Recommender Systems Handbook*; Springer: Heidelberg/Berlin, Germany, 2011.
18. 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, Ceara, Brazil, 16–20 March 2008; pp. 1995–1999.
19. Wetzker, R.; Umbrath, W.; Said, A. A hybrid approach to item recommendation in folksonomies. In Proceedings of the WSDM'09 Workshop on Exploiting Semantic Annotations in Information Retrieval, Barcelona, Spain, 9 February 2009; pp. 25–29.
20. Szomszor, M.; Cattuto, C.; Alani, H.; O'Hara, K.; Baldassarri, A.; Loreto, V.; Servedio, V.D. Folksonomies, the semantic web, and movie recommendation. In Proceedings of the 4th European semantic web conference, Innsbruck, Australia, 3–7 June 2007; pp. 25–29.

21. 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.
22. Rendle, S.; Balby Marinho, L.; Nanopoulos, A.; Schmidt-Thieme, L. Learning optimal ranking with tensor factorization for tag recommendation. In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, 28 June–1 July 2009; pp. 727–736.
23. Page, L.; Brin, S.; Motwani, R.; Winograd, T. *The PageRank Citation Ranking: Bringing Order to the Web*; Technical Report; Stanford InfoLab: Stanford, CA, USA, 1999.
24. Hotho, A.; Jäschke, R.; Schmitz, C.; Stumme, G. Information retrieval in folksonomies: Search and ranking. In Proceedings of the European Semantic Web conference, Budva, Montenegro, 11–14 June 2006; Springer: Heidelberg/Berlin, Germany; pp. 411–426.
25. Zhang, Z.K.; Zhou, T.; Zhang, Y.C. Personalized recommendation via integrated diffusion on user–item–tag tripartite graphs. *Phys. A Stat. Mech. Appl.* **2010**, *389*, 179–186. [CrossRef]
26. Rendle, S.; Schmidt-Thieme, L. Pairwise interaction tensor factorization for personalized tag recommendation. In Proceedings of the Third ACM International Conference on Web Search and Data Mining, New York, NY, USA, 4–6 February 2010; pp. 81–90.
27. Rendle, S.; Freudenthaler, C.; Gantner, Z.; Schmidt-Thieme, L. BPR: Bayesian personalized ranking from implicit feedback. In Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, 18–21 June 2009; pp. 452–461.
28. 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: Heidelberg/Berlin, Germany; pp. 191–201.
29. Salakhutdinov, R.; Mnih, A.; Hinton, G. Restricted Boltzmann machines for collaborative filtering. In Proceedings of the 24th International Conference on Machine Learning, Corvalis, OR, USA, 20–24 June 2007; pp. 791–798.
30. Wang, H.; Wang, N.; Yeung, D.Y. Collaborative deep learning for recommender systems. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chongqing, China, 19–21 November 2010; pp. 1235–1244.
31. Slaney, M. Web-scale multimedia analysis: Does content matter? *IEEE MultiMedia* **2011**, *18*, 12–15. [CrossRef]
32. 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.
33. Ying, H.; Chen, L.; Xiong, Y.; Wu, J. Collaborative deep ranking: A hybrid pair-wise recommendation algorithm with implicit feedback. In Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, Boston, MA, USA, 15 September 2016; Springer: Heidelberg/Berlin, Germany; pp. 555–567.
34. Kim, D.; Park, C.; Oh, J.; Lee, S.; Yu, H. Convolutional matrix factorization for document context-aware recommendation. In Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, 15–19 September 2016; pp. 233–240.
35. Sedhain, S.; Menon, A.K.; Sanner, S.; Xie, L. Autorec: Autoencoders meet collaborative filtering. In Proceedings of the 24th International Conference on World Wide Web, Florence, Italy, 18–22 May 2015; pp. 111–112.
36. Zhang, S.; Yao, L.; Xu, X. Autosvd++: An efficient hybrid collaborative filtering model via contractive auto-encoders. *arXiv* **2017**, arXiv:1704.00551. [CrossRef]
37. Zhuang, F.; Zhang, Z.; Qian, M.; Shi, C.; Xie, X.; He, Q. Representation learning via Dual-Autoencoder for recommendation. *Neur. Netw.* **2017**, *90*, 83–89. [CrossRef] [PubMed]
38. Chen, W.; Zheng, H.T.; Mao, X.X. Extracting Deep Semantic Information for Intelligent Recommendation. In Proceedings of the International Conference on Neural Information Processing, Guangzhou, China, 14–18 November 2017; pp. 134–144.
39. Zheng, H.T.; Chen, J.Y.; Yao, X.; Sangaiah, A.K.; Jiang, Y.; Zhao, C.Z. Clickbait Convolutional Neural Network. *Symmetry* **2018**, *10*. [CrossRef]
40. Van den Oord, A.; Dieleman, S.; Schrauwen, B. Deep content-based music recommendation. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, Nevada, 5–10 December 2013; pp. 2643–2651.

41. Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; Tikk, D. Session-based recommendations with recurrent neural networks. *arXiv* **2015**, arXiv:1511.06939. [CrossRef]
42. Tan, Y.K.; Xu, X.; Liu, Y. Improved recurrent neural networks for session-based recommendations. In Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, Boston, MA, USA, 15 September 2016; pp. 17–22.
43. Hidasi, B.; Quadrana, M.; Karatzoglou, A.; Tikk, D. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, 15–19 September 2016; pp. 241–248.
44. Quadrana, M.; Karatzoglou, A.; Hidasi, B.; Cremonesi, P. Personalizing Session-based Recommendations with Hierarchical Recurrent Neural Networks. In Proceedings of the Eleventh ACM Conference on Recommender Systems, Como, Italy, 27–31 August 2017; pp. 130–137.
45. Wang, X.; Yu, L.; Ren, K.; Tao, G.; Zhang, W.; Yu, Y.; Wang, J. Dynamic attention deep model for article recommendation by learning human editors' demonstration. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; pp. 2051–2059.
46. Smirnova, E.; Vasile, F. Contextual Sequence Modeling for Recommendation with Recurrent Neural Networks. *arXiv* **2017**,arXiv:1706.07684. [CrossRef]
47. Wu, C.Y.; Ahmed, A.; Beutel, A.; Smola, A.J.; Jing, H. Recurrent recommender networks. In Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, Cambridge, UK, 6–10 February 2017; pp. 495–503.
48. 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. [CrossRef]
49. Wang, J.; Yu, L.; Zhang, W.; Gong, Y.; Xu, Y.; Wang, B.; Zhang, P.; Zhang, D. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval, Tokyo, Japan, 7–11 August 2017; pp. 515–524.
50. Mikolov, T.; Karafiát, M.; Burget, L.; Cernockỳ, J.; Khudanpur, S. Recurrent Neural Network Based Language Model. In Proceedings of the Eleventh Annual Conference of the International Speech Communication Association, Chiba, Japan, 26–30 September 2010; Volume 2, p. 3.
51. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
52. Tieleman, T.; Hinton, G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *Neur. Netw. Mach. Learn.* **2012**, *4*, 26–31.
53. Liu, D.Z.; Singh, G. A Recurrent Neural Network Based Recommendation System; Available online: http://cs224d.stanford.edu/reports/LiuSingh.pdf (accessed on 16 May 2018).
54. Wang, W.P.; Wang, J.H. Hybrid Recommendation Method Based on Tag and Collaborative Filtering. *Comput. Eng.* **2011**, *14*, 10.
55. Benesty, J.; Chen, J.; Huang, Y.; Cohen, I. Pearson correlation coefficient. In *Noise Reduction in Speech Processing*; Springer: Heidelberg/Berlin, Germany, 2009; pp. 1–4.
56. Cohen, J. *Statistical Power Analysis for the Behavioral Sciences*, 2nd ed.; Routledge: Abingdon, UK, 1988.