mathematics

MDPI

Article

DFM-GCN: A Multi-Task Learning Recommendation Based on a Deep Graph Neural Network

Yan Xiao [1,2], Congdong Li [1,3,4,*] and Vincenzo Liu [1,*]

[1] School of Business, Macau University of Science and Technology, Macao 999078, China; yxiao80@163.com
[2] College of Mechanical Engineering, Chongqing University of Technology, Chongqing 400054, China
[3] Management School, Jinan University, Guangzhou 510632, China
[4] Institute of Physical Internet, Jinan University (Zhuhai Campus), Zhuhai 519070, China
* Correspondence: licd@jnu.edu.cn (C.L.); ydliu@must.edu.mo (V.L.)

Abstract: Among the inherent problems in recommendation systems are data sparseness and cold starts; the solutions to which lie in the introduction of knowledge graphs to improve the performance of the recommendation systems. The results in previous research, however, suffer from problems such as data compression, information damage, and insufficient learning. Therefore, a DeepFM Graph Convolutional Network (DFM-GCN) model was proposed to alleviate the above issues. The prediction of the click-through rate (CTR) is critical in recommendation systems where the task is to estimate the probability that a user will click on a recommended item. In many recommendation systems, the goal is to maximize the number of clicks so the items returned to a user can be ranked by an estimated CTR. The DFM-GCN model consists of three parts: the left part DeepFM is used to capture the interactive information between the users and items; the deep neural network is used in the middle to model the left and right parts; and the right one obtains a better item representation vector by the GCN. In an effort to verify the validity and precision of the model built in this research, and based on the public datasets ml1m-kg20m and ml1m-kg1m, a performance comparison experiment was designed. It used multiple comparison models and the MKR and FM_MKR algorithms as well as the DFM-GCN algorithm constructed in this paper. Having achieved a state-of-the-art performance, the experimental results of the AUC and f1 values verified by the CTR as well as the accuracy, recall, and f1 values of the top-k showed that the proposed approach was excellent and more effective when compared with different recommendation algorithms.

Keywords: DeepFM; GCN; knowledge graph; DNN; representation learning; recommendation systems

MSC: 68T07

Citation: Xiao, Y.; Li, C.; Liu, V. DFM-GCN: A Multi-Task Learning Recommendation Based on a Deep Graph Neural Network. *Mathematics* **2022**, *10*, 721. https://doi.org/10.3390/math10050721

Academic Editors: Andrea Prati, Carlos A. Iglesias, Luis Javier García Villalba and Vincent A. Cicirello

Received: 6 January 2022
Accepted: 18 February 2022
Published: 24 February 2022

1. Introduction

Recommendation systems are important research directions in the field of artificial intelligence. Many experts and scholars have worked extensively with recommendation systems, among which the collaborative filtering algorithm is the basic algorithm of most advanced models. However, collaborative filtering suffers from the cold start of the user and the problem of sparse data. Therefore, researchers have proposed many algorithms to improve collaborative filtering.

Additional auxiliary information can be used to solve the problems of data sparseness and cold starts in the collaborative filtering algorithm. For example, Jamali et al. [1] introduced social network information to provide compensation recommendations. Wang et al. [2] introduced item attribute information to solve the cold start problem. Wang et al. [3] and Zhang et al. [4] used pictures and multimedia for presentation learning to supplement the item information. The research of the above experts and scholars all achieved a state-of-the-art performance at that time. The MKR algorithm proposed by Wang et al. [5] had excellent results.

The network structure of the MKR algorithm is mainly composed of three parts. On the left is the collaborative filtering recommendation module. The middle part is the cross-compression unit, mainly used to combine the results of the left and right parts; the knowledge graph is then used to learn the entity representation to supplement the item semantics in collaborative filtering. On the right is the TransE knowledge graph representation module. However, the MKR method has the problem of semantic compression caused by data compression and dimension transformation in the collaborative filtering, especially in the knowledge graph representation learning part, which cannot fully learn the semantic expression of an item. To improve the MKR algorithm, we proposed the DFM-GCN. The FM (factorization machine) was replaced by a DeepFM (deep factorization machine) to extract the higher level information. The DeepFM algorithm effectively combined the advantages of factorization and deep learning whilst extracting low order and high order combination features that could effectively avoid dimension transformation and data compression problems. Instead of TransE, a GCN (graph convolutional network)—a type of graph representation learning method that has powerful performance in many applications—was applied to encode the item embedded in the knowledge graph. We deepened the cross-compression unit to unleash the semantic information from the compressed data.

The main contributions of this paper are summarized as follows:

(1) The DeepFM algorithm was used as the backbone because it can compensate for the dimension transformation and data compression problems brought by the collaborative filtering algorithm.
(2) The GCN was employed to learn the representation of each node and relationship in the knowledge graph, which could inject more reliable item knowledge into the recommendation algorithm.
(3) Using deep neural networks to model the connection between DeepFM and the GCN, we adopted a deep cross-compression unit to capture the high dimension features in the interaction.

1.1. Existing Related Works

Previous scholars have undertaken a great deal of research on recommendation systems. These can be summarized into three major categories: traditional recommendation algorithms, recommendation algorithms based on deep learning, and fusion recommendation algorithms that rely on external knowledge. Traditional recommendation systems include recommendation algorithms based on collaborative filtering [6,7] (user-based collaborative filtering, content-based collaborative filtering) and hybrid recommendation algorithms. Recommendation algorithms based on deep learning include deep neural networks (DNNs [8]), convolutional neural networks (CNNs [9]), recurrent neural networks (RNNs [10]), long short-term memory neural networks (LSTMs [11]), and graph neural networks (GNNs [12]). Fusion recommendation algorithms, with the help of external knowledge, include fusion knowledge graphs, images, and label information.

1.1.1. Traditional Recommendation Algorithms

Traditional recommendation system algorithms include three categories: collaborative filtering-based recommendation algorithms (user-based collaborative filtering) (Figure 1a); content-based collaborative filtering (Figure 1b); and hybrid recommendation algorithms.

Content-based collaborative filtering was the first recommendation algorithm proposed. The idea was to recommend items that were similar to the past interests of the users; the user had to analyze the item information that they were interested in first. Content-based recommendations need to build recommendations based on items that users are interested in based on their history. The advantage of this method was that it could generate effective recommendations for users with niche tastes and there was no problem of a project cold start. However, this method was very dependent on the attribute characteristics of the

item and had extremely high requirements for the marking characteristics. Additionally, it was difficult to measure the pros and cons of the items being recommended.

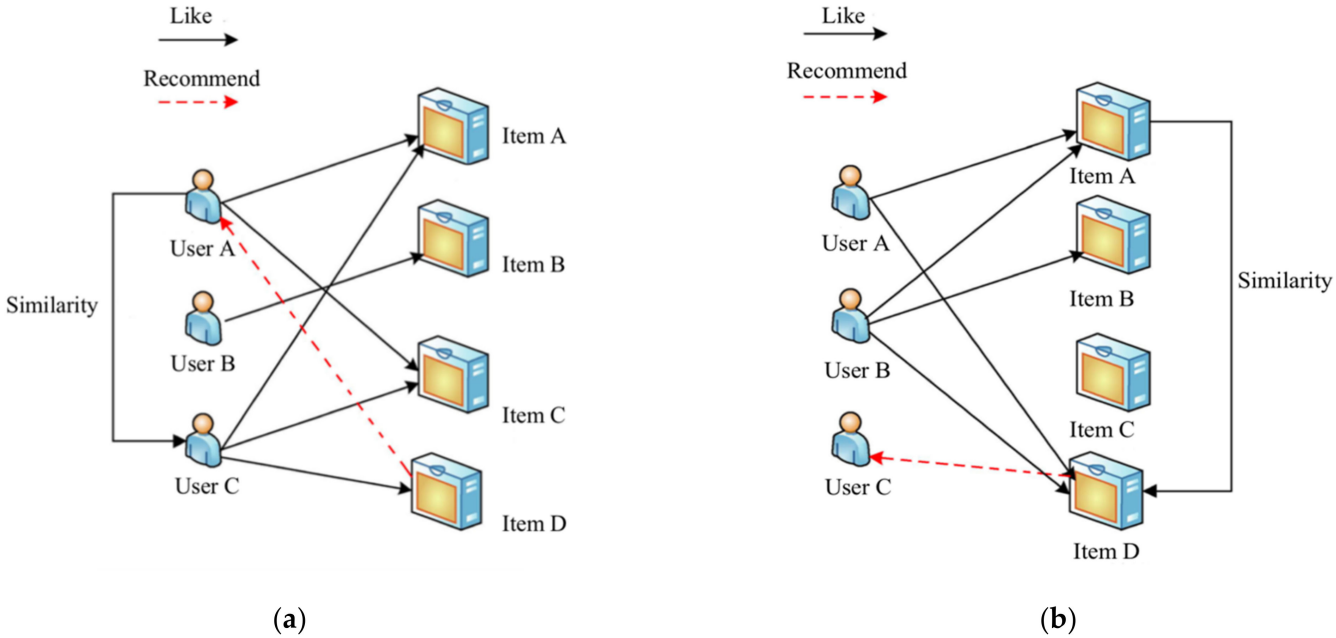

Figure 1. The ideas of the collaborative filtering algorithm. (**a**) User-based collaborative filtering algorithm; (**b**) content-based collaborative filtering algorithm.

The user-based collaborative filtering calculated the highest similarities between the users according to their first preferences and interests. It then calculated the highest scored item to be recommended to the target user. The collaborative filtering recommendation method was convenient and simple to use; only the similarity between users needed to be calculated, based on the historical rating data of the user. However, this method often encounters the problem of insufficient scoring data. Therefore, in the actual use process, a sparse matrix and cold start problems of new users without item scoring data were generated.

Hybrid recommendation algorithms refer to systems that combine two or more actual situations, thus achieving better results for the user. There are many mixed recommendation methods such as weighted, switched, crossed, feature combination, waterfall, feature incremental, and meta-level. To an extent, these methods can achieve the purpose of complementing each other but not every combination of methods has practical effects for specific problems.

1.1.2. Recommendation Algorithms Based on Deep Learning

Recommendation algorithms based on deep learning include recommendation methods based on deep neural networks (DNNs), convolutional neural networks (CNNs), recurrent neural networks (RNNs), long and short-term memory neural networks (LSTMs), and graph neural networks (GNNs). In recommendation systems, the deep learning extracts the potential features of the user and the potential features of the item, then recommends generated items for the users. The neural network can not only learn the latent feature representations of users or items but also the complex nonlinear interaction characteristics between the users and items; it can deeply analyze user preferences, which can solve several problems in traditional recommendation methods to better implement the recommendations.

Deep neural networks (DNNs) have been widely used in computer vision, image classification, natural language processing, and other fields. The first use of a DNN in a recommendation field was in YouTube video recommendations. Chen et al. [13]

considered the large data size, high data freshness, and large noise characteristics of the YouTube website and then proposed the use of a DNN to achieve efficient recommendations. The system architecture is shown in Figure 2. The biggest advantage of the DNN-based recommendation algorithm is that the input of the DNN can easily handle discrete and continuous variables. However, when calculating the distance between the user vector and the item vector, it may not fully reflect the similarities. In addition, the learned user vector and the item vector may have a distribution deviation. In this way, Zhang et al. [14] proposed a model that combined a collaborative filtering recommendation algorithm with a DNN. This model improved the traditional matrix factorization algorithm and used a quadratic polynomial regression model to capture the potential feature representations, making the potential feature representations obtained by the model more precise. However, this method suffered from a user cold start.

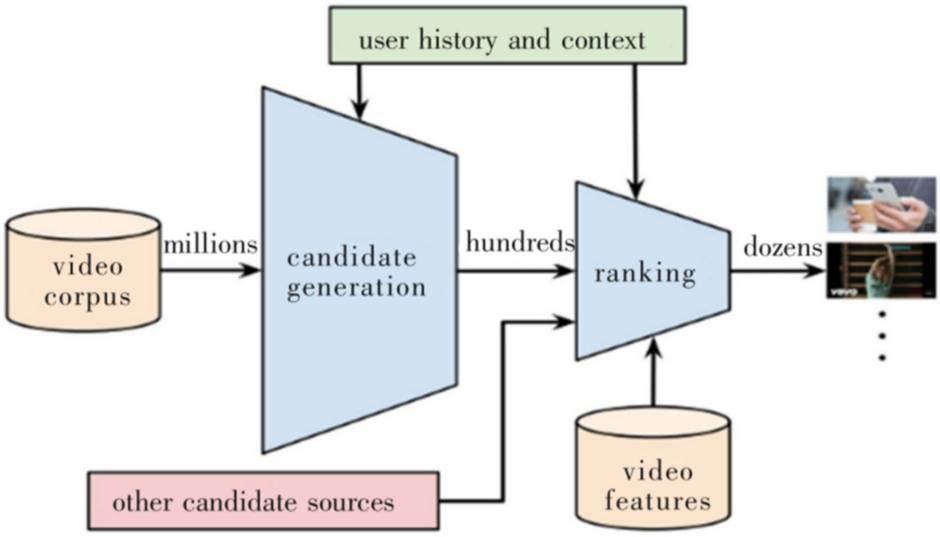

Figure 2. Youtube video recommendation model based on a DNN.

A CNN can achieve operations such as sharing weights and local connections. In addition, a CNN has a strong fault tolerance and robustness [15] and is easy to train and optimize. Tang et al. [16] proposed a convolution-based sequence embedding the Caser model. The Caser model solved the problem where the nearest item within the top-N sequence recommendation had a greater impact on the next item. Kim et al. [17] proposed a novel context-aware recommendation model that integrated a CNN into a probabilistic matrix factorization (PMF) and a convolutional matrix factorization (ConvMF). Experiments have proven that a ConvMF can capture the context information of a document, thereby improving the prediction accuracy of the scores.

An RNN is inherently capable of processing time series data. Santos et al. [18] proposed a context-aware recurrent neural network (CA-RNN) based on an RNN that could model rich context and sequence information, thus improving the text semantic capture to improve the recommendation effect. Manotumruksa et al. [19] proposed a novel contextual attention loop architecture (CARA) to solve the problem that different types of contexts have different effects on user preferences. Yang et al. [20] proposed a context-aware citation recommendation model based on long short-term memory to recommend relevant and appropriate scientific paper citations for users.

Several NLP models based on transformers have become very popular recently such as BERT and GPT-2. It has become a recent trend to introduce transformer models into recommendation algorithms. Transformers can better model user behavior sequences than traditional LSTMs, GRUs, and other models. Kang et al. [21] introduced a two-layer transformer decoder (i.e., transformer language model) called SASRec to capture the sequential behaviors of users and achieved state-of-the-art results on several public datasets. Sun et al. [22]

proposed BERT4Rec, which modeled the user behavior sequence based on bidirectional self-attention and achieved a state-of-the-art sequential recommendation performance.

1.1.3. Recommendation Algorithms Based on a Graph Neural Network

A graph neural network can capture the dependency of graphs through a message passing between nodes. The recommendation systems based on a GNN regard items and users as nodes and the relationships between items and items, users and users, users and items, and content information as node status information. Most prominent among these recent advancements is the success of deep learning architectures known as graph convolutional networks (GCNs). The main idea behind GCNs is to learn how to iteratively aggregate the feature information from local graph neighborhoods using neural networks. Ying et al. [23] proposed a scalable GCN algorithm, PinSage. The PinSage model used a local convolution, which reduced the complexity of the model during the training calculations and could improve the recommendation effect. Wang et al. [24] proposed an end-to-end framework of KGNN-LS, which could effectively capture the correlation between items by mining the related attributes between the items in the knowledge graph.

1.1.4. Recommendation Algorithms Incorporating External Knowledge

Recommendation algorithms that integrate external knowledge include recommendation algorithms that integrate knowledge graphs, images, and label information. The introduction of external knowledge is mainly to solve the problems of a sparse matrix and cold starts in the recommendation systems.

In several of the above methods, only the historical interaction information (explicit or implicit feedback) between the user and the item are used as an input. This causes two problems: (1) the interaction information between the user and the item is often very sparse and, therefore, it is easy for a sparse matrix to appear; and (2) for new users or items, because the system has no historical interaction information, it cannot be accurately modeled and recommended. This is also known as the cold start problem. Therefore, the recommendation algorithm that integrates external knowledge can solve the above two problems to a certain extent.

Jamali et al. [1] embedded social network information into a graph in order to compensate for a lack of expressed user interest. Zhang et al. [4] constructed recommendation algorithms by learning the representation of multimedia, pictures, and other information to better express items through the introduction of picture information. Wang et al. [5] proposed network structures such as MKR and Ripplenet by introducing additional auxiliary information such as knowledge graphs to compensate for the representation of the items to achieve a better recommendation effectiveness.

A large number of studies by predecessors and scholars have shown that the cold start and data sparsity of recommendation systems are urgent problems to be solved and that the recommendation algorithm that integrates external knowledge also plays an important role, especially the recommendation algorithm integrating the knowledge graph. Currently, a recommendation algorithm fused with knowledge graphs has the problem of semantic compression caused by data compression and dimension transformation in the collaborative filtering, especially in the knowledge graph representation learning part; the semantic expression of items cannot be fully learned and needs to be strengthened in terms of item representation and the interaction between graphs and recommendations. In this paper, we focused on the cold start and data sparsity problems appearing in the recommendation systems. Based on an MKR [5], we proposed the DFM-GCN algorithm to inject the knowledge into the recommendation systems for mitigating the cold start and data sparsity problems.

2. Method

2.1. Problem Definition

In the recommendation systems, we set up a group of users $U = \{u_1, u_2 \dots u_M\}$ and a set of items $I = \{i_1, i_2 \dots i_N\}$, where M and N were the number of users and items respectively. The items could refer to commodities, movies, or televisions. We then defined the interaction matrix of user-item $Y \in \mathbb{R}^{M \times N}$, where Y represented the implicit feedback of the user on the item. If $y_{ui} = 1$, this indicated that the user liked, viewed, or clicked on the item. If $y_{ui} = 0$, this indicated that the user did not interact with the corresponding item.

In this paper, we introduced the knowledge graph for improving the recommendations. The knowledge graph G = (head, relation, tail) contained a large number of triples, which refers to the connection between two entities. For example, (Christopher Nolan, film.film.director, Tenet) indicated that the director of the film "Tenet" was Christopher Nolan. In the actual recommendations, if the users liked Tenet, they may also like other movies made by Christopher Nolan. That is, the knowledge graph could obtain a better representation for each item, playing a cohesive role in the entire recommendation. Therefore, given a set of users U and items I fused with knowledge graph G, this paper focused on recommending an item $i \in I$ for $u \in U$ with the help of item knowledge.

2.2. Model

The model structure mainly consisted of three parts. One was the DeepFM recommendation on the left half where FM was used to capture the low order features and a deep network was added to capture the high order features between the user and item. The other was GCN knowledge graph embedding to encode the knowledge graph on the right half. This part used a GCN to establish the connection between the items in order to obtain a more comprehensive item feature expression in the recommendation. The middle part used DNN cross and compressed units to connect the DeepFM and GCN. Among them, the DNN was mainly used to capture the high dimension features. This unit could automatically learn the high level feature interactions between the items in the recommendation systems and entities in the knowledge graph. The overall network structure is shown in Figure 3.

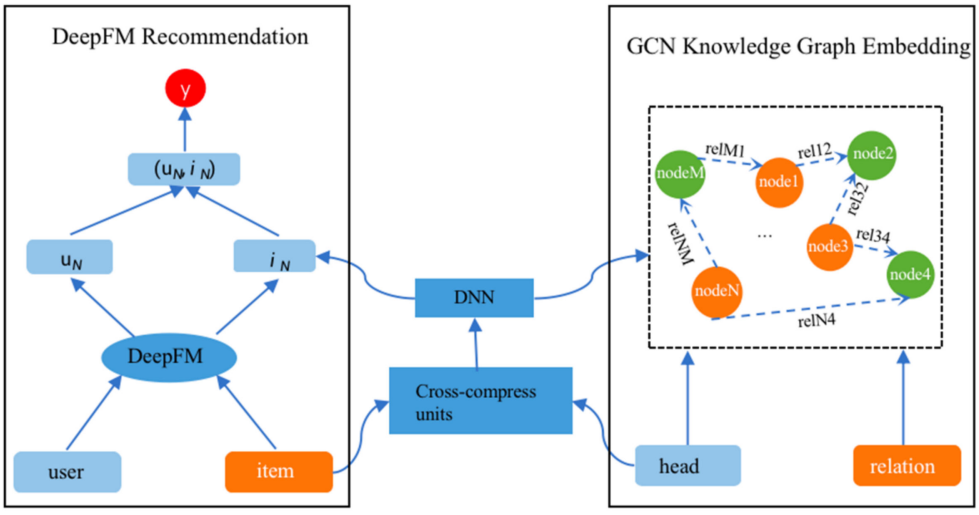

Figure 3. The DFM-GCN network structure model of recommendation systems.

The input of the entire model was user, item, and knowledge graph triples; the output was the probability ranking of the preferences of the users for a series of items.

The embedding layer included three parts, which were user feature embedding, user behavior sequences, and item feature embedding. The user embedding was obtained by inputting into the DeepFM model. The item embedding came from two parts; one was the DeepFM and the other was obtained by the GCN using relational data. The two modules

were fused through the DNN module to obtain the embedding of final item. The schematic diagram is shown in Figure 4.

2.2.1. DeepFM Recommendation

The DeepFM algorithm effectively combined the advantages of factorization and deep learning whilst extracting low order and high order combination features. In the DeepFM framework, the FM was mainly responsible for the extraction of first-order features and second-order features (the pairwise combination of first-order features) and the DNN was responsible for extracting the high order features.

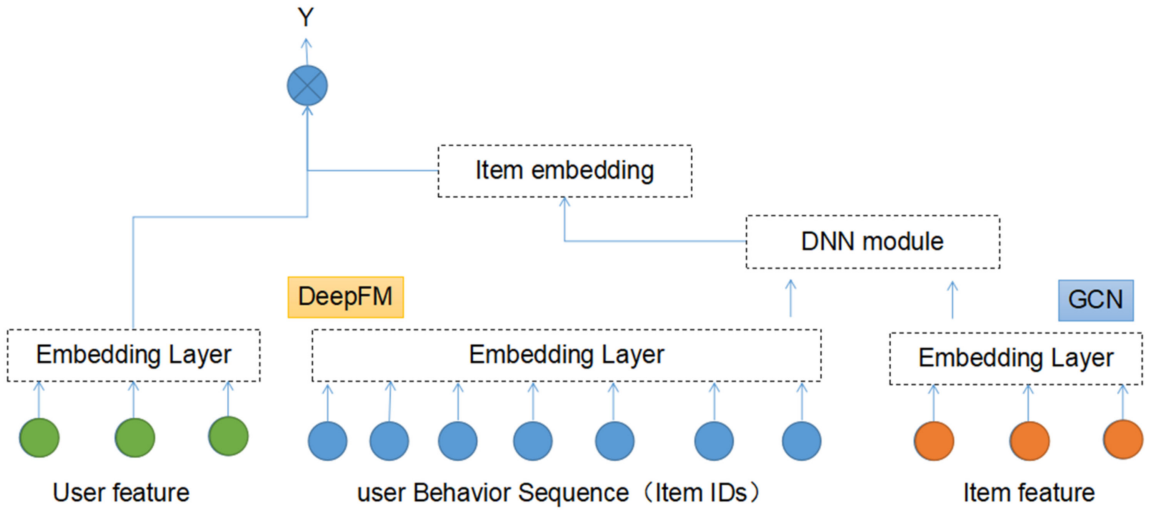

Figure 4. The diagram of user embedding and item embedding.

The mathematical expression of the commodity feature i_{FM} extracted by the FM factorization model is as follows:

$$i_{FM} = \omega_0 + \sum_{i=1}^{n} \omega_i v_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \omega_{ij} v_i v_j \tag{1}$$

where ω is the parameter that the model needs to learn, v_i is the value of the product feature representing the i-th dimension, and v is the initial feature vector representation of the product.

The DNN was responsible for constructing the high dimensional feature representation of the product. Its input shared the product feature vector representation v with the FM; the high dimensional feature representation $i_{DNN}^{(l)}$ of the l-th level product was then as follows:

$$i_{DNN}^{(l)} = \sigma\left(W^{(l)} i_{DNN}^{(l-1)} + b^{(l)}\right) \tag{2}$$

where l is the layer depth and σ is the nonlinear activation function such as ReLU. $W^{(l)}$, $b^{(l)}$ are the model weight and bias of the l-th layer, respectively, which it needs to learn. After that, a dense real value feature vector was generated. Thus, the final representation of the product $i_N = i_{FM} + i_{DNN}$, after having the latent feature vector u_N of the user u and the latent feature i_N of the item v, the expression of the recommended score y was as follows:

$$y = \text{sigmoid}\left(u_N^T \cdot i_N\right) \tag{3}$$

where we could obtain the latent feature vector U_N of u through the DNN module, which was used for the high order feature combinations. The value of the second-order item was equal to the embedding vector product of the two features so the user embedding matrix

and the transpose of the item embedding matrix could be multiplied to obtain the user-item "rating" matrix and then recommend products.

2.2.2. Embedding of the GCN Knowledge Graph

The knowledge graph embedding representation aims to transform entities and relationships into a continuous vector space whilst maintaining the structure between them. In recent years, researchers have proposed many knowledge graph embedding technologies, including Trans series models and semantic depth matching models. In this paper, we used a GCN to embed the knowledge graph. A GCN can be regarded as a message transmission framework and its calculation expression is as follows:

$$h_i^{l+1} = \sigma\left(\sum_{j \in M_i} g_m\left(h_i^l, h_j^l\right)\right) \tag{4}$$

where h_i^l is the feature representation of node v_i in the l-th layer, $g_m(\cdot,\cdot)$ represents the message aggregation function, and σ is the activation function. Specifically, the standard GCN model $g_m\left(h_i^l, h_j^l\right) = Wh_j$ for knowledge graph embedding not only has node (entity) features but also edge (relation) features. Therefore, the node feature encoding expression of the GCN could be converted into the following form:

$$h_i^{l+1} = \sigma\left(\sum_{r \in R} \sum_{j \in N_i^r} \frac{1}{c_{i,r}} W_r^l h_j^l + W_0^l h_i^l\right) \tag{5}$$

where N_i^r represents the set of neighbor nodes of node i under relation r and $c_{i,r}$ is the trainable parameter. From the above formula, it could be seen that the features of each layer of nodes were obtained from the features and relationships (edges) of the previous layer. The neighbors of the nodes and their own features were weighted to obtain new features. The feature encoding of the edge (relation) was as follows:

$$r^{l+1} = Wr^l \tag{6}$$

where W is the learnable parameter and r^l is the feature vector representation of the relationship on the edge of the l layer. After splicing the relationship features of the nodes and edges, the fully connected layer was then used to predict the tail entity representation:

$$\hat{t} = N\begin{bmatrix} h \\ r \end{bmatrix} \tag{7}$$

where N is the learnable parameter and $\hat{t}$ is the vector representation of the tail entity t predicted for the head entity h and the relationship r.

2.2.3. DNN Cross and Compress Units

This module is mainly used to model the interaction features between commodities and their corresponding entities in the knowledge graph. For each commodity v and its corresponding entity e, the interaction feature matrix C between them is as follows:

$$C = v \cdot e^T = \begin{bmatrix} v_0 e_0 & \cdots & v_0 e_{n-1} \\ \cdots & \cdots & \cdots \\ v_{n-1} e_0 & \cdots & v_{n-1} e_{n-1} \end{bmatrix} \tag{8}$$

where $v \in \mathbb{R}^{n \times 1}$, $e \in \mathbb{R}^{n \times 1}$, and $C \in \mathbb{R}^{n \times n}$. This process is called a cross-operation because each feature interaction $v_i e_j$ between the product and its corresponding entity is displayed

and modeled in the interaction feature matrix C. We could then obtain the commodity and entity feature representations $\overline{v}$ and $\overline{e}$ after the interaction:

$$\overline{v} = DNN\left(CW_1 + C^T W_2 + b_v\right) \tag{9}$$

$$\overline{e} = DNN\left(CW_3 + C^T W_4 + b_e\right) \tag{10}$$

where W_1, W_2, W_3, W_4, b_v, and b_e are trainable parameters and the function of DNN represents the deep neural network such as a multi-layer fully connected network. The forward process was:

$$DNN(\mathrm{i}^{(l)}) = \sigma\left(\mathrm{W}^{(l)}\mathrm{i}^{l-1} + \mathrm{b}^{(l)}\right) \tag{11}$$

where l is the layer depth and σ is the nonlinear activation function. $DNN(\mathrm{i}^{(l)})$, $\mathrm{W}^{(l)}$, and $\mathrm{b}^{(l)}$ are the output, model weight, and the bias of the l-th layer, respectively. This process is called a compress operation and compresses a two-dimensional matrix into a one-dimensional vector representation, which then extracts high order features through a deep neural network.

2.2.4. Optimized Objective Function

The complete loss function contained two parts: the depth recommendation loss $\mathcal{L}_{rec}$ and the knowledge graph embedding loss $\mathcal{L}_{graph}$. The function expression of $\mathcal{L}_{rec}$ was as follows:

$$\mathcal{L}_{rec} = \mathrm{MSE}\left(y, y_{gold}\right). \tag{12}$$

The expression of knowledge graph embedding loss $\mathcal{L}_{graph}$ was as follows:

$$\mathcal{L}_{graph} = \mathrm{sigmoid}\left(t^T \hat{t}\right) \tag{13}$$

where t is the real tail entity and $\hat{t}$ is the prediction of the entity t in which we aimed to increase the score for all true triples whilst reducing the score for all false triples.

The final optimization objective (loss) function was then as follows:

$$\mathcal{L} = \mathcal{L}_{rec} + \mathcal{L}_{graph}. \tag{14}$$

Training Complexity. The time cost of our proposed DFM-GCN mainly arose from three parts: the DeepFM, the DNN cross and compress units, and the GCN. The time complexity of the DeepFM was O(n*latent_factor), where n is the number of features. For the GCN, the time complexity was O(K | E |), where K is the number of parameters and E is the number of edges in network G. For the DNN cross and compress units, the time complexity was O(l), where l is the layer depth. Therefore, the total complexity of the DFM-GCN was O(K | E | +n*latent_factor+l).

3. Experiment and Analysis

In order to verify the effectiveness of our proposed model, two public datasets were used for ablation experiments. The baseline was the MKR algorithm, using the AUC and f1 score estimated by the CTR; the AUC represented the area under the curve and the AUC was equivalent to the probability of the positive samples being ranked higher than the negative samples. We also used the precision, recall, and f1 score of the top-k to verify the experimental results, in which the precision was for the prediction results as it indicated how many of the predicted positive samples were true positive samples. The recall rate was for the original sample and it indicated how many positive examples in the sample were correctly predicted.

3.1. Datasets

The dataset used in this article was the public dataset MovieLens, which contains two versions, ml1m_kg20m and ml1m_kg1m. The details of the data are shown in Table 1. The format of the dataset was as follows. The dataset included three files: ratings.dat, the rating file of the user for items that consisted of approximately 1 million explicit ratings (rating from 1 to 5) on the MovieLens website; kg.txt, the triple file of the knowledge graph; and item_id2entity_id.txt, which mapped the item ID to the entity ID.

Table 1. Basic statistics of the datasets.

	Ml1m_kg1m			**ml1m_kg20m**		
	UserID	**MovieID**	**Score**	**UserID**	**MovieID**	**Score**
Count	1,000,209	1,000,209	1,000,209	20,000,263	20,000,263	20,000,263
Mean	3024.51	1865.54	3.58	69,045.87	9041.57	3.53
Std	1728.41	1096.04	1.12	40,038.62	19,789.48	1.05
Min	1	1	1	1	1	0.5
25%	1506	1030	3	34,395	902	3
50%	3070	1835	4	69,141	2167	3.5
75%	4476	2770	4	103,637	4770	4
Max	6040	3952	5	138,493	131,262	5

In the dataset of ml1m_kg1m, there were 6036 users, 2347 items, 753,772 interactions, 20,195 KG triples, and 6729 entities. It contained seven types of relationship data such as actor, director, genre, and country in the dataset. This dataset (ml1m_kg20m) described a five-star rating and free-text tagging activity from MovieLens, a movie recommendation service. It contained 20,000,263 ratings and 465,564 tag applications across 27,278 movies. The ratio of the training, development, and test set was 6:2:2.

3.2. Baselines

In this paper, we set up three sets of comparative experiments with different network structures. The baseline used the MKR algorithm, the second set of experiments used the FM_MKR algorithm, and the third set of experiments was our algorithm based on the DFM-GCN.

3.3. Experiment Settings

The hyper-parameter settings mainly included the following: rs_lr (the learning rate of the recommended module); kge_lr (the learning rate of the knowledge graph encoding learning); batchsize; and epoch. In the experiment, the hyper-parameters we set were: rs_lr = 0.02, kge_lr = 0.01, batchsize = 4096, epoch = 20.

The evaluation metrics used the AUC and f1 values estimated by the CTR as well as the accuracy, recall, and f1 values of the top-k.

3.4. Experiment Results

The experimental results of all methods on the two datasets are shown in Tables 2 and 3.

Table 2. The results of the AUC and f1 on CTR prediction based on the datasets of ml1m-kg20m.

	AUC	**F1**	**Top 100 (Precision)**	**Top 100 (Recall)**	**Top 100 (F1)**
MKR	0.82777	0.66641	8.080	20.076	11.523
FM_MKR	0.89637	0.82169	10.760	43.793	17.275
DFM + GCN	0.91435	0.8441	20.693	49.364	29.162

Table 3. The results of the AUC and f1 on CTR prediction based on the datasets of ml1m-kg1m.

	AUC	F1	Top 100 (Precision)	Top 100 (Recall)	Top 100 (F1)
MKR	0.83786	0.65432	8.981	22.163	12.782
FM_MKR	0.91123	0.84011	12.364	43.865	19.291
DFM + GCN	0.91781	0.84773	21.003	49.847	29.554

It can be seen from the tables that our proposed method, DFM-GCN, achieved the best performance on the two datasets. Due to the data compression of the collaborative filtering part and the semantic compression caused by the dimension transformation in the MKR algorithm, parts of the original information of the item could be lost. Therefore, the vector representation among the items was confused during the recommendation process. The recommendation accuracy on the item was the worst in the comparison model. FM_MKR improved the modeling process of the MKR recommendation model and could better capture the low dimensional features so the performance significantly improved, about 7% on the AUC score compared with the MKR. However, the DFM-GCN used the GCN to learn the representation of the knowledge graph, which solved the problem of insufficient node representation in the TransE framework, and DeepFM could better model the product features so the DFM-GCN achieved a state-of-the-art performance with an improvement of about 8% on the AUC score and 18% on the f1 score compared with the baseline model MKR.

In order to further explore the performance changes of the DFM-GCN during training, we drew the learning curves on a test set of the two datasets, as shown in Figures 5–7.

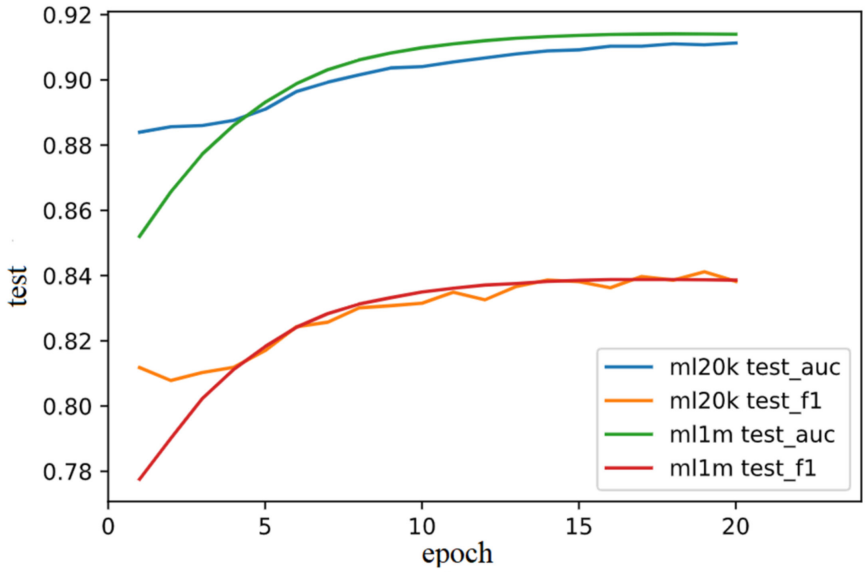

Figure 5. A reflection of the change in the AUC and f1 score of the CTR on the test set.

Figure 5 shows the learning curves of the AUC and f1 score on the test set. We observed that the performance of our model gradually increased on the two datasets as the training continued. Until the max epoch, the AUC and f1 score tended to be stable, which justified the rationality of the epoch hyper-parameter selection and avoided the over-fitting problem. In addition, the AUC and f1 score on the ml20m dataset achieved a fast convergence in the early iterations whereas they did not converge until about the 7th epoch on the ml1m dataset. The reason may be that ml20m possessed less data than ml1m. Therefore, our model could also achieve a stable performance on few numbers of the datasets.

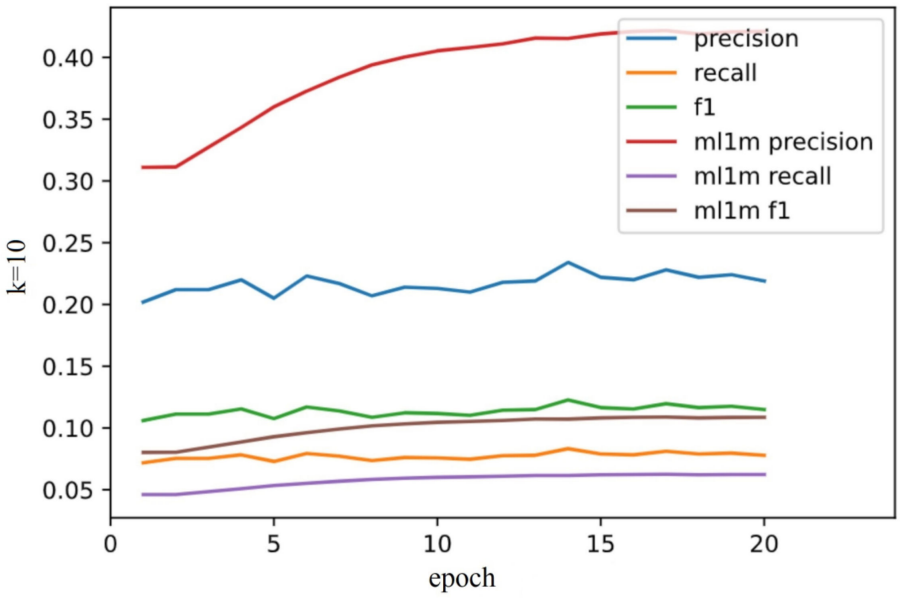

Figure 6. A reflection of the changes in precision, recall, and f1 of the Top 10 on the test set.

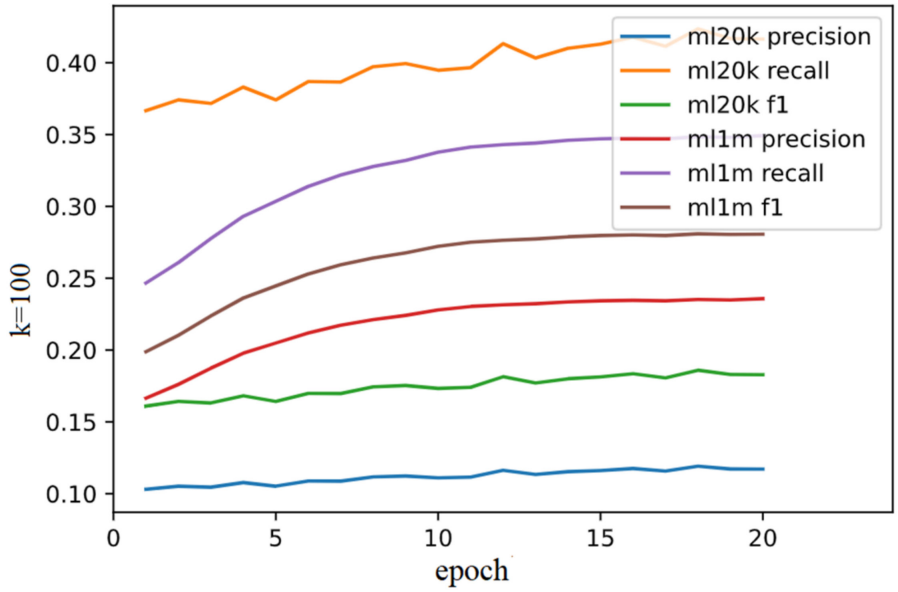

Figure 7. A reflection of the changes in precision, recall, and f1 of the Top 100 on the test set.

As shown in Figures 6 and 7, we reported the precision, recall, and f1 score learning curves of the Top 10 and Top 100 on the test set. We observed that precision was higher than recall consistently on the Top 10 whereas recall was higher than precision on the Top 100, indicating that the golden item most existed in the Top 100 candidate set. The learning curves for the Top 10 were more gentle than the Top 100 in the early steps, which showed that the model mainly focused on recalling more reliable candidate items.

4. Conclusions

This paper mainly focused on improving the MKR algorithm and proposed the DFM-GCN framework for its recommendation. The DFM-GCN comprised the DeepFM recommendation, GCN knowledge graph embedding, and DNN cross and compress units. We adopted the DeepFM to extract the higher level information and the GCN was applied to encode the item embedded in the knowledge graph. In addition, we made the cross-compression unit deeper for unleashing the semantic information from the compressed

data. Therefore, knowledge information could be injected into the recommendation systems to alleviate the problem of data sparseness and a cold start. The experimental results demonstrated that our proposed method achieved a state-of-the-art performance. For future work, the alignment between the item embedding in the recommendation and the entity representation in the knowledge graph is worth exploring for unleashing the knowledge graph.

Author Contributions: Conceptualization, C.L. and Y.X.; methodology, C.L. and V.L.; validation, C.L., Y.X. and V.L.; investigation, V.L.; data curation, V.L.; writing—original draft preparation, Y.X.; writing—review and editing, Y.X.; visualization, V.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: This study does not involve humans or animals research.

Informed Consent Statement: This research does not involve human-related research.

Data Availability Statement: Due to data privacy reasons, this study does not disclose the experimental data information for the time being.

Acknowledgments: The authors would like to acknowledge partial financial support from the National Natural Science Foundation of China, No. 71672074; National Natural Science Foundation of China, No. 72072072; Natural Science Foundation of Guangdong Province of China, No. 2019A1515010045; and 2018 Guangzhou Leading Innovation Team Program (China), No. 201909010006.

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

References

1. Jamali, M.; Ester, M. A matrix factorization technique with trust propagation for recommendation in social networks. In Proceedings of the Fourth ACM Conference on Recommender Systems, Barcelona, Spain, 26–30 September 2010; pp. 135–142.
2. Wang, H.; Zhang, F.; Hou, M.; Xie, X.; Guo, M.; Liu, Q. SHINE: Signed Heterogeneous Information Network Embedding for Sentiment Link Prediction. In Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, Marina Del Rey, CA, USA, 5–9 February 2018; pp. 592–600.
3. 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, Sydney, Australia, 10–13 August 2015.
4. Zhang, F.; Yuan, N.J.; Lian, D.; Xie, X.; Ma, W.-Y. Collaborative Knowledge Base Embedding for Recommender Systems. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 353–362.
5. Wang, H.; Zhang, F.; Zhao, M.; Li, W.; Xie, X.; Guo, M. Multi-Task Feature Learning for Knowledge Graph Enhanced Recommendation. In Proceedings of the World Wide Web Conference, San Francisco, CA, USA, 13–17 May 2019; pp. 2000–2010.
6. Su, X.; Khoshgoftaar, T.M. A Survey of Collaborative Filtering Techniques. *Adv. Artif. Intell.* **2009**, *2009*, 421425. [CrossRef]
7. Xiao, Y.; Li, C.D.; Song, L.J.; Yang, J.; Su, J.F. A Multidimensional Information Fusion-Based Matching Decision Method for Manufacturing Service Resource. *IEEE Access* **2021**, *9*, 39839–39851. [CrossRef]
8. Bengio, Y. *Learning Deep Architectures for AI*; Foundation and Trends® in Machine Learning: Hanover, MA, USA, 2009; Volume 2, pp. 1–127.
9. Dezfouli, P.A.B.; Momtazi, S.; Dehghan, M. Deep neural review text interaction for recommendation systems. *Appl. Soft Comput.* **2020**, *100*, 106985. [CrossRef]
10. Donkers, T.; Loepp, B.; Ziegler, J. Sequential User-based Recurrent Neural Network Recommendations. In Proceedings of the Eleventh ACM Conference on Recommender Systems, Como, Italy, 27–31 August 2017; pp. 152–160.
11. Sahoo, B.B.; Jha, R.; Singh, A.; Kumar, D. Long short-term memory (LSTM) recurrent neural network for low-flow hydrological time series forecasting. *Acta Geophys.* **2019**, *67*, 1471–1481. [CrossRef]
12. Scarselli, F.; Gori, M.; Tsoi, A.; Hagenbuchner, M.; Monfardini, G. The Graph Neural Network Model. *IEEE Trans. Neural Netw.* **2009**, *20*, 61–80. [CrossRef] [PubMed]
13. Chen, M.; Beutel, A.; Covington, P.; Jain, S.; Belletti, F.; Chi, E.H. *Top-K Off-Policy Correction for a REINFORCE Recommender System*; ACM: New York, NY, USA, 2019; pp. 456–464. [CrossRef]
14. Zhang, L.; Luo, T.; Zhang, F.; Wu, Y. A Recommendation Model Based on Deep Neural Network. *IEEE Access* **2018**, *6*, 9454–9463. [CrossRef]
15. Liu, J.; Yuan, Q.; Wu, G.; Yu, X. Review of convolutional neural networks. *Comput. Era* **2018**, 19–23.
16. Tang, J.X.; Wang, K. *Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding*; ACM: New York, NY, USA, 2018; pp. 565–573. [CrossRef]

17. 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.
18. Santos, J.F.; Falk, T.H. Speech Dereverberation with Context-Aware Recurrent Neural Networks. *IEEE ACM Trans. Audio Speech Lang. Process.* **2018**, *26*, 1232–1242. [CrossRef]
19. Manotumruksa, J.; Macdonald, C.; Ounis, I. A Contextual Attention Recurrent Architecture for Context-Aware Venue Recommendation. In Proceedings of the The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, Ann Arbor, MI, USA, 8–12 July 2018; pp. 555–564.
20. Yang, L.B.; Zheng, Y.; Cai, X.Y.; Dai, H.; Mu, D.J.; Guo, L.T.; Dai, T. A LSTM Based Model for Personalized Context-Aware Citation Recommendation. *IEEE Access* **2018**, *6*, 59618–59627. [CrossRef]
21. Kang, W.-C.; McAuley, J. Self-attentive sequential recommendation. In Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM), Singapore, 17–20 November 2018; pp. 197–206.
22. Sun, F.; Liu, J.; Wu, J.; Pei, C.; Lin, X.; Ou, W.; Jiang, P. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing, China, 3–7 November 2019; pp. 1441–1450.
23. Ying, R.; He, R.; Chen, K.; Eksombatchai, P.; Hamilton, W.L.; Leskovec, J. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, London, UK, 19–23 August 2018; pp. 974–983.
24. Wang, H.; Zhao, M.; Xie, X.; Li, W.; Guo, M. Knowledge Graph Convolutional Networks for Recommender Systems. In Proceedings of the World Wide Web Conference, San Francisco, CA, USA, 13–17 May 2019; pp. 3307–3313.