applied sciences

Article

Multi-Feature Extension via Semi-Autoencoder for Personalized Recommendation

Yishuai Geng [1,2], Yi Zhu [1,2,*], Yun Li [1,2], Xiaobing Sun [1,2] and Bin Li [1,2]

[1] School of Information Engineering, Yangzhou University, Yangzhou 225127, China
[2] Jiangsu Province Engineering Research Center of Knowledge Management and Intelligent Service, Yangzhou 225127, China
* Correspondence: zhuyi@yzu.edu.cn; Tel.: +86-152-2150-7475

Abstract: Over the past few years, personalized recommendation systems aim to address the problem of information overload to help users achieve useful information and make quick decisions. Recently, due to the benefits of effective representation learning and no labeled data requirements, autoencoder-based models have commonly been used in recommendation systems. Nonetheless, auxiliary information that can effectively enlarge the feature space is always scarce. Moreover, most existing methods ignore the hidden relations between extended features, which significantly affects the recommendation accuracy. To handle these problems, we propose a Multi-Feature extension method via a Semi-AutoEncoder for personalized recommendation (MFSAE). First, we extract auxiliary information from DBpedia as feature extensions of items. Second, we leverage the LSI model to learn hidden relations on top of item features and embed them into low-dimensional feature vectors. Finally, the resulting feature vectors, combined with the original rating matrix and side information, are fed into a semi-autoencoder for recommendation prediction. We ran comprehensive experiments on the MovieLens datasets. The results demonstrate the effectiveness of MFSAE compared to state-of-the-art methods.

Keywords: multi-feature extension; autoencoder; personalized recommendation; collaborative filtering; knowledge graph

check for updates

Citation: Geng, Y.; Zhu, Y.; Li, Y.; Sun, X.; Li, B. Multi-Feature Extension via Semi-Autoencoder for Personalized Recommendation. *Appl. Sci.* **2022**, *12*, 12408. https://doi.org/10.3390/app122312408

Received: 6 November 2022
Accepted: 1 December 2022
Published: 4 December 2022

1. Introduction

In the past decade, due to the explosive growth of the Internet and the resulting surge in data, people have easier access to a vast array of online products and multimedia content. While this growth allows users to have multiple choices, it also makes it difficult for users to choose their favorite among multiple candidates [1]. Recommendation System (RS) can effectively help users find valuable information and make decisions quickly to reduce information overload, which is recently utilized in e-commerce [2], e-learning [3], and intelligence education [4].

Among all the RS methods, Collaborative Filtering (CF) aims to achieve the feature representations of users and items and predicts new rating information with the feedback information. Traditional model-based CF methods commonly use machine learning models to extract hidden features between users and items and conduct appropriate recommendation models. Matrix Factorization (MF) models are the most used among these models [5]. However, the latent features learned by MF are generally inefficient, especially when the rating matrix is highly sparse. Along this line, there has already been some effort in devoting mining and exploiting hidden features of both implicit and explicit feature information to enhance recommendation accuracy.

Motivated by the application of deep neural network in various fields, including speech recognition and computer vision [6], deep neural network based recommendation methods have a profound impact on practical applications due to their stronger latent feature representation learning capability and flexible model structure. However, the most models still rely on considerable labeled data and the training time is prolonged [7]. AutoEncoder (AE), a type of artificial neural network, has recently observed a lot of work performed on top of them. AEs have also been actively used in CF models due to their superiority in non-labeled data requirements and quick convergence. While autoencoder-based methods have had some success in predicting the correct items for users—such as in Yelp, Netflix, or Amazon—significant challenges remain in the face of data sparsity, which directly affects RS performance. However, some methods have utilized auxiliary information for data sparsity [8,9]. However, two major issues still hinder the further development of these methods. First, the auxiliary information available to users and items for feature extension is usually limited. Second, most existing methods ignore the hidden relations between extended features, which significantly affects the recommendation performance.

To address the above issues, we propose a multi-feature extension method via a semi-autoencoder to improve personalized recommendation performance, called MFSAE. Considering that Knowledge Graph (KG) can produce reliable semantic retrieval data, we extract the auxiliary information of items through an open KG, which can be used as an additional attribute for the recommendation. To be more precise, we first extract the feature expansion information of items from DBpedia (an open KG based on Wikipedia). Secondly, we leverage the LSI model based on the Latent Semantic Analysis (LSA) model to learn hidden relations on top of item features and embed them into low-dimensional feature vectors. Finally, the resulting feature vectors, combined with the original rating matrix and side information, are fed into a semi-autoencoder for recommendation prediction. MFSAE enriches the content of auxiliary information and mines the similarity between expansion features to improve recommendation performance. Extensive experiments on two MovieLens datasets validate the effectiveness of MFSAE. Our contributions are outlined below in brief:

- We propose a novel semi-autoencoder based recommendation method, which introduces an open KG to extend the auxiliary feature information of items, and the captured information can be used flexibly and conveniently.
- We introduce LSI model to learn hidden relations on top of item features and embed them into a low-dimensional feature vector, which is combined with the original rating matrix and item attribute information to feed into a semi-autoencoder for rating prediction. It can effectively enhance recommendation accuracy.
- We conduct extensive experiments on the MovieLens datasets to demonstrate the superiority of MFSAE over competing for MF and deep neural network methods.

2. Related Work

2.1. Collaborative Filtering

CF is one of the most critical methods in RSs. CF-based methods find user preferences by mining historical interaction information between users and items and recommend potentially interesting items to users. In general, CF methods are divided into memory-based methods and model-based methods [10]. Memory-based CF aims to compute the similarity between users and items. This method is widely used due to its efficiency and ease of implementation. However, as the recommendation system size and data sparsity increase, the computation of similarity becomes more difficult. To address the above issues, various model-based CF methods have been proposed, which aim to build recommendation models based on extracting hidden features between users and items, such as latent semantic models [11], clustering models [12], MF models [13], etc.

Among various CF methods, MF is commonly used, which maps users and items to vectors of the same dimension to represent the latent features of the user or item. The classical MF methods include Singular Value Decomposition (SVD) [14] and Probabilistic Matrix Factorization (PMF) [15]. Compared with MF, Juan et al. introduced the feature domain and proposed the field-aware factorization machine model to further enhance the feature crossover ability [16]. Although the MF-based methods achieves excellent performance in

RS, it still suffers from the severe cold-start problem, that is, what suggestions should be given when a new user or item enters the system [17].

Recently, incorporating side information in CF is effective in dealing with the issues with cold starts and data sparsity [18]. For example, Symeonidis and Malakoudis proposed a multidimensional MF model using CF to predict course ratings based on external resource information and make them available for users to reference [19]. Rashed and Grabocka et al. proposed a Graph features-based Recommendation metohd (GraphRec) to extract generic graph-based attribute features for both users and items to improve recommendation performance [20]. Wang et al. proposed a method for collaborative knowledge-aware attentive networks that use the KGs to acquire side feature information and analyze potential semantic relationships [21]. To some extent, the aforementioned methods can ease the existing difficulty, however, how to learn more effective feature representations has influenced the further development of these methods.

2.2. *Deep Learning-Based Recommendation Methods*

At present, the existing methods based on deep learning have become effective resources for learning feature representation and provide various methods for improving RS performance [22]. Deep learning can disentangle the underlying explanatory factors behind the data and has made remarkable progress in learning effective feature representations for recommendation [23]. For example, Wei et al. offered two models to solve the incomplete and complete cold-start problem for new items, which combine CF methods to incorporate content features into cold-start item rating predictions [24].

With the rapid development of deep learning, various approaches, including Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and AE, have been extensively utilized in the recommended system [25]. For example, Zhou et al. proposed an integrated CNN-RNN framework to model and analyze patient-physician-generated data [26], but the model still relies on considerable labeled data and the training time is prolonged. In addition, Dau and Salim conducted a comprehensive and systematic investigation of RS based on deep learning, which revealed that AE is the most widely used deep learning framework until 2020 [27]. AEs are deep learning architectures for unsupervised learning that can efficiently recover data from a reduced encoded representation to the original input representation [28,29]. Due to the fast convergence rate and the no label requirement, Sedhain et al. proposed an Autoencoder-based Recommendation system (AutoRec) [30]. It encodes the user or item representation and leverages the generalization ability of the autoencoder to make recommendations. Along this line, several variants of autoencoder, such as denoising autoencoder [31], variational autoencoder [32], and stacked autoencoder [33] were proposed by later researchers and applied to recommendation systems.

Along this line, Zhang et al. proposed a Hybrid Collaborative Recommendation model based on Semi-Autoencoder (HCRSA) [8], which can generate personalized recommendations with auxiliary information and learned nonlinear features of users and items; the details are shown in Section 3.1. Moreover, Yang et al. proposed a Personalized Recommendation method via KG (PRKG), which obtains single side information through KG [34]. These works are close to ours, but with the following two main differences: (i) Considering less auxiliary information available, we utilize the KG to obtain more side feature information as additional attributes for the recommendation; (ii) since the semi-autoencoder models ignore the hidden relationships between mining features, we introduce the LSI model for hidden relationships mining among multiple features, which can be used as a favorable basis for the recommendation. We also transform the feature information into a low-dimensional feature vector, which helps to reduce the sparsity of the input model data.

3. Preliminaries

3.1. AutoEncoder and Semi-AutoEncoder

The AEs are deep learning architectures for unsupervised learning, which is mainly characterized by learning and representing input information as an objective. Through the reconstruction of the representation, the model seeks to minimize the discrepancy between the input and output data [28,29]. Recently, AEs have been frequently used for feature extraction, anomaly detection, and dimensionality reduction [35].

In light of the AE, Zhang et al. suggested to break the dimensionality restriction of the output, and the input must be equal [8], which can design the two variants of AE. The input layer of these variants can be longer/shorter than the output layer. The model shown in Figure 1b can be properly trained, but it is difficult to give a reasonable interpretation for these generated entries. The structure of semi-autoencoder is shown in Figure 1a; compared to traditional autoencoder, semi-autoencoder models make it easier to combine more features in the input layer, which can learn a better feature representation and mitigate data sparsity of the rating matrix.

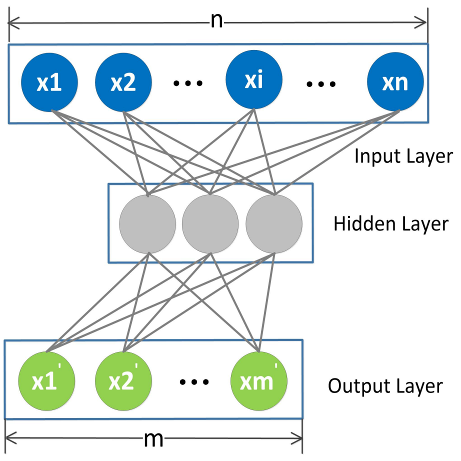
Similar to the autoencoder model, the structure of semi-autoencoder has three layers as well: input layer $x \in \Re^S$, hidden layer $\xi \in \Re^H$ and output layer $x' \in \Re^D$, where $H < D < S$. The network is formulated as (1) and (2):

$$\xi = f(W \cdot x + b) \tag{1}$$

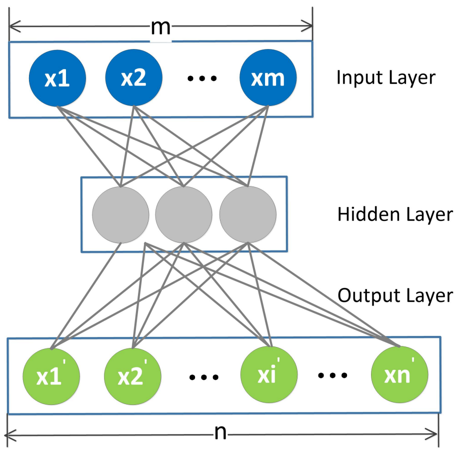
$$x' = g(W' \cdot \xi + b') \tag{2}$$

where $W \in \Re^{H \times S}$, $W' \in \Re^{D \times H}$ are the weighting matrices, and $b \in \Re^H$, $b' \in \Re^D$ are the bias vectors. Moreover, the nonlinear activation functions of the encode and decode layers are represented by f and g, respectively. Unlike the traditional autoencoder model, the subset $sub(x)$, which is taken from the input x and has the same dimension as the output x', is reconstructed by the semi-autoencoder from the input. The objective function can be formulated as (3):

$$\min_{W,W',b,b'} J = \left\| sub(x) - x' \right\|^2 \tag{3}$$



(**a**)



(**b**)

Figure 1. The two variant structures of AutoEncoder. (**a**) The input layer is longer than the output layer. (**b**) The input layer is shorter than the output layer.

3.2. Knowledge Graph

As a representation of information as a semantic graph, KG has attracted extensive research both in industry and academia [36]. The KG aims to describe the knowledge and establish the relationship between various things with a graph model. The KG consists of nodes and edges. Nodes can be entities, such as a person, a book, etc., or abstract concepts, such as artificial intelligence, KGs, etc. Edges can be attributes of entities, such as names and book titles, or relationships between entities, such as friends and spouses [37]. The users can easily obtain precise information from the KG and also obtain assistance in understanding the relationships between various items. They have led to significant possible solutions for numerous tasks, including question answering, personalized recommendation, and information retrieval.

DBpedia (https://www.dbpedia.org/ (accessed on 12 September 2022)) is a crowdsourced community that is an early semantic web project and a linked dataset extracted from Wikipedia. It provides a way to gather, organize, share, search, and use information while storing knowledge in a machine-readable format. After several years of continuous evolution, DBpedia now contains 3 billion resource description frameworks, of which 0.58 billion are extracted from the English Wikipedia and 2.46 billion are taken from other language editions, collected in DBpedia Commons, Wikidata, and other databases. Until 2017, DBpedia has become one of the largest representatives of linked open data, which is used in URI lookup services, query generators, etc. [34]. In this paper, we introduce DBpedia to obtain auxiliary feature information for items and analyze the hidden relationships between features, which can effectively mitigate data sparsity.

4. Methodology

4.1. Feature Representation Method Based on LSI Model

In traditional feature representation learning algorithms, One-Hot coding is mostly applied to process discrete feature data. However, when there are numerous types of data, the generated vectors are highly sparse, which is called "dimension explosion" [38]. Vector mapping is an effective approach to address this problem, which has the capability of learning a low-dimensional continuous vector representation of items, and the mapping results for items with similar actual meanings in the vector space have similar distances. Vector mapping methods reduce noise and redundant information and preserve structural information between data. Common vector mapping methods include MF, random walk algorithms, and deep neural network models. Alternatively, vector mapping methods can use dense low-dimensional numerical feature vectors to represent high-dimensional sparse features [39]. This method also supports subsequent network processing tasks, including node classification [40], node clustering [41], and network visualization [42].

LSA belongs to the topic model, which is constructed by SVD. It relies on a statistical analysis of a large set of texts to extract the semantics of words hidden in the context from the analyzed data. Figure 2 depicts the basic structure of the LSA model. First, the text dataset is analyzed and then the vocabulary text matrix corresponding to the text dataset is established. The resulting lexical text matrix is then decomposed into singular values with SVD. The process aims to map the vocabulary and text into the same vector space. The decomposed matrix is then dimensionally reduced and the processed low-dimensional data is used to construct the final latent semantic space. Compared to the traditional vector spaces, the semantic space constructed by the LSA principle has a smaller dimension, which effectively addresses the negative effects of synonyms and enhances the semantic relationship between words. LSA is a classic bag-of-words model and neglects word order. However, the main task of LSA in our method is to implement a vectorized mapping of the collected item features; the involved item features do not exist sequentially and do not affect the accuracy of the recommendation results.

In our proposed method, we introduce the LSI model from Gensim (http://pypi.python.org/pypi/gensim, accessed on 20 September 2022), which is a python library for automatically extracting semantic topics from documents to complete the vectorized mapping of features. LSI model first builds a dictionary according to the vocabulary data set and generates the corresponding bag of words model. It calculates the weight of words based on Term Frequency-Inverse Document Frequency (TF-IDF) [43]. TF-IDF refers to the larger the weight, the greater the importance of the word to the document. Finally, it reduces the dimensionality through the LSA algorithm to filter out word noise.

The processed results are used as auxiliary information for the items, which effectively solves the sparsity problem in the personalized recommendation.

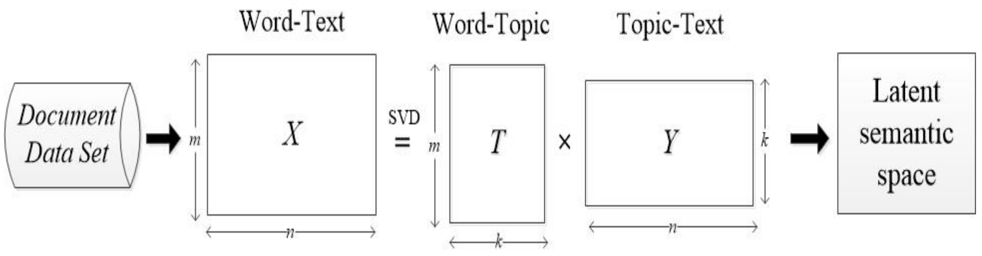


Figure 2. The whole framework of the LSA model.

4.2. MFSAE for Personalized Recommendation

The overall framework and detailed description of MFSAE are illustrated in Figure 3. Considering that the rating matrix $R \in \Re^{n\times m}$ in real-world scenarios is usually extremely sparse, we introduce the own attributes information of items such as the genres and release date in the movie datasets, which is denoted as $\Re^{k_1}$, and $A^i \in \Re^{k_1}$ denote the attributes information for item i, $A^I \in \Re^{m\times k_1}$ is denoted as the attributes information vectors of all m items. In addition, considering the limitation of extensible feature information, the DBpedia is introduced to obtain all movie languages as auxiliary information for the model. The LSI model is introduced to embed language feature information into a low dimensional vector $R \in \Re^{k_2}$, where k_2 represents the output dimension of LSI model data in the model, $L^i \in \Re^{k_2}$ denote the movie language vector of item i, and we denote $L^I \in \Re^{m\times k_2}$ as the language vectors of all m items. MFSAE is first introduced to incorporate the rating vector R^i, the attributes vector A^i and the extended language features L^i. The model's input can be formulated as $con\left(R^i, A^i, L^i\right)$:

$$con\left(R^i, A^i, L^i\right) \overset{\text{def}}{=} connection\ of\ R^i\ \ A^i\ and\ L^i \tag{4}$$

The $con\left(R^i, A^i, L^i\right) \in \Re^{n\times(m+k_1+k_2)}$ refers to the connection of R^I, A^I and L^I, where $R^I \in \Re^{n\times m}$ represents the rating vectors, $A^I \in \Re^{n\times k_1}$ represents the attribute information vectors of all items and $L^I \in \Re^{n\times k_2}$ represents the language vectors of all items. Then, the model learns the compressed reconstructed output from the input $con\left(R^I, A^I, L^I\right)$; the encoding stage can be formulated as (5):

$$\xi = f\left(con\left(R^I, A^I, L^I\right) \cdot W + b\right) \tag{5}$$

where $W \in \Re^{(m+k_1+k_2)\times h}$ is the weight matrix, $b \in \Re^{n\times h}$ is bias vector and f is the *sigmoid* function for nonlinear activation. The decoding stage can be formulated as (6):

$$\begin{aligned} R' &= g\left(\xi \cdot W' + b'\right) \\ &= g\left(f\left(con\left(R^I, A^I, L^I\right) \cdot W + b\right) \cdot W' + b'\right) \end{aligned} \tag{6}$$

where $W' \in \Re^{h\times m}$ and $b' \in \Re^{n\times m}$ represent the decoding layer's weight matrix and bias vector, respectively, and g stands for the *identity* function for nonlinear activation. Notably, the Stochastic Gradient Descent (SGD) method is used for model optimization in the semi-autoencoder. Furthermore, the weight matrix W and W' of the ℓ_2 norm regularization are added to the objective function for avoiding overfitting, which can be formulated as (7):

$$J = \|W\|_2^2 + \left\|W'\right\|_2^2 \tag{7}$$

Similar to the autoencoder, we compute the loss function of the weight matrix of the encoding and decoding stages, which can enhance the auxiliary information for the reconstruction of R^I and improve prediction accuracy. Thus, the objective function can be formulated as (8):

$$J_{item} = \left\| (R' - R^I) \right\|^2 + \alpha \cdot J \tag{8}$$

where α is the trade-off parameter that controls the balance of the regularization terms. The best recommendation result is obtained by minimizing the error between input R and output R'. When the model converges, the matrix R', the output layer of the semi-autoencoder, is used for prediction. The details of MFSAE are summarized, as follows Algorithm 1.

Algorithm 1 Multi-feature extension via semi-autoencoder for personalized recommendation

```
Require: The rating matrix $R \in \Re^{m \times n}$, trade-off parameter $\alpha$, the dimension of hidden
    layers $h$.
Ensure: The predicated rating matrix $R'$.
1:     Get the original attribute information vector $A^i$ for each item;
2:     Apply DBpedia to obtain the language of the movie, and use LSI model to obtain
    low-dimensional feature vectors $L^i$ for each item;
3:     Input the concatenation vectors $con(R^I, A^I, L^I)$ of the item's rating vector $R^I$ and
    extended information $A^I$, $L^I$ into the semi-autoencoder;
4:     Initialize $W$, $b$ and $W'$, $b'$ randomly, respectively;
5:     Minimize $\|R^I - R'\|_2^2$ with SGD method until convergence;
6: return The predicated rating matrix $R'$;
```

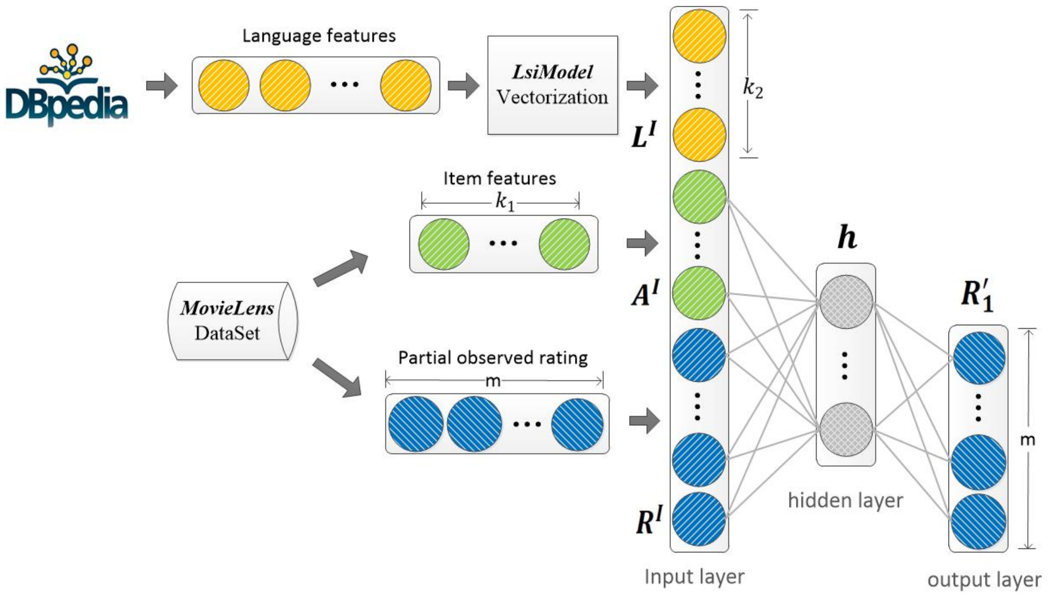


Figure 3. The whole framework of the MFSAE.

5. Experiments

5.1. Datasets

In this paper, we conduct experiments using three movie datasets, MovieTweetings (https://github.com/sidooms/MovieTweetings (accessed on 27 September 2022)) (10 K) and MovieLens (http://files.grouplens.org/datasets/movielens (accessed on 29 September 2022)) (100 K and 1 M). Each dataset is highly sparse and contains information about users, movies, movie features, and ratings. Table 1 displays details about the three datasets.

MovieTweetings: This dataset is a collection of movie data from Twitter that has been widely used over the past decade [44]. Each rating is rated on a scale of 1 to 10, the higher the rating, the more the user prefers the item. For our experiments, we select a 10 K snapshot of MovieTweetings and retained only users who rated at least 10 movies.

MovieLens: It is a well-known and frequently used recommendation dataset. Each user has rated at least 20 movies, with a rating being an integer between 1 and 5. Moreover, the 1 M dataset is sparser than the 100 K dataset.

Table 1. Details of three datasets.

Dataset	#Users	#Items	Ratings	Sparsity%	Item Features
MovieTweetings 10 K	123	3096	2233	99.41%	Release date; Genres
MovieLens 100 K	943	1682	100,000	93.70%	Release date; Genres
MovieLens 1 M	6040	3900	1,000,209	95.74%	Release date; Genres

5.2. Compared Methods

To evaluate the performance of MFSAE, we compare it with MF, deep neural networks, and graph methods. Details of these baseline recommendation methods are listed below:

- SVD++ [14]. SVD++ combines latent factors and neighborhood models into a single recommendation model by leveraging both explicit and implicit user feedback.
- I-AutoRec [30]. Item-based AutoRec (I-AutoRec) model learns effective feature representations of items for recommendation using an autoencoder.
- HCRSA [8]. HCRSA breaks the restriction that the dimensions of the autoencoder's input and output layers be equal and introduces auxiliary information for representation learning.
- GraphRec [20]. GraphRec co-embeds users and items into a shared latent space and utilizes the Laplacian of the user–item interaction graph to extract generic graph-based attribute features. This model only needs the rating matrix and all attribute information is extracted from the structure of the graph.
- PRKG [34]. PRKG extracts side information from DBpedia and encodes it into a low-dimensional representation with an autoencoder and introduces a semi-autoencoder to fuse the side information for the recommendation.

5.3. Implementation Details

In all experiments, we set the regularization parameter $\alpha = 0.05$ and the $learning\ rate = 0.001$ for all datasets. For all AE-based methods, we choose *Identity* and *Sigmoid* as the nonlinear activation functions of the encode and decode layers, respectively. The optimization method is the *Adam* method of SGD, $epoch = 65$, $batch\ size = 300$. Notably, in this paper, all results are acquired by averaging five repetitions of the experiment. Moreover, we also clearly list the other implementation details of all methods, as follows:

- SVD++: This method is implemented through the Personalized Recommendation Algorithms (PREA) toolkit [45].
- I-AutoRec: We choose the item-based autoencoder recommendation model and set the $hidden\ neuron = 500$.
- HCRSA: We performed the source code of HCRSA (https://github.com/cheungdaven/semi-ae-recsys (accessed on 2 October 2022)). The side information includes the release date and genres of movies.
- GraphRec: We performed the source code of GraphRec (https://github.com/ahmedrashed-ml/GraphRec (accessed on 4 October 2022)). We choose GraphRec with extended side feature for comparison.
- PRKG: According to the source paper, we obtained the language of all movies via DBpedia. We reproduce the model in the experiment with all parameter settings consistent with the source paper.
- MFSAE: For fairness, we introduce the same item extension feature information as HCRSA, GraphRec and PRKG.

5.4. Evaluation Metrics

We use Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) in the experiments to evaluate the performance of all methods in this paper. These formulas of metrics are defined as (9) and (10). Significantly, smaller MAE and RMSE values indicate higher performance.

$$\mathrm{MAE} = \frac{\sum\limits_{r_{u,i} \in \mathrm{TestSet}} \left| r_{u,i} - r'_{u,i} \right|}{|\mathrm{TestSet}|} \tag{9}$$

$$\mathrm{RMSE} = \sqrt{\frac{\sum\limits_{r_{u,i} \in \mathrm{TestSet}} \left(r_{u,i} - r'_{u,i} \right)^2}{|\mathrm{TestSet}|}} \tag{10}$$

where $r_{u,i}$ represent the original rating matrix and $r'_{u,i}$ represent the predication matrix from the semi-autoencoder.

5.5. Experimental Results

For each dataset, a sample of 50%, 60%, 70%, 80%, and 90% is utilized for training, while the remainder is used for testing. The experimental results of MAE and RMSE on two MovieLens datasets are presented in Tables 2–4 and Figures 4 and 5. With different numbers of training samples, the values in bold in the Tables indicate the best among all methods and the following observations can be drawn from the experimental results:

- The histograms in Figures 4 and 5 reflect that the RMSE and MAE values of all methods decrease periodically as training data increases, indicating that the performance of all methods improves as training data increases.
- Traditional models based on MF (such as SVD++) perform poorly due to data sparsity. Compared with this method, deep neural network methods (such as I-AutoRec, HCRSA, PRKG, and MFSAE) can achieve better performance, showing deep neural networks' powerful ability to learn personalized feature representations.
- In the autoencoder-based methods, the performance of MFSAE is better than the I-AutoRec, HCRSA, and PRKG models. The HCRSA method solves the data sparsity problem on top of I-AutoRec by introducing additional attribute information and the PRKG adds side information by introducing additional features via a KG. However, MFSAE introduces external extended feature information via DBpedia and uses LSI model to learn hidden relations. This shows the superiority of effective side information in improving personalized recommendation performance and the advantage of feature representation learning.
- Moreover, GraphRec achieves excellent recommendation performance on the MovieLens dataset due to the introduction of item and user graph features. GraphRec performs better than MFSAE when the training rate is 0.5 or 0.6, which means that the graph feature representation is more efficient when the data is sparse. Moreover, MFSAE outperforms GraphRec when the training data is large, reflecting the strong advantage of deep learning feature representations. Notably, the model structure of MFSAE is simpler.
- Typically, data sparsity has been an essential factor for recommendation performance. As shown in Table 1, MovieTweetings 10 K, with tiny rating data, is the sparsest and MovieLens 1 M is more sparse than MovieLens 100 K. From the results, we can conclude that the performance of RS methods, especially deep learning models, will be considerably affected by the amount of training data.
- Overall, MFSAE performs better than most methods. Due to its simple structure for learning the latent features of the side information introduced by DBpedia through LSI model, MFSAE achieves stable and better performance.

Table 2. The performance of MAE and RMSE on MovieTweetings 10 K dataset.

Metrics	**Methods**	**Proportion of Training Data**				
		50%	**60%**	**70%**	**80%**	**90%**
MAE	SVD++	1.472	1.392	1.231	1.227	1.140
	I-AutoRec	1.496	1.448	1.323	1.366	1.298
	HCRSA	1.230	1.174	1.123	1.086	1.015
	PRKG	1.018	0.976	0.931	0.891	0.886
	MFSAE	**0.892**	**0.887**	**0.852**	**0.836**	**0.817**
RMSE	SVD++	1.768	1.624	1.529	1.498	1.452
	I-AutoRec	1.731	1.668	1.607	1.584	1.542
	HCRSA	1.471	1.349	1.288	1.219	1.177
	PRKG	1.217	1.126	1.072	1.034	1.012
	MFSAE	**1.084**	**1.043**	**1.007**	**0.989**	**0.952**

Tip: The bolder ones mean better.

Table 3. The performance of MAE and RMSE on MovieLens 100 K dataset.

Metrics	**Methods**	**Proportion of Training Data**				
		50%	**60%**	**70%**	**80%**	**90%**
MAE	SVD++	0.752	0.747	0.741	0.726	0.722
	I-AutoRec	0.729	0.723	0.714	0.704	0.698
	HCRSA	0.727	0.724	0.713	0.711	0.703
	GraphRec	**0.721**	**0.714**	0.709	0.703	0.701
	PRKG	0.729	0.723	0.714	0.704	0.698
	MFSAE	0.723	0.715	**0.706**	**0.700**	**0.692**
RMSE	SVD++	0.979	0.965	0.949	0.932	0.924
	I-AutoRec	1.021	1.003	0.979	0.964	0.953
	HCRSA	0.927	0.921	0.907	0.891	0.897
	GraphRec	**0.916**	**0.905**	0.899	0.893	0.887
	PRKG	0.928	0.917	0.913	0.899	0.895
	MFSAE	0.917	0.907	**0.899**	**0.890**	**0.882**

Tip: The bolder ones mean better.

Table 4. The performance of MAE and RMSE on MovieLens 1 M dataset.

Metrics	**Methods**	**Proportion of Training Data**				
		50%	**60%**	**70%**	**80%**	**90%**
MAE	SVD++	0.683	0.678	0.674	0.668	0.666
	I-AutoRec	0.729	0.723	0.714	0.704	0.698
	HCRSA	0.692	0.687	0.681	0.675	0.668
	GraphRec	**0.683**	0.679	0.673	0.668	0.664
	PRKG	0.705	0.696	0.690	0.684	0.679
	MFSAE	0.684	**0.677**	**0.672**	**0.667**	**0.662**
RMSE	SVD++	0.879	0.866	0.859	0.851	0.848
	I-AutoRec	0.914	0.905	0.896	0.888	0.881
	HCRSA	0.892	0.885	0.879	0.871	0.863
	GraphRec	**0.870**	0.866	0.857	0.851	0.847
	PRKG	0.898	0.888	0.881	0.874	0.868
	MFSAE	0.872	**0.863**	**0.857**	**0.851**	**0.847**

Tip: The bolder ones mean better.

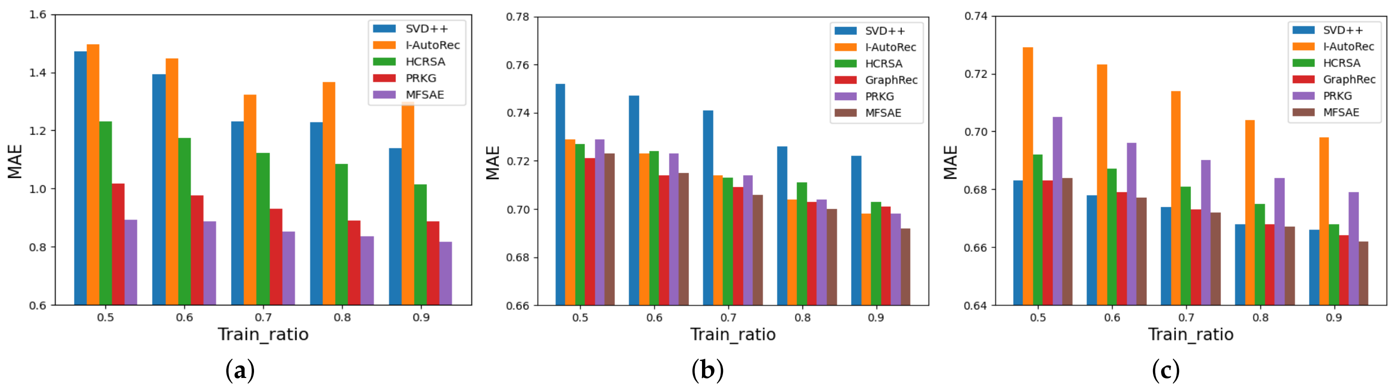


Figure 4. The performance of MAE for all methods on the three datasets. (**a**) In MovieTweetings 10 K. (**b**) In MovieLens 100 K. (**c**) In MovieLens 1 M.

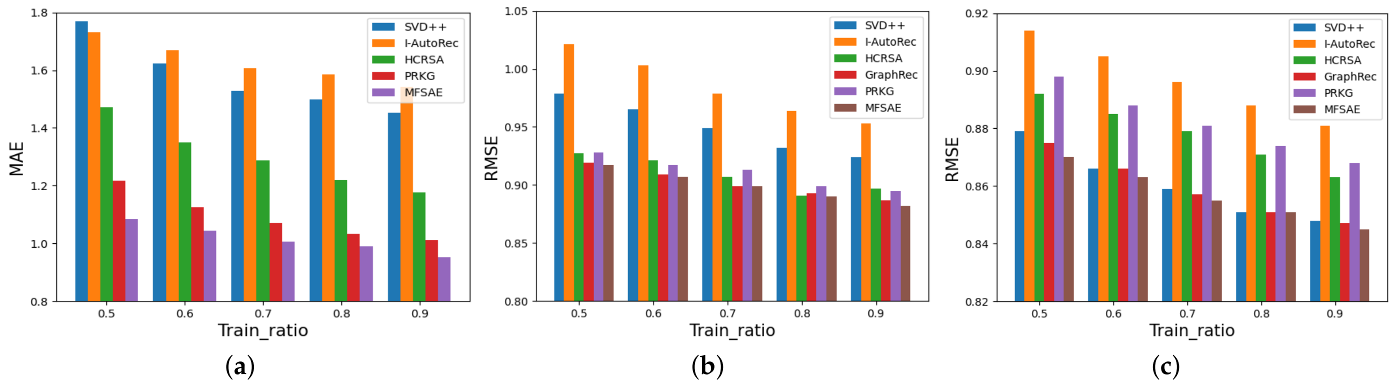


Figure 5. The performance of RMSE for all methods on the three datasets. (**a**) In MovieTweetings 10 K. (**b**) In MovieLens 100 K. (**c**) In MovieLens 1 M.

5.6. Parameter Sensitivity

5.6.1. The Number of Hidden Layer Neurons and Epochs

We explore the performance of MFSAE under different numbers of hidden layer neurons and epochs. Before that, we choose the number of hidden layer neurons to be sampled in $\{100, 200, 300, 400, 500, 600, 700, 800\}$ and the number of epochs to be sampled in $\{20, 30, 40, 50, 60, 70, 80, 90, 100\}$, all the experiments are conducted under the $train\ ratio = 0.8$. When one parameter in the experiment is altered, the others remain constant.

In the main experiments, with $train\ ratio = 0.8$ and $hidden\ neuron = 200$, we find that the model performs close to optimal in both RMSE and MAE. Therefore, for fairness, we set $train\ ratio = 0.8$ and $hidden\ neuron = 200$ to investigate the impact of the number of epochs on the three datasets, and Figure 6 shows the experimental results. When MFSAE achieves the best performance, the epoch values are set to 60, 70 and 60 on the three datasets, respectively. Moreover, by comparing experiments on the three datasets, we discover that the more sparse the datasets, the faster the model converges.

To compare the above experiments, we select the results of the above experiments to investigate the impact of the number of hidden neurons. The experimental results in Figure 7 reveal that when the number of hidden layer neurons is adjusted to 500, 400 and 200, respectively, MFSAE performs best.

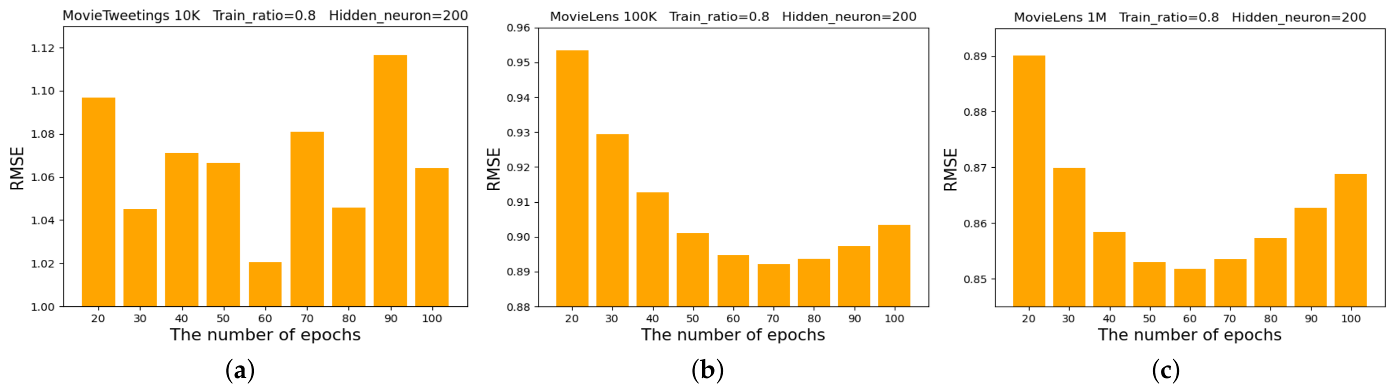


Figure 6. The parameter impact of the different numbers of epochs. (**a**) In MovieTweetings 10 K. (**b**) In MovieLens 100 K. (**c**) In MovieLens 1 M.

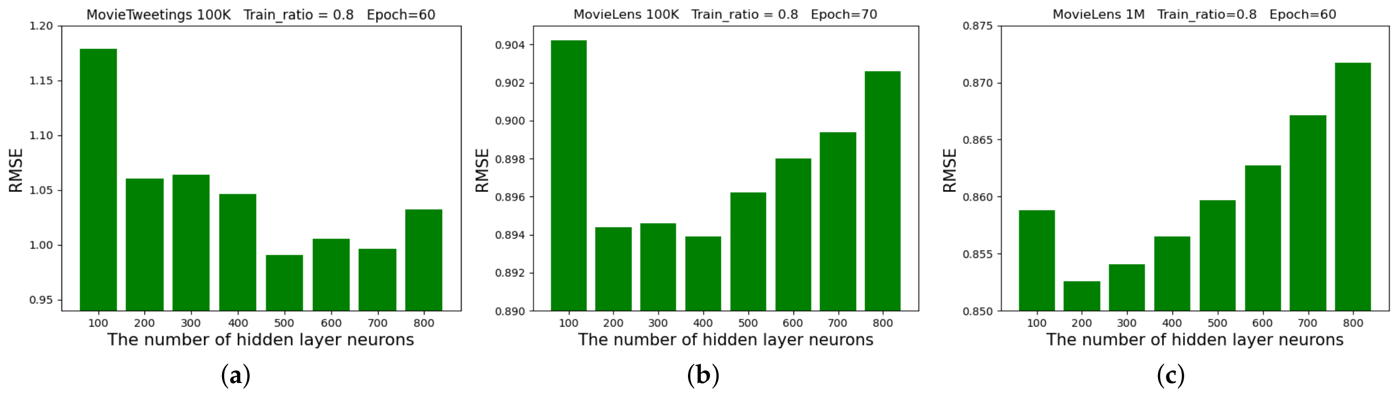


Figure 7. The parameter impact of the different numbers of hidden layer neurons. (**a**) In MovieTweetings 10 K. (**b**) In MovieLens 100 K. (**c**) In MovieLens 1 M.

5.6.2. The Setting of K_2

As shown in the above experiments, we use the LSI model to embed language feature information into the low dimensional vector $R \in \Re^{k_2}$, where k_2 represents the output dimension of the LSI model data in the model. In this part, we explore the impact of the k_2 setting on the model's overall performance. Before that, we determine the dimension of language vector k_2 to be sampled in $\{10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70\}$. As shown in Figure 8, we discovered that MFSAE has the best performance when $k_2 = 45$ on the MovieLens dataset and when $k_2 = 30$ on MovieTweeting 10 K.

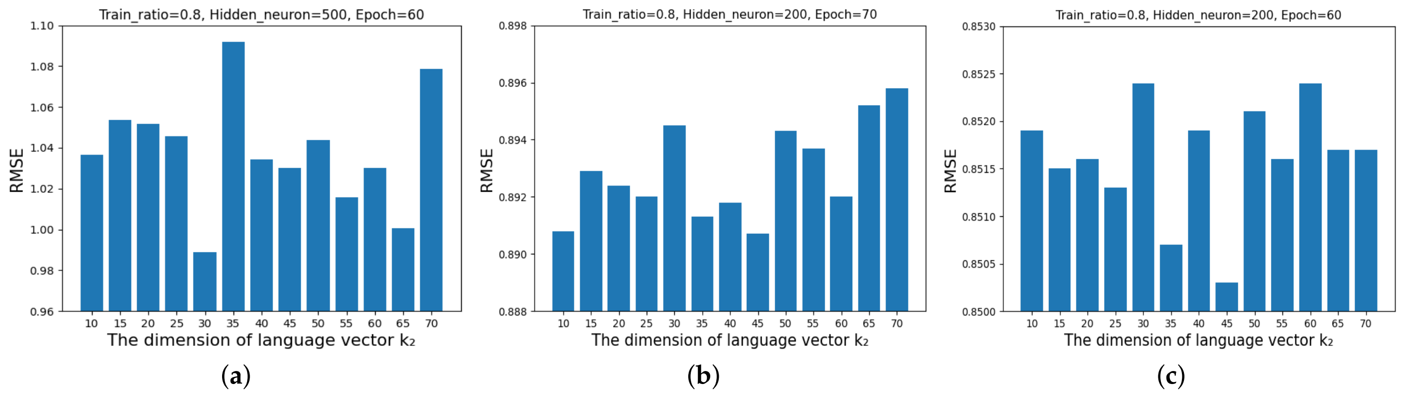


Figure 8. The parameter impact of the different dimensions of language vector k_2. (**a**) In MovieTweetings 10 K. (**b**) In MovieLens 100 K. (**c**) In MovieLens 1 M.

6. Conclusions

In this paper, we propose a multi-feature extension method via a semi-autoencoder to improve recommendation performance, called MFSAE. Because of the advantages of semi-autoencoders in information fusion, MFSAE may utilize auxiliary information derived from DBpedia to obtain effective feature representations of items for reducing data sparsity. In addition, we introduce LSI model to embed the language extension feature information into a low-dimensional vector, which obtains the hidden relations between the items effectively and improves the recommendation accuracy. Experiments on the Movielens dataset validate that MFSAE outperforms the state-of-the-art models.

In future works, first, we will employ graph networks to discover additional critical feature information between users and items, and strengthen the combination of multi-source features by using attention mechanisms. Second, we will also introduce additional deep learning models to learn the relationships of implicit features.

Author Contributions: Conceptualization, Y.G. and Y.Z.; methodology, Y.G. and Y.Z.; software, Y.G.; validation, Y.L. and X.S.; formal analysis, Y.G. and B.L.; investigation, Y.Z.; resources, Y.G.; data curation, Y.Z.; writing—original draft preparation, Y.G. and Y.Z.; writing—review and editing, Y.L., X.S. and B.L.; visualization, Y.L.; supervision, X.S.; project administration, B.L.; funding acquisition, Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research is partially supported by the Yangzhou University Interdisciplinary Research Foundation (yzuxk202008, yzuxk202015), National Natural Science Foundation of China under grants (61906060), and Open Project Program of Joint International Research Laboratory of Agriculture and Agri-Product Safety (JILAR-KF202104).

Data Availability Statement: The experimental data in the paper can be obtained from the corresponding author.

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

References

1. Ma, C.; Kang, P.; Wu, B.; Wang, Q.; Liu, X. Gated attentive-autoencoder for content-aware recommendation. In Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, Melbourne, Australia, 11–15 February 2019; pp. 519–527.
2. Khoali, M.; Laaziz, Y.; Tali, A.; Salaudeen, H. A Survey of One Class E-Commerce Recommendation System Techniques. *Electronics* **2022**, *11*, 878. [CrossRef]
3. Rahayu, N.W.; Ferdiana, R.; Kusumawardani, S.S. A systematic review of ontology use in E-Learning recommender system. *Comput. Educ. Artif. Intell.* **2022**, *3*, 100047. [CrossRef]
4. García-Peñalvo, F.J.; Corell, A.; Abella-García, V.; Grande-de Prado, M. Recommendations for mandatory online assessment in higher education during the COVID-19 pandemic. In *Radical Solutions for Education in a Crisis Context*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 85–98.
5. Zhang, L.; Luo, T.; Zhang, F.; Wu, Y. A recommendation model based on deep neural network. *IEEE Access* **2018**, *6*, 9454–9463. [CrossRef]
6. Alam, M.; Samad, M.; Vidyaratne, L.; Glandon, A.; Iftekharuddin, K. Survey on Deep Neural Networks in Speech and Vision Systems. *Neurocomputing* **2020**, *417*, 302–321. [CrossRef] [PubMed]
7. Geng, Y.; Xiao, X.; Sun, X.; Zhu, Y. Representation learning: Recommendation with knowledge graph via triple-autoencoder. *Front. Genet.* **2022**, *13*, 891265. [CrossRef]
8. Shuai, Z.; Yao, L.; Xu, X.; Wang, S.; Zhu, L. Hybrid Collaborative Recommendation via Semi-AutoEncoder. In *Neural Information Processing. ICONIP 2017*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2017; pp. 185–193.
9. Wu, Y.; Macdonald, C.; Ounis, I. A hybrid conditional variational autoencoder model for personalised top-n recommendation. In Proceedings of the 2020 ACM SIGIR on International Conference on Theory of Information Retrieval, New York, NY, USA, 14–17 September 2020; pp. 89–96.
10. Zhong, S.T.; Huang, L.; Wang, C.D.; Lai, J.H.; Philip, S.Y. An autoencoder framework with attention mechanism for cross-domain recommendation. *IEEE Trans. Cybern.* **2020**, *52*, 5229–5241. [CrossRef]
11. Wang, X.; Wang, R.; Shi, C.; Song, G.; Li, Q. Multi-component graph convolutional collaborative filtering. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 6267–6274.
12. Cui, Z.; Xu, X.; Fei, X.; Cai, X.; Cao, Y.; Zhang, W.; Chen, J. Personalized recommendation system based on collaborative filtering for IoT scenarios. *IEEE Trans. Serv. Comput.* **2020**, *13*, 685–695. [CrossRef]
13. Pujahari, A.; Sisodia, D.S. Pair-wise preference relation based probabilistic matrix factorization for collaborative filtering in recommender system. *Knowl.-Based Syst.* **2020**, *196*, 105798. [CrossRef]

14. 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.
15. Mnih, A.; Salakhutdinov, R.R. Probabilistic matrix factorization. In Proceedings of the 21th Annual Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 3–6 December 2007; pp. 1257–1264.
16. Juan, Y.; Zhuang, Y.; Chin, W.S.; Lin, C.J. Field-aware factorization machines for CTR prediction. In Proceedings of the 10th ACM Conference on Recommender Systems, New York, NY, USA, 15–19 September 2016; pp. 43–50.
17. Lu, J.; Wu, D.; Mao, M.; Wang, W.; Zhang, G. Recommender system application developments: A survey. *Decis. Support Syst.* **2015**, *74*, 12–32. [CrossRef]
18. Truong, Q.T.; Salah, A.; Lauw, H.W. Bilateral variational autoencoder for collaborative filtering. In Proceedings of the 14th ACM International Conference on Web Search and Data Mining, Virtual Event, 8–12 March 2021; pp. 292–300.
19. Symeonidis, P.; Malakoudis, D. Multi-modal matrix factorization with side information for recommending massive open online courses. *Expert Syst. Appl.* **2019**, *118*, 261–271. [CrossRef]
20. Rashed, A.; Grabocka, J.; Schmidt-Thieme, L. Attribute-aware non-linear co-embeddings of graph features. In Proceedings of the 13th ACM Conference on Recommender Systems, Copenhagen, Denmark, 16–20 September 2019; pp. 314–321.
21. Wang, Z.; Lin, G.; Tan, H.; Chen, Q.; Liu, X. CKAN: Collaborative knowledge-aware attentive network for recommender systems. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, New York, NY, USA, 25–30 July 2020; pp. 219–228.
22. Naumov, M.; Mudigere, D.; Shi, H.J.M.; Huang, J.; Sundaraman, N.; Park, J.; Wang, X.; Gupta, U.; Wu, C.J.; Azzolini, A.G.; et al. Deep learning recommendation model for personalization and recommendation systems. *arXiv* **2019**, arXiv:1906.00091.
23. Liu, L.; Ouyang, W.; Wang, X.; Fieguth, P.; Chen, J.; Liu, X.; Pietikäinen, M. Deep learning for generic object detection: A survey. *Int. J. Comput. Vis.* **2020**, *128*, 261–318. [CrossRef]
24. Wei, J.; He, J.; Chen, K.; Zhou, Y.; Tang, Z. Collaborative filtering and deep learning based recommendation system for cold start items. *Expert Syst. Appl.* **2017**, *69*, 29–39. [CrossRef]
25. Zhu, Y.; Wu, X.; Qiang, J.; Yuan, Y.; Li, Y. Representation learning with collaborative autoencoder for personalized recommendation. *Expert Syst. Appl.* **2021**, *186*, 115825. [CrossRef]
26. Zhou, X.; Li, Y.; Liang, W. CNN-RNN based intelligent recommendation for online medical pre-diagnosis support. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **2020**, *18*, 912–921. [CrossRef]
27. Da'u, A.; Salim, N. Recommendation system based on deep learning methods: A systematic review and new directions. *Artif. Intell. Rev.* **2020**, *53*, 2709–2748. [CrossRef]
28. Cao, S.; Yang, N.; Liu, Z. Online news recommender based on stacked auto-encoder. In Proceedings of the 2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS), Wuhan, China, 24–26 May 2017; pp. 721–726.
29. Nurmaini, S.; Darmawahyuni, A.; Sakti Mukti, A.N.; Rachmatullah, M.N.; Firdaus, F.; Tutuko, B. Deep learning-based stacked denoising and autoencoder for ECG heartbeat classification. *Electronics* **2020**, *9*, 135. [CrossRef]
30. 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, New York, NY, USA, 18–22 May 2015; pp. 111–112.
31. Pan, Y.; He, F.; Yu, H. A correlative denoising autoencoder to model social influence for top-N recommender system. *Front. Comput. Sci.* **2020**, *14*, 143301. [CrossRef]
32. Shenbin, I.; Alekseev, A.; Tutubalina, E.; Malykh, V.; Nikolenko, S.I. Recvae: A new variational autoencoder for top-n recommendations with implicit feedback. In Proceedings of the 13th International Conference on Web Search and Data Mining, Houston, TX, USA, 3–7 February 2020; pp. 528–536.
33. Yu, M.; Quan, T.; Peng, Q.; Yu, X.; Liu, L. A model-based collaborate filtering algorithm based on stacked AutoEncoder. *Neural Comput. Appl.* **2022**, *34*, 2503–2511. [CrossRef]
34. Yang, Y.; Zhu, Y.; Li, Y. Personalized recommendation with knowledge graph via dual-autoencoder. *Appl. Intell.* **2022**, *52*, 6196–6207. [CrossRef]
35. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
36. Zou, X. A survey on application of knowledge graph. *J. Phys. Conf. Ser.* **2020**, *1487*, 012016. [CrossRef]
37. Guan, L.; Zhang, J.; Geng, C. Diagnosis of Fruit Tree Diseases and Pests Based on Agricultural Knowledge Graph. *J. Phys. Conf. Ser.* **2021**, *1865*, 042052. [CrossRef]
38. Sunil Datt, M. The information explosion: Trends in technology 2011 review. *J. Gov. Financ. Manag.* **2011**, *60*, 46.
39. Tancik, M.; Srinivasan, P.; Mildenhall, B.; Fridovich-Keil, S.; Raghavan, N.; Singhal, U.; Ramamoorthi, R.; Barron, J.; Ng, R. Fourier features let networks learn high frequency functions in low dimensional domains. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 7537–7547.
40. Ullah, F.; Naeem, M.R.; Naeem, H.; Cheng, X.; Alazab, M. CroLSSim: Cross-language software similarity detector using hybrid approach of LSA-based AST-MDrep features and CNN-LSTM model. *Int. J. Intell. Syst.* **2022**, *37*, 5768–5795. [CrossRef]
41. Senthil, G.; Raaza, A.; Kumar, N. Internet of Things Energy Efficient Cluster-Based Routing Using Hybrid Particle Swarm Optimization for Wireless Sensor Network. *Wirel. Pers. Commun.* **2022**, *122*, 2603–2619. [CrossRef]

42. Kalepalli, Y.; Tasneem, S.; Teja, P.D.P.; Manne, S. Effective comparison of lda with lsa for topic modelling. In Proceedings of the 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 13–15 May 2020; pp. 1245–1250.
43. Mimura, M.; Ohminami, T. Using LSI to detect unknown malicious VBA macros. *J. Inf. Process.* **2020**, *28*, 493–501. [CrossRef]
44. Dooms, S.; De Pessemier, T.; Martens, L. Movietweetings: A movie rating dataset collected from twitter. In Proceedings of the Workshop on Crowdsourcing and Human Computation for Recommender Systems, CrowdRec at RecSys, Hongkong, China, 12 October 2013; p. 43.
45. Lee, J.; Sun, M.; Lebanon, G.; Sonnenburg, S. PREA: Personalized recommendation algorithms toolkit. *J. Mach. Learn. Res.* **2014**, *13*, 2699–2703.